Optimization of Multi-Class Document Classification with Computational Search Policy

Khin Sandar Kyaw[1] and **Somchai Limsiroratana**[2]

ABSTRACT

In the era of internet communication, many electronic documents are distributed via website every split second. The research interest for the process of knowledge discovery has changed from working with traditional printed documents to processing online data such as online news document classification. Most of the online data is in text documents and therefore the optimization of multi-class document classification is becoming a challenge for society today. Traditional search policy for the feature selection process is degrading with exhaustive search for complex features in document classification. Therefore, meta-heuristic based computational search is also becoming good solution to overcome the problem of exhaustive search. The search policy of a computational algorithm can provide the global optimal solution using a random search approach and selected optimal features can support finding the optimal classification results. In this paper, Cuckoo optimization (CO), Firefly optimization (FO), and Bat optimization (BO) algorithms are observed to overcome the problem of multi-class document classification by applying their adaptive search policies for searching for the optimal feature subset in a feature selection process. In addition, J48 and support vector machine (SVM) classifiers are used to evaluate the quality of selected feature subsets. The results from the proposed system are compared with traditional Best First search (BFS) and Ranker search (RS) based classification results. Furthermore, the results analysis is performed using various measurements from the point of view of performance analysis and complexity cost. According to the experimental results, the proposed system can generate the good multi-class document classification results using our computational search policy.

Keywords: Informational Retrieval, Random Search, Meta-heuristic Algorithm, Bag of Features, Objective Function, Text Categorization

1. INTRODUCTION

Multi-class document classification is the supervised categorization to apply more than one label to a document according to the previous records of a training model. Nowadays, the trend for multi-class document classification has moved toward automation. Consequently, the technology for mining the data has also been adaptably by changing from simple to more sophisticated models for both prediction of classification areas and analysis in clustering areas (descriptive). Meanwhile, computational search policy is becoming a big challenge to enable the optimizing multi-document classification process to achieve optimal classification performance. Computational intelligence-based search policy uses Artificial Intelligence (AI). It can support employing computation power for solving very complex problems with various fitness functions according to application demands such as accuracy, complexity, etc. It involves adaptive mechanisms to perform intelligently with the capabilities of adaption to new situations, generalization, abstraction, discovery, and association, in complex problem areas. The computational intelligence consists of several meta-heuristic algorithms which are based on evolutionary computation (EC), swarm intelligence (SI), nature-inspired based intelligence, etc.

In document classification, feature selection should be considered as the most important criteria for distinguishing the label of a document correctly. Therefore, computational search policy is used to look for the optimal feature subset for optimizing the performance of multi-class document classification. To be specific, computational search policy defines objective functions such as accuracy and error rate to search for the features that can enhance the classification output. Our proposed model is implemented to achieve maximum accuracy (A) and minimum root mean squared error (RMSE) for multi-class document classification by using computational search policies for feature selection. In addition, the minimization of complexity cost, such as the selected number of global subset features (NGF) and building time for classification model (BCT), is improved with parameter tuning in the computation search algorithms. The proposed model includes several stages: feature extraction, feature selection, feature reduction, and building the learning model.

In feature extraction, the term frequency-inverse

Manuscript received on December 1, 2019 ; revised on March 23, 2020.

Final manuscript received on April 3, 2020.

[1,2] The authors are with Department of Computer Engineering, Prince of Songkla University, Thailand., E-mail: 5910130037@psu.ac.th and somchai.l@psu.ac.th

DOI: 10.37936/ecti-cit.2020142.227431

document frequency (TF-IDF) denotes the method used to calculate the weight of text features. Although there are many types of computational search, three new nature-inspired based meta-heuristic algorithms, Cuckoo Optimization (CO), Firefly Optimization (FO) and Bat Optimization (BO), are explored for a correlation-based feature selection approach because they can provide advanced nature search for non-linear complex problems. In addition, principal component analysis is used to compress the selected features by selecting principal components. Furthermore, two classification models, J48 and support vector machine (SVM), are used to evaluate the performance of document classification. In addition, the results of the proposed model for the rate of change of population size (PS) are compared with traditional search-based results. The proposed model achieved optimal performance classification results by reducing the number of selected features dramatically. In addition, the selected number of global subset features (NGF) according to the rate of change of population size is observed. Building times for the classification models (BCT) are evaluated using two different classifiers on individual NGF for each population size (PS).

2. RELATED WORK

Several related works for computational search-based optimization of complex feature selection for classification processes for various problem domains are described in this section. The purpose of reviewing the following related work is to point out the capability of different computational search algorithms, such as cuckoo, firefly, and bat algorithms, to solve the NP-hard feature selection problem for text data [1], medical data, etc.

In [2], Cuckoo search was proposed for optimization of the feature selection process in sentiment analysis. In addition, the baseline supervised learning model (SVM) is implemented. The proposed binary cuckoo search for optimization of the feature selection process outperforms the supervised algorithm with conventional TF-IDF score for the Kaggle dataset.

In [3], the authors proposed hybridizing of a mutual information feature selection (MIFS) filter and a modified binary Cuckoo search (MBCS) wrapper. In addition, the accuracy of K-nearest neighbor classifier is used as the fitness function. MBCS provided higher classification performance with efficient computational time by reducing the number of selected features.

In [4], a hybrid binary ant lion optimizer with rough set and approximate entropy reducts was proposed for the feature selection process. It included two incremental hill-climbing techniques which are hybridized with the binary ant lion algorithm called HBALO. A set of 18 well-known datasets from the UCI repository were used for testing the proposed model. Superior performance was attained in searching for the optimal features.

In [5], a Whale optimization (WOA) based wrapper feature selection approach was developed. Two binary variants of WOA algorithms were proposed using two pairs of operators to enhance the performance of the original Whale algorithm. The proposed system outperformed PSO, GA, ALO, and five standard filter approaches.

In [6], a variant of the Firefly algorithm was used in feature selection for classification and regression models. The proposed algorithm employs Simulated Annealing (SA) to achieve enhanced local and global promising solutions, and chaotic-accelerated attractiveness parameters and diversion mechanisms of weak solutions are used to escape from the local optimum trap and mitigate the premature convergence problem in the original FA algorithm. The results illustrate significant improvements over other state-of-the-art FA variants.

In [7], the authors proposed a Whale optimization algorithm for solving the high-dimensional, small instance feature selection problem. Two transfer functions, S-shaped and V-shaped, are used. The V-shaped system provided superior results to the S-shaped system on nine different high-dimensional medical datasets.

In [8], swarm intelligence based optimal feature selection was investigated to enhance the predictive accuracy of sentiment on Twitter. Two swarm-based algorithms such as binary grey wolf and binary moth flame are applied on two benchmark Twitter corpuses to observe the accuracy performance. Both proposed algorithms provided an improvement of accuracy with a significant reduction in the number of features.

In [9], the authors proposed a new Bacterial Colony optimization algorithm with multi-dimensional population (BCO-MDP) for classifying microarray gene expression cancers. The proposed method is shown to be superior to the binary algorithms in both feature size and efficiency. Moreover, lower computational complexity was obtained when compared to other population-based algorithms with constant dimensionality.

