GBRUN: A Gradient Search-based Binary Runge Kutta Optimizer for Feature Selection

Zhi-Chao Dou[1], Shu-Chuan Chu[1,2], Zhongjie Zhuang[1], Ali Riza Yildiz[3], Jeng-Shyang Pan[1]*

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, China
[2] College of Science and Engineering, Flinders University, Australia
[3] Department of Mechanical Engineering, Bursa Technical University, Turkey
douzhichao2021@163.com, scchu0803@gmail.com, zhongjiezhuang@126.com, aliriza@uludag.edu.tr, jspan@cc.kuas.edu.tw

Abstract

Feature selection (FS) is a pre-processing technique for data dimensionality reduction in machine learning and data mining algorithms. FS technique reduces the number of features and improves the model generalization ability. This study presents a Gradient Search-based Binary Runge Kutta Optimizer (GBRUN) for solving the FS problem of high-dimensional. First, the proposed method converts the continuous Runge Kutta optimizer (RUN) into a binary version through S-, V-, and U-shaped transfer functions. Second, a gradient search method is introduced to improve the exploration capability of the algorithm. Five standard datasets provided by Arizona State University's Data Mining and Machine Learning Lab were selected to verify the performance of the GBRUN algorithm. The experimental results show that GBRUN has better performance than other advanced algorithms regarding classification accuracy and the number of selected features. Moreover, the GBRUN algorithm is also combined with EfficientNet in this manuscript, using the GBRUN algorithm to select the features extracted by EfficientNet. The results show that the V-shaped (GBRUN-V) and U-shaped (GBRUN-U) algorithms have better performance than other algorithms.

Keywords: GBRUN, Feature selection, Runge Kutta method, COVID-19 dataset, EfficientNet

1 Introduction

With the rapid development of science and technology, the complexity and diversity of data in various fields are increasing. However, high-dimensional data often suffer from a range of problems such as missing data and redundant features, making data processing more difficult [1]. FS is a common data pre-processing method that can effectively reduce the number of feature dimensions to reduce the difficulty of data processing, reduce processing equipment's performance requirements, and improve data processing efficiency. FS has been commonly used in many fields such as deep learning and data mining [2-3].

Many researchers use Meta-heuristic algorithms based on the ease of use and flexibility of Meta-heuristic algorithms to deal with FS problems [4]. Meta-heuristic algorithms can be divided into two broad categories: Evolutionary Algorithms (EA) and Swarm Intelligence (SI) optimizers [5-8]. The differential evolution algorithm (DE) is an efficient global optimization algorithm mainly used for solving real number optimization problems [9]. Genetic Algorithm (GA) is an evolutionary algorithm designed by simulating the Darwinian theory of biological evolution [10]. In addition to these two representative algorithms, there are also Simulated Annealing algorithms (SA), Memetic Algorithm (MA), Evolution Strategy (ES), and Neuro Evolution (NE) [11-14]. SI uses the sharing of information by individuals in a population to enable the movement of the whole population to change from disorder to order. Particle Swarm Optimization (PSO) finds the optimal solution through collaboration and information sharing among individuals in the group [15]. Grey Wolf Optimizer (GWO) is designed to simulate the hunting behavior of grey wolves [16]. The Cuckoo Search algorithm (CS) is designed to observe and imitate the flight of cuckoos [17]. Cat Swarm Optimization (CSO) was designed based on feline predation strategies [18-19]. Fish Migration Optimization (FMO) is designed based on fish migratory behavior [20-21]. The basic idea of the Flower Pollination Algorithm (FPA) is derived from the simulation of self-pollination and cross-pollination of flowers in nature [22]. The Sparrow Search Algorithm (SSA) is designed based on the foraging and anti-predatory behaviors of sparrows [23]. The Equilibrium Optimizer (EO) is inspired by the control volume mass balance to estimate the dynamics and equilibrium states [24-25].

Meta-heuristic algorithms usually fall into local optimal solutions when dealing with high-dimensional feature selection problems [26-28]. Researchers have used parallel methods to deal with high-dimensional problems in recent years [29-30]. In this manuscript, parallel strategies are used to increase the information interaction capability between individuals and improved gradient search rules (GSR) are used to improve the exploration capability of the algorithm. the GBRUN algorithm is validated on five high-dimensional datasets and a real-world optimization problem provided by Arizona State University. The experimental results show

*Corresponding Author: Shu-Chuan Chu; E-mail: scchu0803@gmail.com
DOI: 10.53106/160792642024052503001

that the GBRUN algorithm achieves better results in FS problems, and the GBRUN algorithm demonstrates its ability to solve real-world problems in practical applications. The main contributions of this manuscript are as follows:

- Convert RUN algorithm to the binary version using S-, V-, and U-shaped transfer functions.
- Improving the performance of the BRUN algorithm using the enhanced Gradient Search method.
- Validate the performance of the GBRUN algorithm on high-dimensional datasets and compare it with other state-of-the-art algorithms in terms of classification accuracy and number of selected features.
- Combining GBRUN algorithm and EfficientNet to deal with the classification problem of COVID-19 CT images.

The manuscript is organized as follows: Section 2 introduces the basic concepts of the RUN algorithm. Section 3 shows how to convert the RUN algorithm, which is in continuous space, into a binary version and apply the improved GSR mechanism to the BRUN algorithm. Section 4 tests the performance of the GBRUN algorithm using five standard datasets. Section 5 introduces the GBRUN algorithm combined with EfficientNet to handle the classification problem of COVID-19 CT images. Section 6 gives the conclusions.