In [10], hybridizing firefly algorithms with a probabilistic neural network were proposed for solving a classification problem. Levy flight with SFA based firefly (LSFA) provided better classification accuracy performance than the SFA and LFA. In [11], a hybrid, multi-objective Firefly Algorithm (HMOFA) was explored for the optimization of big data. The proposed HMOFA provided promising performance on six single objective problems and six multi-objective problems.

In [12], a particle swarm optimization algorithm was proposed to improve the document clustering problem. The proposed model solved the inefficiency of the performance clustering problem significantly.

In [13], a new binary Dragonfly algorithm based wrapper approach was investigated for the feature selection process. The proposed method outperformed PSO and GAs in terms of classification accuracy and the number of selected attributes.

In [14], a multi-objective genetic algorithm was used for optimizing support vector machines. The proposed model provided good efficiency in terms of a reduced number of features and good accuracy. In [15], the authors proposed an effective hybrid Cuckoo search with Harmony search to detect spam. The proposed model provided good search performance and high accuracy for spam detection.

In [16], binary optimization using a hybrid grey Wolf algorithm was proposed for the feature selection problem. BGWOPSO outperformed the binary GWO (BGWO), the binary PSO, the binary genetic algorithm, and the whale optimization algorithm with simulated annealing in terms of accuracy, selecting the best features and using less computational time.

In [17], the author reviewed variants of the Bat algorithm, its numerous practical optimization problems in engineering, and suggested directions for future research. The description of application of the Bat algorithm to real world optimization problems includes structural optimization, classification and feature selection, electrical power systems, and applications in other areas. In the classification and feature selection application area, the Bat algorithm has been used to tackle the problem of classification of high dimensional microarray datasets to find the optimal structure and the weight in [18].

In addition, the Bat algorithm was used to train a neural network for data classification in [19], which included multiple co-operative sub-populations and a chaotic map to perform selection of an optimal solution. Furthermore, a binary Bat algorithm was used to maximize classifier performance by searching for the optimal features with a wrapper approach for feature selection in [20]. In [21], the capability of the Bat algorithm to solve a high dimensional optimization problem for feature selection in which the most informative features had to be selected was proven. In [22], the Bat algorithm was used to consider image thresholding for a constrained optimization problem.

3. BACKGROUND THEORY

In this section, the background theory and necessary problem domain knowledge are presented.

3.1 Multi-Class Document Classification

Fig. 1 illustrates an example of multi-class document classification which includes more than two classes of document.

When many documents are input to the classification process, the classification process identifies which individual documents belong to which class in a training model. In the case of multi-class document classification, feature selection process is more complicated than bi-class document classification because several features are included in each class and it can lead to a high dimensional problem.

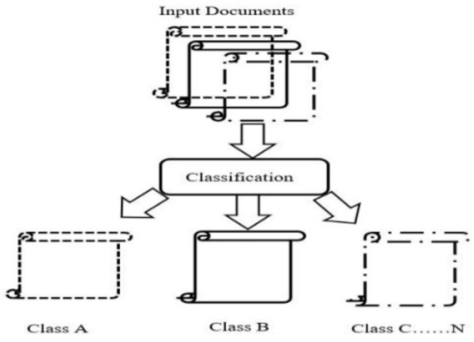


Fig.1: *Multi-Class Document Classification.*

3.2 Feature Extraction and Feature Selection

Feature extraction is the process of implementing a feature vector. There are many kinds of feature extraction schemes [23], such as information gain, mutual information, and so on. However, the TF-IDF scheme [24] is applied in this research to extract the score values of individual text features. The definition of TF-IDF is multiplication of each term in the test document with its normalized inverse document frequency for each document.

The feature selection process is responsible for the removal of irrelevant features and it includes two main parts. A feature evaluation process is performed in the first part and a feature searching process is the second part. There are various types of feature selection processes such as filter, wrapper, hybrid, and embedded schemes. The correlation-based feature subset filter [25] is used for this research. This is because it selects the features that are highly correlated with predicted labels, but not correlated with other labels.

3.3 Cuckoo Optimization Algorithm (CO)

The Cuckoo optimization algorithm (CO) [26] is a novel population based stochastic global search metaheuristic algorithm. It is inspired by the nature of breeding behavior of some cuckoo species that lay their eggs in the nests of host birds. An individual egg represents a solution, and a cuckoo egg represents a new solution. The purpose of CO is to use new and potentially improved solutions to replace worse solutions in the nests.

Three rules of CO can be briefly described. First, each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest. Second, the best nests with high quality eggs (solutions) will carry over to the next generation. Third, the number of available

host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0, 1]$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

As a further estimation, a fraction p_a of the n host nests can be replaced by new nests with new random solutions. In a problem of maximization, the fitness function of a solution is directly proportional to the value of the objective function. In the implementation of CO, a simple set of assumptions is used. Each egg in a nest represents a solution, each cuckoo can lay one egg, the objective function is to look for new potentially better solutions by replacing a not-so-good solution in the nests. However, this basic CO can be extended to be more complicated by modifying some parameters, such as the assumption that each nest has multiple eggs representing a set of solutions.

In the calculation of CO, combination of a local random walk and the global explorative one is performed by switching the parameter p_a in order to have a balanced search. The local random walk is defined by using Equation (1),

$$x_i^{t+1} = x_i^t + \alpha s \otimes H(p_a - \in) \otimes (x_j^t - x_k^t) \quad (1)$$

where x_j^t and x_k^t are two different solutions selected randomly by random permutation, H(u) is a Heaviside function, $\in$ is a random number drawn from a uniform distribution, and s is the step size. Meanwhile, the global random walk is determined by applying Lévy flights in Equation (2),

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda) \quad (2)$$

The Lévy flights are calculated randomly by using Equation (3).

$$L(s, \lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, (s \gg s_0 > 0) \quad (3)$$

Here $\alpha > 0$ is the step size scaling factor, which should be related to the scale of the problem of interest. In most cases, $\alpha = O(L/10)$ is used, where L is the characteristic scale of the problem of interest. In some cases, $\alpha = O(L/100)$ can be more effective and avoid flying too far. Obviously, the α value in these two updating equations can be different, α_1 and α_2. However, $\alpha_1 = \alpha_2 = \alpha$ can be used for simplicity.

The benefit of CO is that Lévy Fights are used for global search rather than standard random walks. It can provide infinite mean and variance that can encourage the exploration of more efficient searches than standard Gaussian processes. In addition, it can provide global convergence by combing the capability of local and global search.

CO has been used in various areas of optimization and computational intelligence with good efficiency. For instance, it has been used in engineering design applications [27][28]. Moreover, it can also be used for training a spiking neural network model, optimizing semantic web service composition processes, optimizing designs for embedded systems, selection of optimal machine parameters in milling operations, and generating independent paths for software testing and data generation.

In addition, modified CO is used for solving nonlinear problems. On the other hand, a discrete CO is often used to solve nurse scheduling problems [29]. Furthermore, a variant of CO in combination with a quantum-based approach can be used to solve the Knapsack problem [30]. CO search and differential evolution algorithms can provide more robust results than PSO and ABC. In complex phase equilibrium applications, CO search can offer a reliable method for solving thermodynamic calculations. It can be used for solving a six-bar double dwell linkage problem [31] and solving the distributed generation allocation problem in distribution networks [32] with a good convergence rate and performance.

Algorithm 1 presents the pseudo-code of the Cuckoo optimization algorithm for multi-class document classification. The objective function F(x) is formulated in Equations (12) and (13).

Algorithm 1. Cuckoo Optimization Algorithm(CO)

Input: Objective function $F(x), x = (x_1, x_2, \ldots, x_n)$.
Output: Best solution x_{best}.

```
1. Define initial population of n host nests
2. while (t < G_max)
3.   | Get a cuckoo randomly, named i, and replace
     | its solution by performing Lévy Fights
4.   | Evaluate the quality of selected feature (F_i)
     | with objective function
5.   | Select a nest among n, named j, randomly
6.   | if (F_i > F_j)
7.   |   | Replace j by the new solution
8.   | end if.
9.   | A fraction or probability (p_a) of the worse
     |   nests are abandoned and new ones are built
10.  | Keep the best solution for nests
11.  | Rank the solutions for nests and find the
     |   current best
12.  | Pass the current best solutions to next cycle
13. end while.
```

In Algorithm 1, t is the time step and G_{max} is the maximum number of generations. In addition, i is the randomly selected cuckoo ($i = 1, 2, \ldots$), and j is the randomly selected nest from n, number of nest ($j = 1, 2, \ldots$).

3.4 Firefly Optimization Algorithm (FO)

The Firefly algorithm (FO) [33] is a nature-inspired meta-heuristic algorithm. That is based es-

pecially on the flashing patterns and the behavior of fireflies. It includes three basic rules. First, since fireflies are unisex, each firefly will be attracted to every other firefly regardless of their sex. Second, the level of attractiveness is proportional to the brightness. Therefore, for any two flashing fireflies, the less bright firefly will move towards the brighter one. However, a firefly will move randomly if there is no brighter firefly nearby. Third, the brightness of a firefly is determined by the landscape of the objective function.

The attractiveness is directly proportional to the light intensity of adjacent fireflies and the variation of attractiveness β for distance r can be defined using Equation (4),

$$\beta = \beta_0 e^{-\gamma r^2} \tag{4}$$

where β_0 is the attractiveness at distance $r = 0$. In addition, r is the distance between two fireflies, and it can be formulated as Equation (5),

$$r(i,k) = |x(\vec{i}) - x(\vec{k})| = \sqrt{\sum_{j=1}^{d}(x_{ij} - x_{kj})} \tag{5}$$

where x_{ij} is the j^{th} component of the i^{th} firefly and x_{kj} is the j^{th} component of the k^{th} firefly. The variable d is the dimension or size of the studied document. Moreover, the movement of firefly i to brighter firefly j is defined by using Equation (6),

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha_t \epsilon_i^t \tag{6}$$

where the second term is due to the attraction and the third term is randomization with respect to α_t being the randomization parameter. ϵ_i^t is a vector of random numbers derived from a Gaussian distribution or uniform distribution at time t.

If $\beta_0 = 0$, it becomes a simple random walk. On the other hand, if $\gamma = 0$, it reduces to a variant of particle swarm optimization. Furthermore, the randomization ϵ_i^t can easily be extended to other distributions such as Lévy flights.

Algorithm 2 illustrates the pseudo-code of the Firefly optimization algorithm for multi-class document classification. The objective function F(x) is formulated in Equations (12) and (13).

The FO algorithm has attracted much attention for many applications such as digital image compression with minimal computation time [34]. Meanwhile, it is used for feature selection with consistent and better performance by means of time and optimality.

Algorithm 2. Firefly Optimization Algorithm(FO)

Input: Objective function $F(x), x = (x_1, x_2, \ldots, x_n)$.
Output: Best solution x_{best}.

```
1. Generate a population of fireflies randomly
2. Define light density I // TF_IDF
3. Define absorption coefficient, γ, with 0.001
4. Define randomization parameter, α = 1
5. Define attractive value, β0 = 0.33
6. Define a maximum number of iterations,
   MaxGeneration
7. Initialize t = 0
8. while (t < MaxGeneration) do
9.     for (i = 1 : n) do
10.        for (k = 1 : n) do
11.            if (Ik > Ii) then
12.                Calculate the distance r using
                   Equation (5)
13.                Calculate the attractiveness β
                   using Equation(4)
14.                Evaluate the solution by updating the
                   light intensity
15.            end
16.        end
17.        Rank fireflies and find the current best
18.        Record the position of the current best
           firefly and discretize the current best
           real position
19.    end
20.    Return the intensity with the discrete
       position.
21. end.
```

3.5 Bat Optimization Algorithm (BO)

The Bat optimization algorithm (BO) [35] is a bio-inspired algorithm which was developed recently. The idea of BO was based on the echolocation features of microbats. It uses a frequency-tuning technique to increase the diversity of the solutions in the population and it also employs automatic zooming in order to balance exploration and exploitation for a search process by mimicking the variations of pulse emission rates and loudness of bats when looking for prey.

Three basic rules are used to develop the BO algorithm. First, echolocation is used by all bats to sense distance between food/prey and background barriers in some magical way. Second, bats fly randomly using velocity v_i at position x_i with a frequency f_{min}, varying wavelength λ, and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target. Third, the assumption of loudness variation from a large (positive) A_0 to a minimum constant value A_{min} is used, although the loudness can vary in many ways.

Algorithm 3 shows the pseudo-code for the Bat op-

timization algorithm for multi-class document classification in which $\mathfrak{o} \sim U$ (0,1). The objective function F(x) is formulated in Equations (12) and (13). The velocity v_i^t and a location x_i^t, at iteration t, in a d dimensional search or solution space, are always important factors for the consideration of each bat. There exists a current best solution x_* for all of the bats. The following three Equations (7), (8), and (9),

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{7}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*)f_i \tag{8}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{9}$$

where $\beta \in [0,1]$ is a generated random vector using a uniform distributed function, are used to find x_*.

Algorithm 3. Bat Optimization Algorithm(BO)

Input: Objective function $F(x), x = (x_1, x_2, \ldots, x_n)$.
Output: Best solution x_{best}.
1. Define velocity v_i^t with frequency f_i at x_i^t,, $\forall_i = 1, 2, \ldots, m$
2. Define pulse rate r_i and loudness $A_i, i = 1, 2, \ldots, m$
3. **while** $t < T_{max}$ **do**
4. **for** each bat B_i **do**
5. Generate new solutions through Equations (6), (7), (8)
6. **if** $\mathfrak{o} > r_i$ **then**
7. Select a solution among the best ones
8. Generate a local solution around the best solution
9. **end**
10. **if** $\mathfrak{o} < A_i$ and $F(xi) < F(x_{best})$ **then**
11. Accept the new solutions
12. Increase r_i and reduce A_i
13. **end**
14. **end**
15. Rank the bats and find the current best solution x_{best}
16. **end**

In the simple model of BO, ray tracing is not used, although it could be an interesting feature for further extension. Though ray tracing can be computationally expensive, it can be a very useful feature for computational geometry and other applications. Furthermore, frequency is always intrinsically linked to a wavelength. Therefore, frequency f or wavelength λ should be changed depending on the ease of implementation and other factors for different applications.