2 Runge Kutta Optimizer

Runge Kutta optimizer is an algorithm proposed by Iman et al. in 2021 [31]. The RUN algorithm is an intelligence model based on random conditions. The main idea of the RUN algorithm is inspired by the idea of calculating the slope in the Runge Kutta method [32]. Firstly, the RUN algorithm uses the fourth-order Runge-Kutta (RK4) equation to construct the RK4 search logic. Moreover, the RUN algorithm is based on the RK4 search logic to design a solution for position updating. Finally, the RUN algorithm introduces an Enhanced Solution Quality (ESQ) mechanism to improve the quality of the solution.

2.1 Search Logic

The RUN algorithm uses the fourth-order Taylor equation derived from the Runge Kutta Method as the main exploration logic, usually noted as the RK4 equation. It can be expressed as Equation 1.

$$y(x+\Delta x) = y(x) + \frac{1}{6}(k_1 + 2\times k_2 + 2\times k_3 + k_4)\times \Delta x . \quad \textbf{(1)}$$

Where k_1 is the slope of the curve S at the point (x, y) and k_2, k_3 and k_4 correspond to the slope of the point at $(x+\frac{\Delta x}{2}, y+\frac{\Delta y}{2}\times k_1)$, $(x+\frac{\Delta x}{2}, y+\frac{\Delta y}{2}\times k_2)$ and $(x+\frac{\Delta x}{2}, y+\frac{\Delta y}{2}\times k_3)$ respectively.

The RK4-based search mechanism can effectively improve the algorithm's exploration capability in the solution space. To further improve the search speed, RUN abandons the traditional method of adjusting the search position according to the fitness value of the objective function and uses a position-based update method for solving. Thus k_1 can be written as Equation 2.

$$k_1 = \frac{1}{2\Delta x}(R\times X_{rw} - u\times X_{rb}) . \quad \textbf{(2)}$$

$$u = round(1+R)\times(1-R) . \quad \textbf{(3)}$$

Where X_{rw} and X_{rb} represent the best and worst positions of the three randomly selected individuals. The parameter u increases the influence of the current global optimal solution on the next stage of the search process of the algorithm. Δx is the position increment and R is a random number between [0,1].

Based on the above equation, the remaining three parameters k_2, k_3, k_4 can likewise be rewritten as Equation 4, Equation 5, and Equation 6.

$$k_2 = \frac{1}{2\Delta x}(R\times(X_{rw} + R\times k_1\Delta x) - (u\times X_{rb} + R\times k_1\Delta x)) . \quad \textbf{(4)}$$

$$k_3 = \frac{1}{2\Delta x}(R\times(X_{rw} + R\times \frac{k_2}{2}\Delta x) - (u\times X_{rb} + R\times \frac{k_2}{2}\Delta x)) . \quad \textbf{(5)}$$

$$k_4 = \frac{1}{2\Delta x}(R\times(X_{rw} + R\times k_3\Delta x) - (u\times X_{rb} + R\times k_3\Delta x)) . \quad \textbf{(6)}$$

Ultimately, the equation for the RUN search mechanism can be written as Equation 7.

$$SM = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k4)\times\Delta x . \quad \textbf{(7)}$$

2.2 Update Method

The RUN algorithm starts searching for a random solution. It uses the RK4 method as the basic equation for position updating in subsequent iterations, with global and local searches designed for position updating. The local search is performed when $R < 0.5$ and the global search is performed when $R \geq 0.5$. The location update method is shown as Equation 8.

$$X_n^{(t+1)} = \begin{cases} X_c\times(1 + r\times g\times SF) + SF\times SM + \mu\times X_s & R < 0.5 \\ X_m\times(1 + r\times g\times SF) + SF\times SM + \mu\times X_s & R \geq 0.5 \end{cases} . \quad \textbf{(8)}$$

$X_n^{(t+1)}$ represents the individual position of the $n-th$ individual at the $(t+1)-th$ iteration, r is an integer taking values of -1, 0 or 1. g is a random number taking values in the range [0,2] and μ is a random number based on the standard normal random number *randn*. μ denoted as Equation 9.

$$\mu = 0.5 + 0.1 \times randn\,. \tag{9}$$

SF function is used to balance the exploration and exploitation phases of the algorithm. The *SF* decreases as the number of iterations increases, which is defined as Equation 10.

$$SF = 2 \times (0.5 - R) \times \exp(-b \times R \times \frac{t}{Max_{iter}})\,. \tag{10}$$

The $b = 2$ is a constant value. Max_{iter} is the maximum number of iterations, and t represents the current number of iterations.

X_c, X_m, X_s, and $X_{s'}$ represent the positions of the individuals., X_c, X_m, X_s, and $X_{s'}$ can be represented by Equation 11, Equation 12, Equation 13 and Equation 14.

$$X_c = R \times X^{(t)} + (1 - R) \times X_{r1}\,. \tag{11}$$

$$X_m = R \times X_{best} + (1 - R) \times X_{lbest}\,. \tag{12}$$

$$X_s = randn \times (X_m - X_c)\,. \tag{13}$$

$$X_{s'} = randn \times (X_{r1} - X_{r2})\,. \tag{14}$$

X_{best} represents the best position that can be reached. X_{lbest} represents the best position that can be reached in the current iteration. X_{r1}, X_{r2} represent the positions of two randomly selected individuals, where $X_{r1} \neq X_{r2} \neq X_n$.

2.3 Enhanced Solution Quality Mechanism

In the RUN algorithm, the solution quality is improved by applying the Enhanced Solution Quality (ESQ) mechanism. ESQ mechanism calculates the average of three random solutions X_{avg} and generates a new solution X_{new} by combining the location of the solution X_{best}. The final equation to generate the new solution X_{new1} is shown as Equation 15.

$$X_{new1} = \begin{cases} X_{new} + R \times w \times \left| X_{new} - X_{avg} + randn \right|, & w < 1 \\ X_{new} - X_{avg} + r \times w \times \left| R \times X_{new} - X_{avg} + randn \right|, & w \geq 1. \end{cases}. \tag{15}$$

$$X_{new} = R \times X_{avg} + (1 - R) \times X_{best}\,. \tag{16}$$

Parameter W is shown as Equation 17, which decreases with increasing number of iterations.

$$w = 2 \times R \times \exp(-5 \times R \times \frac{t}{Max_{iter}})\,. \tag{17}$$

The X_{new1} generated in this solution may not necessarily have a better fitness value than the optimal global solution. In order to create a better solution, the RUN algorithm designs an additional solution to generate a new position X_{new2}. X_{new2} is calculated as Equation 18.

$$X_{new2} = (1 - R) \times X_{new1} + (SF \times (R \times X' + r \times X_{best} - X_{new1}))\,. \tag{18}$$

Algorithm 1 and Figure 1 give the pseudo-code and flowchart for the RUN.