The implementation of BO can be used for various domain sizes of the problem of interest. Since frequency is assigned randomly for each bat, a frequency tuning algorithm should be used to support a balanced combination of exploration and exploitation. In addition, the loudness and pulse emission rates can be used for controlling and auto zooming into the region with promising solutions.

Since the loudness decreases when a bat has found prey while the rate of pulse emission increases, the loudness A_i and the rate of pulse emission r_i must vary between A_{min} and A_{max} during the iterations. In the case of $A_{min} = 0$, a bat has just found the prey and temporarily stops emitting any sound. The pulse rate can be defined using Equation (10),

$$v_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0(1 - e^{\gamma t}) \tag{10}$$

where α and γ are constants. For any $0 < \alpha < 1$ and $\gamma > 0$. The loudness can be defined using Equation (11).

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \quad \text{as } t \to \infty \tag{11}$$

where $\alpha = \gamma$ can be used for simple case. In addition, BO algorithms can be applied in various areas of optimization, scheduling, feature selection, classification, data mining, image processing, etc.

3.6 J48 and Support Vector Machine (SVM)

J48 [36] is a simple and popular classification learning model which includes three main components for making decisions: internal nodes, branches and leaf nodes. In addition, it can be used commonly in classification and clustering problems. The unknown attribute value of each tuple for the corresponding class label is tested against the J48 decision tree by tracing from the top root to a leaf node and then the path is converted to rules of classification.

SVM [37] is a supervised learning algorithm for selecting a modest amount of significant limit samples from all labels of a class. Then, the maximum margin hyperplane is constructed. The most supreme division among the classes is given by the hyperplane with the greatest margin. Non-linear functions can be used if the system exceeds the boundary of linear functions. The optimal hyperplane is achieved by maximizing the distance from the separating boundary to the closet point of the separating hyperplane.

3.7 Performance Evaluation Measurements

In the evaluation process of the proposed system, accuracy (A) and root mean squared error (RMSE) were used as performance measurements. Accuracy is the measurement of correctness for true positive and true negative. RMSE is one popular type of measurement for calculating the "deviation of estimated or predicted numeric values from their actual values" by averaging a set of errors. A smaller mean squared error value shows less error exists between the estimated value and the actual value in learning models. In addition, a selected number of global features (NGF) and building time for classification (BCT) are

used to evaluate the complexity cost of the proposed model. The calculation for accuracy and RMSE are shown in Equations (12) and (13) respectively.

$$\text{Accuracy} = \frac{(\text{TP}+\text{TN})}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} * 100\% \qquad (12)$$

where TP is the number of true positive classification, TN is the number of true negative classification, FP is the number of false positive classification, and FN is the number of false negative classification.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=2}^{N}(\text{Predicted}_i - \text{Actual}_i)^2}{N}} \qquad (13)$$

In Equation (13), N is the total number of samples.

4. SYSTEM IMPLEMENTATION

This section presents the detailed implementation of our proposed system and it has four sub-sections: experimental setup, system design, objective function, and parameter setting.

4.1 Experimental Setup

In this experiment, we used 2,250 news documents with the labels for sports, politics, business, technology, and entertainment. Moreover, the Weka library file was used for the feature selection process and learning model implementation. In addition, an HP Desktop Parvilion PC with an Intel Core I3-9100, 4 GB DDR4 RAM, 1TB of non-volatile storage, running the Microsoft Windows 10 operating system was used as the platform on which to implement the proposed model. The total number of extracted features was 2,591, and the correlation-based feature subset filter approach was used to select the relevant features from the extracted feature vector.

4.2 System Design

Document classification is the process of labelling documents into specified categories. Fig. 2 shows the optimization of multi-class document classification using a computational search policy. In the training stage, removal of irrelevant words, and extraction of n-gram based TF-IDF are performed as a preprocessing stage. Document classification can be regarded as multi-dimensional feature problem because the extracted number of features is greater than the size of dataset. If a traditional algorithmic search is applied for multi-dimensional feature selection, it can provide only local optimum features and that leads to high computation cost for the learning model.

Local search for the individual class feature hypothesis is not enough for complex features because some of the features are common to all classes. Therefore, a global search approach is observed for multi-dimensional feature selection problems in various applications of computer science and engineering. A highly intelligent classification learning model is used to evaluate the performance of selected global optimal feature subsets. In the testing stage, the trained, highly intelligent classification model is applied to the test documents to classify the labels of those documents.

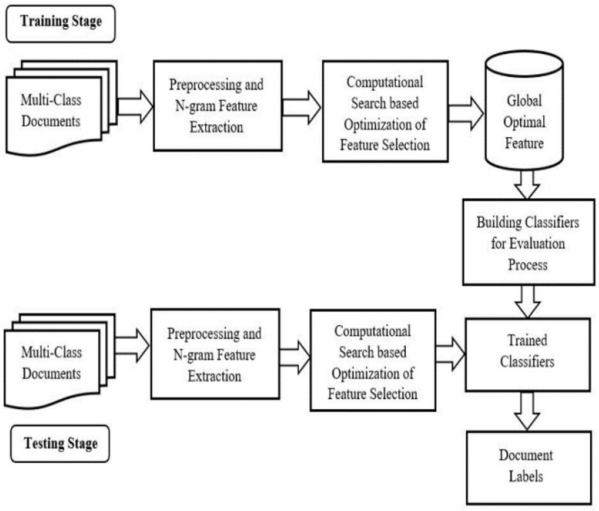


Fig.2: *Optimization of Multi-Class Document Classification using Computational Search Policy.*

4.3 Objective Function of Proposed System

The objective function is the main component of all computational search algorithms. It describes the goal of the optimization problem with the specification of minimal or maximal values of each objective function. Basically, there are two types of objective function: single objective functions and multi-objective functions.

In the implementation of our proposed system, three computational search algorithms are used to optimize the classification performance of multi-label documents. The objective function for each computational search algorithm maximizes the accuracy (A) and minimizes the root mean squared error (RMSE) of classifier output. In addition, various population sizes (PS) in the range of 20 to 200 are used to search the global optimal feature subset which can enhance the performance results of two of the classifiers.

4.4 Parameter Setting

In this section, the parameter settings for system implementation are described. In Table 1, the main parameter settings for BFS and RS are shown. The prominent parameter settings for the BO, CO, and FO algorithms are summarized Table 2.

Table 1: *Parameter Settings: Traditional Search.*

Parameters	BFS	RS
Direction	Forward	Forward
Start	Empty Set	Empty Set
Termination	Consecutive non-improving nodes (CNI: 5)	End of the feature vector and rank ascending order

Table 2: *Prominent Parameter Settings: Computational Search Algorithm.*

Search Approach	Parameter	Value
BO	Frequency	0.5
	Loudness	0.5
CO	Constant rate (pa)	0.25
	Constant rate (sigma)	0.69657
FO	Coefficient of absorption	0.001
	Coefficient (betaMin)	0.33

5. RESULTS AND DISCUSSION

In this experiment, we measured the performance of modern computational search-based optimization of multi-class document classification by comparing it to the traditional determinative search. Three modern nature-inspired algorithms, Cuckoo Optimization (CO), Firefly Optimization (FO), and Bat Optimization (BO), are used with a filter feature selection approach for searching for the global optimal subset features for our document classification model.

Moreover, the evaluation process for measuring the performance of the proposed models was performed by two different classifiers. The measurement of accuracy and root relative square error for performance, and the measurements of the number of selected features and time taken for computation complexity, are used to evaluate the capability of the proposed computational search algorithms. All detailed testing results are discussed in the following four subsections.

5.1 Optimization of Classification Accuracy Results with Computational Search

In this section, the accuracy results for multi-class document classification using three different computational search algorithms, according to the rate of change of the population size parameter, for performing the evaluation process with two different classifiers are discussed.

Fig. 3 depicts accuracy results for the J48 learning model using Cuckoo (CO), Firefly (FO), and Bat Optimization search (BO) for classifying five labels of multi-class documents. The best accuracy result was achieved using CO with the value of (**92.03%**) at population size (PS = **80**). Moreover, the second-best accuracy result (**90.55%**) was obtained continuously at (PS = **90**). The approximate value of second-best accuracy, (**90.23%**) happened at (PS = **200**). Meanwhile, the accuracy value for FO provided the highest accuracy value of ((**91.81%**) at the initial population size (PS = **20**). The second highest one, (**90.77%**), was provided at (PS = **100**). For BO, the peak accuracy value (**89.42%**) existed at (PS = **90**), and the results (**89.15%**) at (PS = **200**) are not different from the peak value.

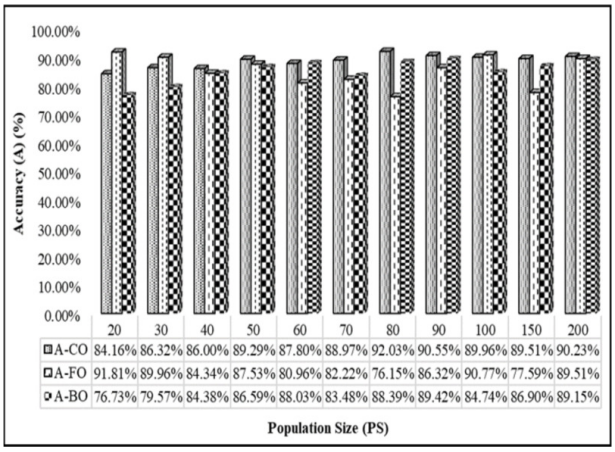
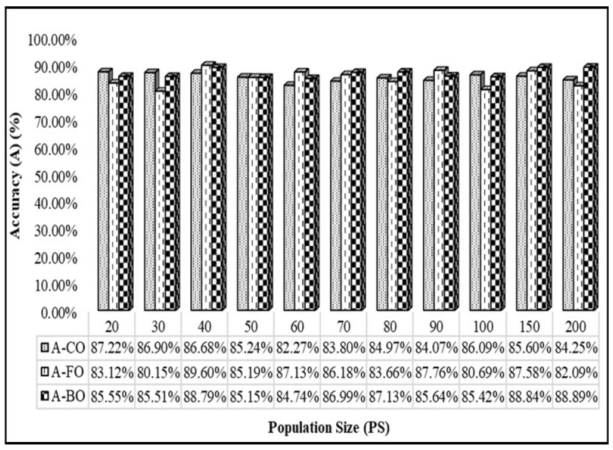


Fig.3: *Accuracy Results using J48 with Computational Search Policy.*

Since the purpose of the objective function for the proposed system was to maximize the accuracy of classification, the proposed model was tested using different population sizes in this experiment in order to find the optimal feature subset that would yield the optimal accuracy of classification results. The trend for the three optimization search algorithms was to fluctuate when the size of the population changed because computational search uses a random search policy according to the probability value of individual candidate solutions for both exploitation and exploration search. However, the ability of intelligent search according to the fitness value of neighbor candidate solutions can be used to find the global optimal results. Nevertheless, the average accuracy results for the three nature-inspired based computational search algorithms was approximately 90% for the J48 learning model.

Fig. 4 shows the evaluation process using an SVM classifier to measure the performance for our proposed system in the range of population sizes from 20 to 200. In contrast with J48 classification, the peak accuracy value (**89.60%**) was provided by the FO search policy and the second-best accuracy (**88.89%**) was achieved in the case of BO for multi-class document classification. The CO-based optimization model was third best in the measurement of accuracy. Its highest accuracy value of (**87.22%**) was obtained at (PS = **20**). Meanwhile, the lowest accuracy value (80.15%) of FO was still not better than the lowest one of BO (84.74%) or the lowest one of CO (82.27%).

In addition, the fluctuation rate for the three optimization algorithms did not change dramatically, unlike with the J48 classifier (Fig. 3). Moreover, there were no accuracy results that were less than 80% in all three algorithms using the SVM evaluation process when compared to the accuracy results of J48. In summary, the accuracy value trend is different for each computational search algorithm when synchronizing with different classifiers. Therefore, more classifiers should be set up with computational search policies, and their effects measured in the future.



	20	30	40	50	60	70	80	90	100	150	200
A-CO	87.22%	86.90%	86.68%	85.24%	82.27%	83.80%	84.97%	84.07%	86.09%	85.60%	84.25%
A-FO	83.12%	80.15%	89.60%	85.19%	87.13%	86.18%	83.66%	87.76%	80.69%	87.58%	82.09%
A-BO	85.55%	85.51%	88.79%	85.15%	84.74%	86.99%	87.13%	85.64%	85.42%	88.84%	88.89%

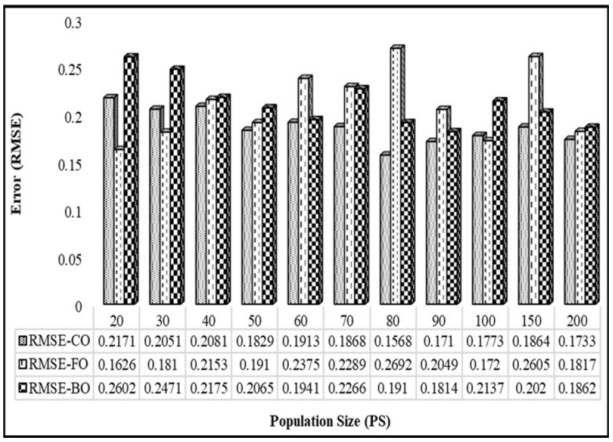
Fig.4: *Accuracy Results using SVM with Computational Search Policy.*

5.2 Error Results for Multi-Class Document Classification using Computational Search

Another objective function for our proposed model minimizes the error for multi-class document classification. Therefore, Fig. 5 and 6 demonstrate the evaluation process of our proposed system using J48 and SVM in terms of root mean square error.

In Fig.5, the highest minimum means squared error (RMSE) value (**0.1568**) happened at (PS = **80**) in CO and the highest maximum RMSE (**0.2171**) occurred at (PS = **20**). Meanwhile, FO based optimization of multi-class document classification provided the lowest RMSE (**0.1626**) at (PS = **20**) and also the highest one (**0.2692**) at (PS = **90**). In BO-based optimization, the smallest error rate (**0.1814**) was achieved at (PS = **90**), and the largest one (**0.2602**) occurred at (PS = **20**). In addition, all RMSE results of our proposed model using computational search policy were not higher than the average of RMSE (**0.5**), which means our proposed model satisfies the objective function of accuracy and error rate.