3 Gradient Search-based Binary Runge Kutta optimizer (GBRUN)

The RUN algorithm is designed based on the Runge Kutta Method and searches for the optimal solution using the improved Runge Kutta method. Moreover, it balances the exploration and exploitation capability of the algorithm with the help of the ESQ mechanism to avoid falling into local optimal solutions during the iteration. The authors, Iman et al., have demonstrated the better performance of the RUN algorithm than other algorithms in benchmark functions and practical engineering applications. Because of this, the BRUN algorithm is designed to solve the FS problem.

Algorithm 1. The pseudo-code of RUN

Initialize the individuals' positions X_n (n = 1, 2, ..., N)
Calculate the Fitness value $Fitness_n$ (n = 1, 2, ..., N) for each individual
while $t \leq Max_{iter}$ **do**
  for n=1:N **do**
    Calculate the position of $X_n^{(t+1)}$ according to Eq. (7) and Eq. (9)
    Calculate the fitness value $Fitness_{new}$ for individual $X_n^{(t+1)}$
    if $Fitness_{new} < Fitness_n$ **then**
      Update position X_n and Fitness value $Fitness_n$
    end if
    if $R < 0.5$ **then**
      Calculate position X_{new} using Eq. (16)
      Calculate position X_{new1} using Eq. (15)
      Calculate the fitness value $Fitness_{new1}$ for individual X_{new1}
      if $Fitness_{new1} < Fitness_n$ **then**
        Update position X_n and Fitness value $Fitness_n$
      else
        Calculate position X_{new2} using Eq. (17)
        Calculate the fitness value $Fitness_{new2}$ for Individual X_{new2}
        if $Fitness_{new2} < Fitness_n$ **then**
          Update position X_n and Fitness value $Fitness_n$
        end if
      end if
    end if
  end for
end while

In FS problems, the problem of how to process feature data efficiently is still a pressing issue. The feature data is

often high-dimension, non-linear, and has many redundant features. Therefore the feature selection problem often yields multiple locally optimal solutions. This requires the algorithm to have a strong exploration capability during the iterative process and to avoid falling into local optimal solutions as much as possible. The RUN algorithm is used for problems with continuous values, so converting the RUN algorithm to a binary version is necessary.

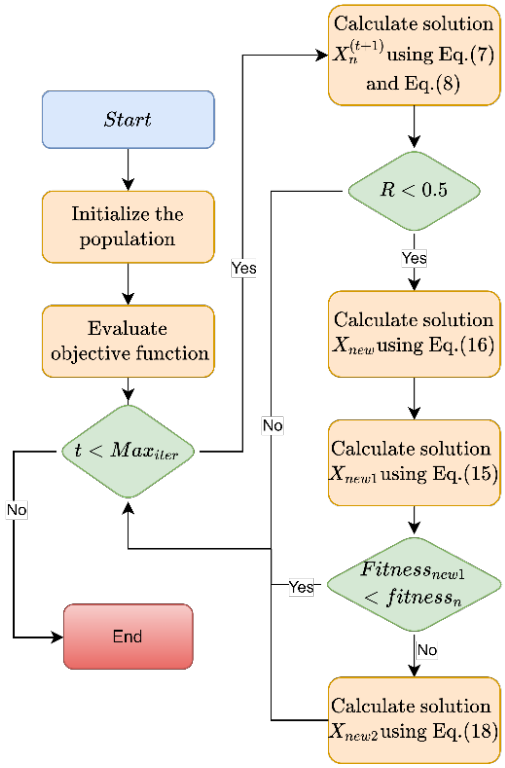

Figure 1. The flowchart of RUN algorithm

To achieve this, the RUN algorithm replaces the position update method with a 0-1 value update method by transfer functions. The BRUN algorithm changes the position of an individual in the 0-1 hypercube by changing the 0-1 value. The transfer function describes the probability that an individual will change from 0 to 1 or from 1 to 0. In this manuscript, three transfer functions are designed to convert the RUN algorithm's position update method to update probabilities with 0-1 values.

In addition, existing data tend to vary towards high-dimension and high complexity. In order to obtain better exploration capability of the algorithm, an improved GSR is introduced in this study to improve the search capability of the BRUN algorithm. The GSR mechanism is based on a gradient search method and is combined with Newton's method to effectively improve the exploration capability of the algorithm in solution space.

3.1 Initialization

The RUN algorithm generates a random initial position by giving the function the upper bounds (ub) and lower bounds (lb). In the BRUN algorithm, the initial position is created by a random method. The initial position method can be denoted as Equation 19.

$$X_{(n,d)} = \begin{cases} 0, & R \geq 0.5 \\ 1, & R < 0.5 \end{cases}. \tag{19}$$

$X_{(n,d)}$ represents the position of the $n-th$ individual in the $d-th$ dimension.

3.2 Transfer Functions

Transform the algorithm to a binary version using a transfer function is efficient and straightforward. The transfer function maps the velocity values of the individual moves in the RUN algorithm to the range [0,1]. The structure of the RUN algorithm is preserved to the maximum extent possible. The S-, V-, and U-Shaped transfer functions used in this manuscript are represented by Equation 20, Equation 21, and Equation 22. The transfer functions curves are shown in Figure 2, Figure 3, and Figure 4.