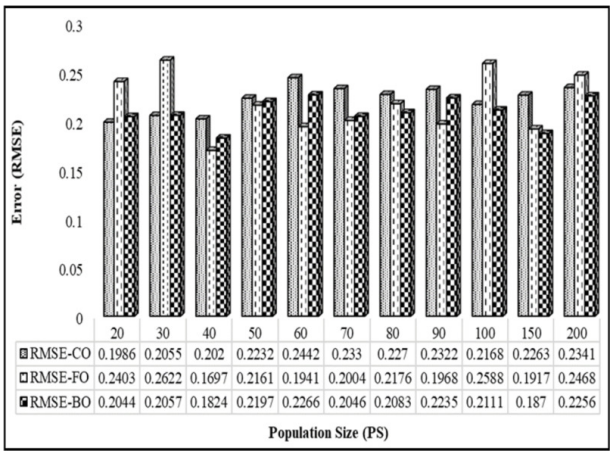
CO-based optimization for multi-class document classification provided the least RMSE value, followed by FO-based optimization of error occurrence. However, the BO-based proposed model presented the highest RMSE value, similar to the performance accuracy results in section 5.1. In addition, the rate of change of RMSE results for all three computational search algorithms is linear. That means the proposed model provides a good approach for this optimization problem.



	20	30	40	50	60	70	80	90	100	150	200
RMSE-CO	0.2171	0.2051	0.2081	0.1829	0.1913	0.1868	0.1568	0.171	0.1773	0.1864	0.1733
RMSE-FO	0.1626	0.181	0.2153	0.191	0.2375	0.2289	0.2692	0.2049	0.172	0.2605	0.1817
RMSE-BO	0.2602	0.2471	0.2175	0.2065	0.1941	0.2266	0.191	0.1814	0.2137	0.202	0.1862

Fig.5: *Error Results using J48 with Computational Search Policy.*

Fig. 6 shows the value of RMSE for the CO, FO, BO, and SVM classifiers used to evaluate the performance of our proposed system. One can see in the trend of results in accuracy measurement (first objective function) (Fig. 2) that the minimum RMSE value (**0.1697**) was obtained at (PS = **40**) in the optimization of classification using the FO algorithm, and the second most minimum value (**0.1824**) at (PS = **40**) occurred in the BO algorithm. The highest RMSE value (**0.1986**) at (PS = **20**) happened with the CO-based search policy. However, the highest RMSE value (**0.2622**) occurred with the FO search policy. Although CO and BO-based search policy cannot provide a lower RMSE value than FO, their fluctuation graphs are linear for population sizes of 20 to 200.



	20	30	40	50	60	70	80	90	100	150	200
RMSE-CO	0.1986	0.2055	0.202	0.2232	0.2442	0.233	0.227	0.2322	0.2168	0.2263	0.2341
RMSE-FO	0.2403	0.2622	0.1697	0.2161	0.1941	0.2004	0.2176	0.1968	0.2588	0.1917	0.2468
RMSE-BO	0.2044	0.2057	0.1824	0.2197	0.2266	0.2046	0.2083	0.2235	0.2111	0.187	0.2256

Fig.6: *Error Results using SVM with Computational Search Policy.*

5.3 Complexity Cost Results for Multi- Class Document Classification with Computational Search

In this section, the cost of complexity for individual, nature-inspired based computational search strategies are discussed using selected number of global subset features (NGF) and building time for the classification models (BCT).

Table 3 shows the complexity cost for CO-based optimization classification. The lowest NGF is (***481***) and the second lowest one (***498***) was selected in the first and next smallest population sizes (PS = **20**) and (PS = **30**) respectively. The trend for NGF results is to increase when the PS value is increased, and the maximum NGF value (**858**) was obtained at (PS = **200**). As a consequence of having the lowest NGF, the shortest BCT (**0.69**) was obtained at the lowest PS value in the evaluation process of J48. Similarly, the highest cost for computation time (**2.06**) was obtained at maximum NGF in J48 classification.

In contrast, the best computation cost for building a classification model (**6.35**) in SVM was achieved at the second highest NGF value (PS = **150**), and the longest BCT (**9.94**) happened at the second lowest NGF value (PS = **30**). Among these two classifiers evaluation processes, the computation cost results for optimization of multi-class document classification using the J48 classifier was better than the one using the SVM classifier for all PS values. Like the measurement of performance, the results for both J48 and SVM always have inclination and declination according to the rate of change of population size.

Table 3: *Complexity Cost Results using CO.*

PS	NGF	BCT-J48	BCT-SVM
20	***481***	**0.69**	6.76
30	498	1.08	9.94
40	500	0.72	7.64
50	693	1.15	9.09
60	777	1.42	9.86
70	778	1.53	7.82
80	785	0.94	6.74
90	765	1.39	7.70
100	843	1.05	7.43
150	770	1.41	**6.35**
200	858	2.06	8.89

Table 4 shows the complexity cost results for the FO-based optimization model for document classification in terms of NGF and BCT values in both two-evaluation processes for J48 and SVM classifiers. The smallest selected number of feature subsets (***85***) was achieved at (PS = **40**), while the largest one (**1021**) occurred at (PS =**100**). Similarly, the shortest BCT values for both J48 (**0.31**) and SVM (**6.34**) classifiers were obtained at the smallest NGF value at the lowest PS value. However, the occurrence of the highest BCT value in both J48 and SVM classifiers was different. It was (**1.59**) at (PS = **20**) in J48, and (**12.58**) at (PS = **60**) in SVM.

Table 4: *Complexity Cost Results using FO.*

PS	NGF	BCT-J48	BCT-SVM
20	942	1.59	10.42
30	868	1.58	8.42
40	***85***	**0.31**	**6.34**
50	636	0.87	9.23
60	263	0.47	12.58
70	295	0.80	6.35
80	227	0.50	9.47
90	547	0.83	7.27
100	1021	1.56	10.42
150	256	0.44	6.51
200	936	1.49	9.12

In addition, the computation complexity results for the BO-based mode are shown in Table 5. The same result trend for NGF and BCT-J48 occurred like in the case of CO. For example, the smallest NGF value (***279***) was obtained at the lowest PS (PS = **20**), and the best computation cost (**0.60**) was achieved at the smallest NGF in the J48 evaluation process. In contrast, the shortest BCT value (**6.49**) in SVM classifier was provided at (PS = **80**). As in the results of CO and FO, better BCT values are given by the J48 classifier than the SVM one for all values for various population sizes. Moreover, the computation cost is directly proportional to the selected number of feature subsets in the J48 classification model, but the reverse condition exists in SVM.

Table 5: *Complexity Cost Results using BO.*

PS	NGF	BCT-J48	BCT-SVM
20	***279***	**0.60**	7.96
30	311	0.64	10.82
40	396	0.78	10.26
50	595	1.39	8.96
60	732	1.50	7.79
70	473	1.04	12.01
80	645	1.34	**6.49**
90	776	1.41	7.58
100	469	1.17	10.10
150	478	1.45	10.43
200	731	1.67	10.57

5.4 Performance Comparison: Computational Search Policy and Traditional Determinative Search

In this section, the comparison of performance between computational search and traditional determinative search for optimization of multi-label document classification is discussed.

According to the accuracy and error results in Table 6, the accuracy value of CO, (**92.03%**), is the best one among three computational search algorithms. The accuracy of RS (**92.44%**) is also the best one among the two traditional search algorithms for the J48 classifier. Therefore, the proposed model using CO search algorithm provides the optimization of classification accuracy with a reduced NGF value (***785***) when compared to the one in traditional search (***2591***). Better results can be obtained according to the parameter tuning of PS in (Fig. 1) if we increase the computation cost.

In contrast, traditional search policy can only provide a fixed accuracy value that can lead lower accuracy when more unseen datasets are considered for testing. Similarly, the accuracy values of FO and BO also reached (***91.81%***) and (***89.42%***), and their results are approximately the same as the results in RS (**92.44%**) and BFS (**92.08%**) with adaptive intelligence search and reduced NGF values. However, the best accuracy result was provided in the evaluation process of SVM with FO search (**89.60%**), while the best accuracy value for traditional search was also changed from RS to BFS (**95.18%**). In addition, the accuracy result for RS (***60.44%***) dropped dramatically with the SVM classifier, while the computational search policy-based optimization of classification accuracy did not decline sharply for all three algorithms' results.

In the result comparison of RMSE between computational search and traditional determinative search, the same trend of results was achieved as in the case of accuracy comparison, except for the change of best RMSE value in BFS for the J48 evaluation process. Although the same result of RMSE is obtained with the CO algorithm (**0.1568**) when compared to BFS (**0.1569**), the approximate RMSE value of BFS at (NGF = **102**) and RS (***0.1603***) can be obtained in FO search (***0.1626***) at (NGF = **2591**) with the lowest computation cost of NGF (***85***) in FO. In the case of the SVM classification process, the error value for RS (***0.3786***) inclined obviously, while a linear change of RMSE values for all three optimization algorithms was obtained. However, the lowest RMSE value changed from CO in J48 classification to FO (**0.1697**) in SVM, and the RMSE value of CO increased when compared to RMSE in the J48 evaluation process.

Table 6: *Accuracy and Error Results Comparison: Computational Search and Traditional Search.*

Experiments	A-J48	A-SVM	RMSE-J48	RMSE-SVM
CO	**92.03%**	87.22%	**0.1568**	0.1986
FO	***91.81%***	**89.60%**	***0.1626***	**0.1697**
BO	***89.42%***	88.89%	0.1814	0.1824
BFS	**92.08%**	**95.18%**	**0.1569**	**0.1197**
RS	**92.44%**	***60.44%***	***0.1603***	***0.3786***

Table 7 shows the comparison of complexity cost results between computational search and traditional search. FO based optimization of document classification model (**85**) provided the smallest reduced number of selected feature subset when compare to the results in traditional search. In addition, CO (481) and BO (279) computational search algorithms also provided reduced NGF when compared to RS (***2591***). In the point of view of computation time for the J48 classifier, the best BCT value for FO (**0.31**) is better than the best BCT value for BFS (**0.34**).

For the other computational search algorithms of CO (0.69) and BO (0.60), the results for BTC were better than the traditional search algorithm of RS (***2.59***). But BCT values for two traditional search algorithms, BFS (**10.89**) and RS (***18.12***), and three conventional search algorithms, CO (6.35), BO (**6.34**), and FO (6.49), increased sharply with SVM. However, better BCT results for three computational searches was achieved when compared to the two traditional searches, and the worst BCT value of BO (6.49) is still better than the best one of BFS (***18.12***).

Table 7: *Complexity Cost Results Comparison: Computational Search and Traditional Search.*

Experiments	NGF	BCT-J48	BCT-SVM
CO	481	0.69	6.35
FO	**85**	**0.31**	**6.34**
BO	279	0.60	6.49
BFS	**102**	**0.34**	**10.89**
RS	***2591***	***2.59***	***18.12***

6. CONCLUSIONS AND FUTURE WORK

Our proposed system provides good optimization results of multi-class document classification with a computational intelligence-based search policy by reducing the number of selected features dramatically. The best accuracy results, with RMSE and BCT, are achieved in the evaluation process of J48. It can be applied to text document classification for BBC news. However, if the behavior of data is changed in volume, complexity, dimensionality, and so on, adaptation of the scheme for document classification processing is always needed. Therefore, the following issues could be considered for future research. First, the proposed model should be implemented on Hadoop distribution platform in order to process higher population sizes for computational algorithm search by decentralizing individual tasks for each agent with a real time big data classification model. Second, the proposed model should be upgraded by combining knowledge of NLP to build a more intelligent learning model which makes decisions more like a human being so it can provide higher accuracy for the learning model with a lower error rate.

ACKNOWLEDGEMENTS

This work was supported by Thailand's Education Hub for ASEAN Countries (Grant No. TEH-AC 058/2016).

References

[1] M. Abdollahi, X. Gao, Y. Mei, S. Ghosh, and J. Li, "An Ontology-based Two-Stage Approach to Medical Text Classification with Feature Selection by Particle Swarm Optimisation," *IEEE Congr. Evol. Comput. CEC*, pp. 119–126, 2019.

[2] A. Kumar, A. Jaiswal, S. Garg, S. Verma, and S. Kumar, "Sentiment Analysis using Cuckoo Search for Optimized Feature Selection on Kaggle Tweets," *Int. J. Inf. Retr. Res.*, vol. 9, no. 1, pp. 1–15, 2018.

[3] Y. Jiang, X. Liu, G. Yan, and J. Xiao, "Modified Binary Cuckoo Search for Feature Selection: A Hybrid Filter-Wrapper Approach," *Proc. - 13th Int. Conf. Comput. Intell. Secur. CIS 2017*, no. 2, pp. 488–491, 2018.

[4] M. M. Mafarja and S. Mirjalili, "Hybrid Binary Ant Lion Optimizer with Rough Set and Approximate Entropy Reducts for Feature Selection," *Soft Comput.*, vol. 23, no. 15, pp. 6249–6265, 2019.

[5] M. Mafarja and S. Mirjalili, "Whale Optimization Approaches for Wrapper Feature Selection," *Appl. Soft Comput. J.*, vol. 62, pp. 441–453, 2018.

[6] L. Zhang, K. Mistry, C. P. Lim, and S. C. Neoh, "Feature Selection using Firefly Optimization for Classification and Regression Models," *Decis. Support Syst.*, vol. 106, pp. 64–85, 2018.

[7] M. Mafarja, I. Jaber, S. Ahmed, and T. Thaher, "Whale Optimisation Algorithm for High-Dimensional Small-Instance Feature Selection," *Int. J. Parallel, Emergent Distrib. Syst.*, pp. 1–17, 2019.

[8] A. Kumar and A. Jaiswal, "Swarm Intelligence based Optimal Feature Selection for Enhanced Predictive Sentiment Accuracy on Twitter," *Multimed. Tools Appl.*, vol. 78, no. 20, pp. 29529–29553, 2019.