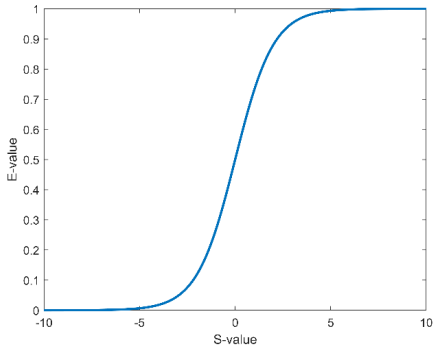

Figure 2. The curve of S-Shape transfer function

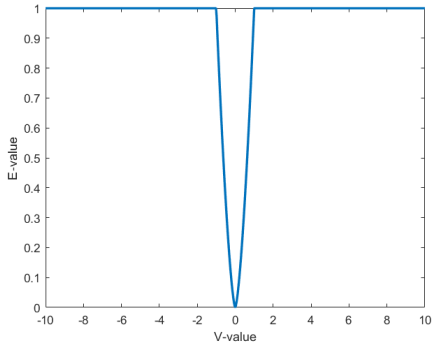

Figure 3. The curve of V-Shape transfer function

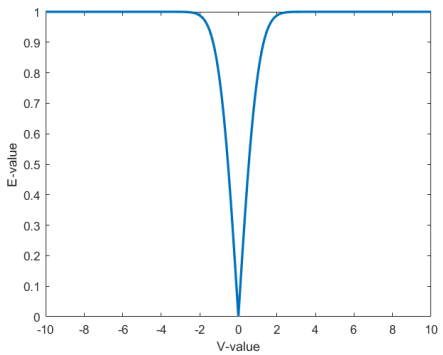

Figure 4. The curve of U-Shape transfer function

$$S(n,d)=\frac{1}{1+e^{-E(n,d)}}. \tag{20}$$

$$V(n,d)=\left|erf\left(\frac{\sqrt{\pi}}{2}\times E(n,d)\right)\right|. \tag{21}$$

$$U(i,d)=min\left(\left|E(n,d)^{1.5},1\right|\right). \tag{22}$$

$E(n, d)$ represents the velocity value of the $n-th$ individual in the $d-th$ dimension.

The transfer function is used to obtain the probability of individual position transformation in the 0-1 space. Then, a random number R is introduced to increase the randomness of the position transformation. The positions of the individuals were updated using Equation 23, Equation 24, and Equation 25, respectively.

$$X_{(n,d)}^{t+1}=\begin{cases}0, & R<S(n,d)\\ 1, & R\geq S(n,d)\end{cases}. \tag{23}$$

$$X_{(n,d)}^{t+1}=\begin{cases}X_{(n,d)}^{t+1\ -1}, & R<V(n,d)\\ X_{(n,d)}^{t+1}, & R\geq V(n,d)\end{cases}. \tag{24}$$

$$X_{(n,d)}^{t+1}=\begin{cases}X_{(n,d)}^{t+1\ -1}, & R<U(n,d)\\ X_{(n,d)}^{t+1}, & R\geq U(n,d)\end{cases}. \tag{25}$$

Where $X_{(n,d)}^{t}$ and $X_{(n,d)}^{t+1}$ represent the position of the $n-th$ individual's $d-th$ dimension at the $t-th$ iteration and the $(t+1)-th$ iteration, respectively. When using the S, U, V transfer function, the position is updated using the Equation 23, Equation 24, Equation 25 respectively. S-shape functions convert infinite continuous space values to values between [0,1] in a stable manner and update individual positions to 0 or 1. V-Shaped function and U-Shaped function do not force the individual position to set 0 or 1. The individual position will only move to the opposite position if the individual moves at a large speed; otherwise, it will not change.

3.3 Gradient Search Rule (GSR)

GSR starts from the current solution and explores the solution space along a specified gradient [33]. The GSR mechanism uses the numerical gradient method instead of function derivation, effectively avoiding many cases where the optimization problem is not differentiable.

$$GSR=randn\times p1\times\frac{2\Delta x\times X_n}{yp_n-yq_n+\varepsilon}. \tag{26}$$

Where $p1$ decreases as the number of iterations increases. $p1$ is used to balance the GSR exploration and exploitation process. ε is a random number between [0,0.1]. yp_n and yq_n are two positions obtained based on Newton's method to improve the GSR performance design, they can be expressed as Equation 27 and Equation 28.

$$yp_n=R\times\left(\frac{Z_n+X_n}{2}+R\times\Delta x\right). \tag{27}$$

$$yq_n=R\times\left(\frac{Z_n+X_n}{2}-R\times\Delta x\right). \tag{28}$$

Z_n can be expressed as Equation 29.

$$Z_n=X_n-R\times\frac{2\Delta x\times X_n}{X_{worst}-X_{best}+\varepsilon}. \tag{29}$$

X_{worst} represents the worst position. Δx is determined by X_{best}, X_{r1} and the parameter δ. δ is determined by randomly selecting four positions X_{r1}, X_{r2}, X_{r3} and X_{r4}, where ($r1\neq r2\neq r3\neq r4\neq n$). In order to improve the search capability of GSR, a random number R is introduced here, taking values in the range [0,1]. Δx can be expressed as Equation 30, δ can be expressed as Equation 31.

$$\Delta x=R\times\frac{X_{best}-X_{r1}+\delta}{2}. \tag{30}$$

$$\delta=2\times R\times\left(\frac{X_{r1}+X_{r2}+X_{r3}+X_{r4}}{4}-X_n\right). \tag{31}$$

To increase the influence of all individuals in the population on the change in the current individual's position, we rewrite the parameter δ as Equation 32.

$$\delta=2\times R\times\left(\frac{\Sigma_{i=1}^{N}}{N}-X_n\right). \tag{32}$$

N represents the number of individuals, and X_i represents the position of the $i-th$ individual.

In addition, the GSR uses the Direction of Movement (DM) method, which helps to explore the area around the individual better. DM can represent as Equation 33.

$$DM=R\times p2\times(X_{best}-X_n). \tag{33}$$

$$p2=2\times R\times\alpha-\alpha. \tag{34}$$

According to the GSR and DM methods, the position of $X1$ can be calculated by Equation 35.

$$X1=X_n-GSR+DM. \tag{35}$$

Likewise $X1$ can be expressed as Equation 36.

$$X1=X_n-randn\times p1\times\frac{2\Delta x\times X_n}{yp_n-yq_n+\varepsilon}+R\times p2\times(X_{best}-X_n). \tag{36}$$

However, Equation 36 is suitable for global search but not for local search. The position of $X2$ is obtained by replacing X_n and X_{best} in Equation 36. $X2$ is shown in Equation 37. Equation 37 performs a local search using the positions of X_{r1}, X_{r2} and X_{best}.

$$X2 = X_{best} - randn \times p1 \times \frac{2\Delta x \times X_n}{yp_n - yq_n + \varepsilon} + R \times p2 \times (X_{r1} - X_{r2}). \quad (37)$$

Finally, based on the positions of $X1$ and $X2$, the new position $X_n^{(t+1)}$ be represented as Equation 38.

$$X_{GSR}^{(t+1)} = R \times (R \times X1 + (1-R) \times X2) + (1-R) \times (X_n - p1 \times (X2 - X1)). \quad (38)$$

In addition, an Execution Control function (EC) has been added to this manuscript. The EC decreases as the number of iterations increases. This function controls the GSR mechanism to be executed with a higher probability in the early stages of the algorithm to balance the exploration and exploitation capabilities of the algorithm. The EC function is expressed as Equation 39, and the curve of the EC is shown in Figure 5.

$$C = min((1 - \frac{i}{Max_{iter}} \times \theta + \frac{i}{Max_{iter}})^3 \times \frac{1}{Max_{iter}}, 1). \quad (39)$$

Parameter $\theta = 0.01$ is a constant value.

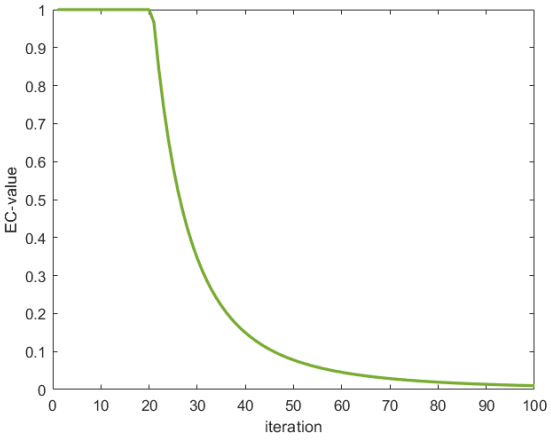

Figure 5. The execution control function evolution curve

The final GBRUN algorithm pseudo-code is represented in Algorithm 2, and the flowchart is represented in Figure 6.

Algorithm 2. The pseudo-code of GBRUN

Initialize the individuals' positions $X_n (n = 1, 2, ..., N)$ using Eq. (18)
Calculate the Fitness value $Fitness_n (n = 1, 2, ..., N)$ for each individual
while $t \leq Max_{iter}$ **do**
  for n=1:N **do**
    if $C \geq R$ **then**
      Calculate position $X_n^{(t+1)}$ using Eq. (37)
      if fitness of $X_{GSR}^{(t+1)}$ < fitness of X_n then
        Update position X_n and Fitness value $Fitness_n$
      end if
    end if
  end for
  for n=1:N **do**
    Calculate the position of $X_n^{(t+1)}$ according to Eq. (7), Equation 8 and transfer function
    Calculate the fitness value $Fitness_{new}$ for individual $X_n^{(t+1)}$
    if $Fitness_{new} < Fitness_n$ **then**
      Update position X_n and Fitness value $Fitness_n$
    end if
    if $R < 0.5$ **then**
      Calculate position X_{new} using Eq. (16) and transfer function
      Calculate position X_{new1} using Eq. (15) and transfer function
      Calculate the fitness value $Fitness_{new1}$ for individual
      if $Fitness_{new1} < Fitness_n$ **then**
        Update position X_n and Fitness value $Fitness_n$
      else
        Calculate position X_{new2} using Eq. (17) and transfer function
        Calculate the fitness value $Fitness_{new2}$ for Individual X_{new2}
        if $Fitness_{new2} < Fitness_n$ **then**
          Update position X_n and Fitness value $Fitness_n$
        end if
      end if
    end if
  end for
end while

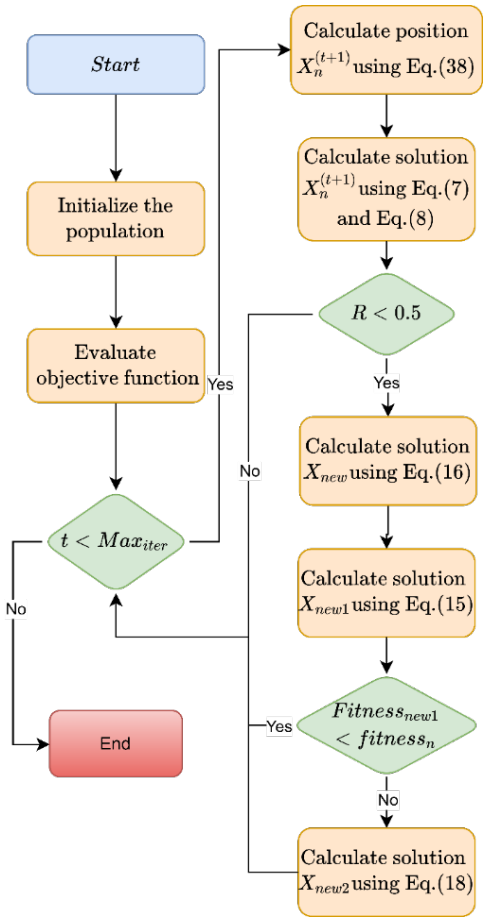

Figure 6. Flowchart of GBRUN algorithm

4 GBRUN for Feature Selection

FS is a multi-objective optimization problem. The effect of FS should be considered in two metrics: classification accuracy and the number of selected features. The ideal result is a dynamic balance between these two evaluation metrics. In other words, FS should achieve higher classification accuracy while selecting fewer features.