[9] H. Wang, L. Tan, and B. Niu, "Feature Selection for Classification of Microarray Gene Expression Cancers using Bacterial Colony Optimization with Multi-Dimensional Population," *Swarm Evol. Comput.*, vol. 48, pp. 172–181, 2019.

[10] M. Alweshah and S. Abdullah, "Hybridizing Firefly Algorithms with a Probabilistic Neural Network for solving Classification Problems," *Appl. Soft Comput. J.*, vol. 35, pp. 513–524, 2015.

[11] H. Wang et al., "A Hybrid Multi-Objective Firefly Algorithm for Big Data Optimization," *Appl. Soft Comput. J.*, vol. 69, pp. 806–815, 2018.

[12] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "A New Feature Selection Method to improve the Document Clustering using Particle Swarm Optimization Algorithm," *J. Comput. Sci.*, vol. 25, pp. 456–466, 2018.

[13] M. M. Mafarja, D. Eleyan, I. Jaber, A. Hammouri, and S. Mirjalili, "Binary Dragonfly Algorithm for Feature Selection," *International Conference on New Trends in Computing Sciences, ICTCS*, pp. 12–17, 2017.

[14] A. Bouraoui, S. Jamoussi, and Y. BenAyed, "A Multi-Objective Genetic Algorithm for Simultaneous Model and Feature Selection for Support Vector Machines," *Artif. Intell. Rev.*, vol. 50, no. 2, pp. 261–281, 2018.

[15] S. P. Rajamohana, K. Umamaheswari, and S. V. Keerthana, "An Effective Hybrid Cuckoo Search with Harmony Search for Review Spam Detection," *Proc. 3rd IEEE Int. Conf. Adv. Electr. Electron. Information, Commun. Bio-Informatics, AEEICB*, pp. 524–527, 2017.

[16] Q. Al-Tashi, S. J. Abdul Kadir, H. M. Rais, S. Mirjalili, and H. Alhussian, "Binary Optimization using Hybrid Grey Wolf Optimization for Feature Selection," *IEEE Access*, vol. 7, pp. 39496–39508, 2019.

[17] T. Jayabarathi, T. Raghunathan, and A. H. Gandomi, "The Bat Algorithm, Variants and Some Practical Engineering Applications: A Review," *Stud. Comput. Intell.*, vol. 744, pp. 313–330, 2018.

[18] S S. Mishra, K. Shaw, and D. Mishra, "A New Meta-Heuristic Bat Inspired Classification Approach for Microarray Data," *Procedia Technol.*, vol. 4, pp. 802–806, 2012.

[19] N. S. Jaddi, S. Abdullah, and A. R. Hamdan, "Multi-Population Cooperative Bat Algorithm-based Optimization of Artificial Neural Network Model," *Inf. Sci. (Ny).*, vol. 294, pp. 628–644, 2015.

[20] D. Rodrigues et al., "A Wrapper Approach for Feature Selection based on Bat Algorithm and Optimum-Path Forest," *Expert Syst. Appl.*, vol. 41, no. 5, pp. 2250–2258, 2014.

[21] R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa, and X. S. Yang, "BBA: A Binary Bat Algorithm for Feature Selection," *Brazilian Symp. Comput. Graph. Image Process.*, pp. 291–297, 2012.

[22] Z. W. Ye, M. W. Wang, W. Liu, and S. Bin Chen, "Fuzzy Entropy based Optimal Thresholding using Bat Algorithm," *Appl. Soft Comput. J.*, vol. 31, pp. 381–395, 2015.

[23] J. H. Correia, R. Wille, G. Stumme, and U. Wille, "Conceptual Knowledge Discovery a Human Centered Approach," *Appl. Artif. Intell.*, vol. 17, no. 3, pp. 281–302, 2003.

[24] J. T. Medler, "Term-Weighting Approaches in Automatic Text Retrieval," *Insect Syst. Evol.*, vol. 17, no. 3, pp. 323–337, 1986.

[25] M. A. Hall, "Correlation-based Feature Selection for Machine Learning," 1999.

[26] X. S. Yang and S. Deb, "Cuckoo Search: Recent Advances and Applications," *Neural Comput. Appl.*, vol. 24, no. 1, pp. 169–174, 2014.

[27] X. S. Yang and S. Deb, "Engineering Optimisation by Cuckoo Search," *Int. J. Math. Model. Numer. Optim.*, vol. 1, no. 4, pp. 330–343, 2010.

[28] A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Cuckoo Search Algorithm: A Metaheuristic Approach to solve Structural Optimization Problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, 2013.

[29] L. H. Tein and R. Ramli, "Recent Advancements of Nurse Scheduling Models and A Potential Path," In *Proc. 6th IMT-GT Conference on Mathematics, Statistics and its Applications (ICMSA)*, pp. 395–409, 2015.

[30] A. Layeb, "A Novel Quantum inspired Cuckoo Search for Knapsack Problems," *Int. J. Bio-Inspired Comput.*, vol. 3, no. 5, pp. 297–305, 2011.

[31] R. R. Bulatović, S. R. Dordević, and V. S. Dordević, "Cuckoo Search Algorithm: A Metaheuristic Approach to solving the Problem of Optimum Synthesis of a Six-Bar Double Dwell Linkage," *Mech. Mach. Theory*, vol. 61, pp. 1–13, 2013.

[32] T. T. Nguyen, A. V. Truong, and T. A. Phung, "A Novel Method based on Adaptive Cuckoo Search for Optimal Network Reconfiguration and Distributed Generation Allocation in Distribution Network," *Int. J. Electr. Power Energy Syst.*, vol. 78, pp. 801–815, 2016.

[33] X. S. Yang and X. He, "Firefly Algorithm: Recent Advances and Applications," *Int. J. Swarm Intell.*, vol. 1, no. 1, p. 36, 2013.

[34] M. H. Horng, "Vector Quantization using the Firefly Algorithm for Image Compression," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 1078–1091, 2012.

[35] X. S. Yang, "Bat Algorithm: Literature Review and Applications," *Int. J. Bio-Inspired Comput.*, vol. 5, no. 3, pp. 141–149, 2013.

[36] Quinlan, J. R., "C 4.5: Programs for Machine Learning," Elsevier, 2014.

[37] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO Algorithm for SVM Classifier Design," *Neural Comput.*, vol. 13, no. 3, pp. 637–649, 2001.

Khin Sandar Kyaw graduated with a Master of Engineering (Computer Engineering and Information Technology) from Yangon Technological University (YTU) in 2014. She is a Ph.D. research scholar of the Thailand's Education Hub for ASEAN Countries Scholarship (Grant No. TEH-AC 058/2016) attached to the Artificial Intelligence and Machine Learning research group, Department of Computer Engineering, Prince of Songkla University, Thailand. Her research interests are Data Mining, Nature-inspired and Swarm-based Optimization Algorithms, Natural Language Processing, Data Science, and Web Development.

Somchai Limsiroratana was born in Thailand. He received the B.Eng. degree in electrical engineering from Prince of Songkla University in 1991, the M.Arg and Dr.Arg degrees from division of environmental science and technology, Kyoto University in 2000 and 2005 respectively for the detection of fruits on natural background research. He has been working at Department of Computer Engineering, Prince of Songkla University since 1991. His research interests are agricultural image processing, medical image processing, optimization and AI.