4.1 Experimental Setup and Dataset Explanation

Both Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) have commonly used classification algorithms [34-35]. The KNN classifier is used as an evaluator in this experiment. The KNN classifier first calculates the distance between unknown and known instances based on the parameter *K*. The obtained distances are sorted in ascending order. If the *K* most similar samples in a given feature space belong to a certain class, the sample will be classified into that class. The more relevant the features in the feature subset, the higher the classification accuracy. This manuscript uses a wrapper- based approach as the fitness function, as shown in Equation 40.

$$Fitness = \lambda \times Acc + \eta \times \left| \frac{F_S}{F_M} \right|. \quad \textbf{(40)}$$

Where λ is equal to 0.95 and η is equal to 0.05. λ, η are used to balance the two evaluation metrics. *Acc* is the classification accuracy. F_S is the number of features selected, and F_M is the total features number of the dataset.

Five standard datasets provided by the Data Mining and Machine Learning Laboratory of Arizona State University were used in this manuscript. The parameters such as dataset attributes are shown in Table 1.

To verify the performance of the proposed GBRUN algorithm, BRUN-S, BRUN-V, BRUN-U, Binary JAYA (BJAYA) [36], Binary Whale Optimization Algorithm (BWOA) [37], Binary Harris Hawks Optimizer (BHHO) [38], Binary Grey Wolf Optimization (BGWO) [39], binary Salp Swarm Algorithm (BSSA) [40], and Binary Sine–Cosine Algorithm (BSCA) are selected for comparison with the GBRUN algorithm in this experiment [41]. Table 2 gives information about the attributes of the datasets.

4.2 Results Analysis

In this section, the proposed algorithm is compared with BRUN-S, BRUN-V, BRUN-U, BJAYA, BWOA, BHHO, BGWO, BSSA, and BSCA in classification accuracy, the number of selected features, and fitness values. The following discussion revolves around the algorithm's performance on these three evaluation metrics.

Table 3 shows the average classification accuracies of the GBRUN algorithm and other comparison algorithms on the five datasets. Table 3 shows that the proposed GBRUN-U achieves the highest classification accuracy on *GLIOMA*, *Yale*, *colon*, and *warpPIE*10*P* datasets. GBRUN-V achieves the highest classification accuracy on *GLIOMA*, and *lung* datasets. The algorithm performance is significantly improved with the introduction of parallelism and improved GSR strategy.

Table 4 shows the average number of features selected by the algorithm on the dataset. The GBRUN-V algorithm selected the minimum number of features on the *Yale* and *lung* datasets. The GBRUN-U algorithm selected the minimum number of features on the *GLIOMA*, *colon*, and *warpPIE*10*P* datasets. The GBRUN algorithm has a significant advantage in selecting the number of features compared to BRUN and other comparison algorithms.

Table 1. The datasets' descriptions

Data set	Set no.	Number of instances	Number of features	Number of classes	Keywords
GLIOMA	DS1	50	4434	4	continuous, multi-class
Yale	DS2	165	1024	15	continuous, multi-class
colon	DS3	62	2000	2	discrete, binary
warpPIE10P	DS4	210	2420	10	continuous, multi-class
lung	DS5	203	3312	5	continuous, multi-class

Table 2. Parameter settings of the considered algorithms

Algorithms	Parameters	Values
Common to all algorithms	K for cross-validation	10
	Number of iterations	100
	Population size	10
	Number of runs	30
	Dimensions	Number of features
	Domain	{0,1}-Binary
GBRUN	Transfer Function	S, V, U
BRUN	Transfer Function	S, V, U
BJAYA	lb, ub	0, 1
BWOA	b	1
BHHO	β	1.5
BGWO	α	[0,2]
BSSA	lb, ub and Transfer Function	0, 1 and S
BSCA	lb, ub	0, 1

Table 3. Mean classification accuracy

Data set	GBRUN-S	GBRUN-V	GBRUN-U	BRUN-S	BRUN-V	BRUN-U	BJAYA	BWOA	BHHO	BGWO	BSSA	BSCA
DS1	0.1617	**0.2725**	**0.2725**	0.1553	0.2133	0.2122	0.0914	0.1542	0.1611	0.1600	0.1361	0.1406
DS2	0.0963	0.1397	**0.1523**	0.0968	0.1302	0.1334	0.0589	0.0932	0.0961	0.1099	0.0857	0.0923
DS3	0.3135	0.3683	**0.3882**	0.3177	0.3388	0.3388	0.2325	0.3118	0.3123	0.3175	0.3090	0.3072
DS4	0.2254	0.4629	**0.4473**	0.2257	0.3553	0.2973	0.1689	0.2242	0.2277	0.2327	0.2208	0.2236
DS5	0.8656	**0.9100**	0.9050	0.8671	0.8926	0.8976	0.8421	0.8643	0.8652	0.8749	0.8645	0.8636

Table 4. Mean selected feature subsets

Data set	GBRUN-S	GBRUN-V	GBRUN-U	BRUN-S	BRUN-V	BRUN-U	BJAYA	BWOA	BHHO	BGWO	BSSA	BSCA
DS1	2433.80	647.40	**509.40**	2340.30	929.07	1136.10	2760.47	2463.63	2141.20	2271.63	3244.40	2781.73
DS2	543.00	**137.07**	155.93	577.77	205.23	251.30	636.13	548.83	504.07	529.07	695.50	645.53
DS3	1161.87	363.20	**354.23**	1213.90	547.53	734.83	1250.57	1144.10	1088.50	1034.03	1499.93	1280.50
DS4	1214.20	142.90	**98.57**	1187.43	251.10	505.83	1482.60	1348.30	1146.33	1245.27	1751.53	1501.20
DS5	1732.83	**217.23**	387.43	1830.80	538.37	672.87	2061.77	1832.67	1657.30	1733.50	2400.50	2101.20

Table 5. Mean fitness values

Data set	GBRUN-S	GBRUN-V	GBRUN-U	BRUN-S	BRUN-V	BRUN-U	BJAYA	BWOA	BHHO	BGWO	BSSA	BSCA
DS1	0.8239	0.6984	**0.6969**	0.8289	0.7578	0.7612	0.8943	0.8313	0.8211	0.8236	0.8554	0.8478
DS2	0.8851	0.8240	**0.8129**	0.8863	0.8364	0.8356	0.9251	0.8882	0.8833	0.8715	0.8985	0.8939
DS3	0.6812	0.6092	**0.5901**	0.6786	0.6418	0.6465	0.7604	0.6824	0.6805	0.6742	0.6939	0.6902
DS4	0.7610	**0.5132**	0.5271	0.7601	0.6176	0.6780	0.8202	0.7649	0.7574	0.7547	0.7747	0.7686
DS5	0.1539	**0.0887**	0.0961	0.1539	0.1102	0.1074	0.1812	0.1566	0.1530	0.1450	0.1649	0.1613

Table 5 shows the performance of all algorithms. GBRUN-V achieves optimal fitness on the *warpPIE*10*P* and *lung* datasets. GBRUN-U achieves optimal fitness on the *GLIOMA*, *Yale*, and *colon* datasets. The parallel strategy can effectively improve the inter-individual information interaction ability and increase the search stability, and the improved GSR mechanism can enhance the algorithm search ability and achieve better performance.

The above data show that the parallel strategy can effectively improve the inter-individual information interaction ability and increase the search stability, and the improved GSR mechanism can enhance the algorithm search ability and achieve better performance. In particular, the GBRUN algorithm can filter out redundant features more effectively in terms of the number of features selected.

5 Combining with Neural Networks for Feature Selection Problems

Since the outbreak of COVID-19, many researchers have proposed machine learning and deep learning-based methods to classify images of COVID-19 [42-44]. However, neural networks often contain huge number of parameters, and it is exceptionally difficult to train neural networks from scratch. This manuscript proposes a method to perform migration learning using trained neural networks, extract the output features of the previous layer of the fully connected layer of the model after migration learning, and use the GBRUN algorithm to perform feature selection on the extracted features for image classification. This manuscript combines the GBRUN algorithm of EfficientNet to classify the CT images of COVID-19 [45].

5.1 EfficientNet

Convolutional neural networks are usually developed with a fixed resource budget and scaled up to obtain better accuracy if more resources are available [46-47]. Mingxing Tan and Quoc V. Le propose an efficient and straightforward composite coefficient to scale up the network from depth, width, and resolution dimensions. Experiments show that EfficientNet is much faster than other networks and has higher accuracy.

This manuscript uses EfficientNet for feature extraction of CONVID-19 CT images. EfficientNet designs the baseline network by multi-objective neural structure search.After the image input A, it first passes through the convolution layer, followed by the MBConv stage, the pooling stage, and the output stage.

EfficientNet provides a pre-trained model of its benchmark network EfficientNet-b0. First, this manuscript transfers the retained weights to the new model by transfer learning so that the new model can perform the binary classification task [48]. Finally, the 1280-dimensional feature vector of the pooling layer is extracted for the next step of feature selection. efficientNet-b0 and its detailed parameters are shown in Table 6.

Table 6. EfficientNet-b0 baseline network

Stage	Operator	Image resolution	Channels	Layers
1	Conv 3×3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 Pooling & FC	7×7	1280	1

5.2 Database Used

In this section, the COVID-19 CT database was chosen to predict COVID-19 [49]. There are 699 chest CT images of COVID-19 patients and 397 chest CT images of non-COVID-19 patients in the COVID-19 CT dataset. The accuracy of this dataset has been confirmed by the physicians involved [50].

5.3 Experimental Settings

In this experiment, SVM is used to classify the feature data. The approach is to find a hyperplane that separates the features of a given data set into two classes. SVM trains a model that maps the training samples as points in space and maximizes the separation between the two categories. Finally, it maps the test samples into space and determines which category they belong to based on their position in space. There are five evaluation metrics in SVM: *Accuracy*, *Recall*, *Precision*, $F1-Score$, Area Under Curve (AUC) and Receiver Operating Curve (ROC).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (41)$$

$$Recall = \frac{TP}{TP + FN}. \quad (42)$$

$$Precision = \frac{TP}{TP + FP}. \quad (43)$$

$$F1 - Score = \frac{Precision \times Recall}{Precision + Recall}. \quad (44)$$

- True Positive (TP): COVID-19 patients were diagnosed as positive.
- True Negative (TN): Healthy people were diagnosed as negative.
- False Positive (FP): Healthy people diagnosed with COVID-19 infection.
- False Negative (FN): COVID-19 patients were diagnosed as negative.

The relevant parameters regarding the experiment are shown in Table 7.

Table 7. Parameter settings of the considered algorithms

Algorithms	Parameters	Values
Common to all algorithms	Kernel	rbf
	Test Size	40%
	Number of iterations	100
	Population size	10
	Dimensions	1280
	Domain	{0,1}-Binary
GBRUN	Transfer Function	S, V, U
BRUN	Transfer Function	S, V, U
BJAYA	lb, ub	0, 1
BWOA	b	1
BHHO	β	1.5
BGWO	α	[0,2]
BSSA	lb, ub and Transfer Function	0, 1 and S
BSCA	lb, ub	0, 1

Table 8. Results of the considered algorithms

Evaluation indicators	GBRUN-S	GBRUN-V	GBRUN-U	BRUN-S	BRUN-V	BRUN-U	BJAYA	BWOA	BHHO	BGWO	BSSA	BSCA
Fitness	0.0721	**0.0292**	0.0361	0.0754	0.0420	0.0510	0.0869	0.0778	0.0763	0.0851	0.0880	0.0744
Accuracy	0.9404	**0.9725**	0.9679	0.9450	0.9587	0.9541	0.9450	0.9450	0.9450	0.9450	0.9450	0.9495
Recall	0.9510	**0.9902**	0.9706	0.9608	0.9706	0.9608	0.9510	0.9510	0.9608	0.9510	0.9510	0.9510
Precision	0.9238	**0.9528**	0.9612	0.9245	0.9429	0.9423	0.9327	0.9327	0.9245	0.9327	0.9327	0.9417
F1-core	0.9372	**0.9712**	0.9659	0.9423	0.9565	0.9515	0.9417	0.9417	0.9423	0.9417	0.9417	0.9463
AUC	0.9775	**0.9887**	0.9842	0.9830	0.9855	0.9837	0.9817	0.9819	0.9838	0.9806	0.9814	0.9807
Feature subsets	396	77	142	591	**72**	190	898	681	619	846	904	664

5.4 Experimental Results and Discussion

Figure 7 shows the convergence curves of all algorithms. As shown in Figure 7, the GBRUN algorithm can obtain better fitness values than BRUN and other advanced algorithms in the early iterations. During the 10th iteration to the 85th iteration, the GBRUN algorithm relies on its powerful search ability to search for better solutions in high-dimensional space. The best fitness value is obtained for GBRUN-V in the proposed algorithm.

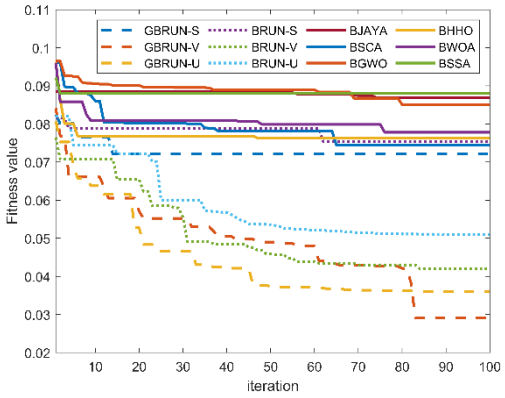

Figure 7. The convergence curves of all algorithm

Figure 8, Figure 9, and Figure 10 are the *ROC* curves of GBRUN-S, GBRUNV and GBRUN-U, respectively.

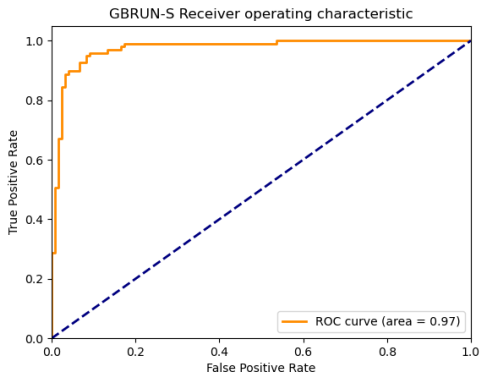

Figure 8. The ROC curve of GBRUN-S

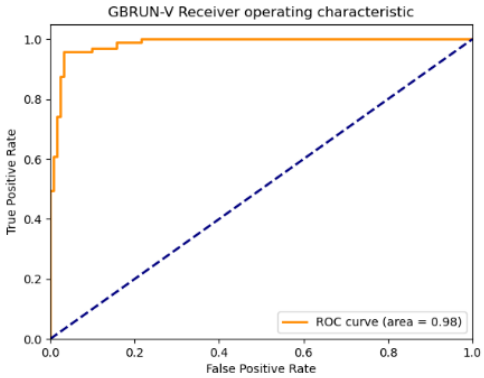

Figure 9. The ROC curve of GBRUN-V

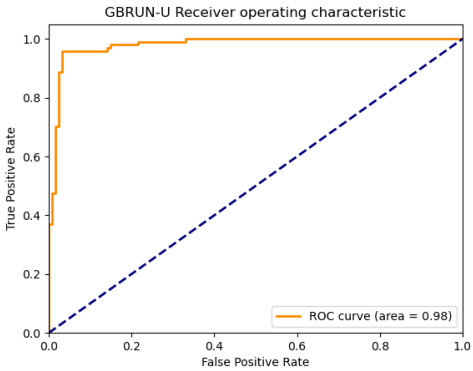

Figure 10. The ROC curve of GBRUN-U

The experimental results of all algorithms in *Acc*, *recall*, *precision*, $F1-Score$, and *AUC* are shown in Table 8. The Table 8 shows that the proposed GBRUN-V, GBRUN-U method can reduce the number of features from 1280 to 77 and 142, which reduces the feature dimensionality by 85%-90% compared to BJAYA, BWOA, BHHO, BGWO, BSSA, and BSCA. Analysis of the data in Table 8 shows that GBRUN-V achieves the best fitness value of 0.0291, improving 66.51%, 60.88%, 65.76%, 61.86%, 62.59% and 66.93% over the BJAYA, BSCA, BGWO, BHHO, BWOA, BSSA algorithms. Similarly, the GBRUN algorithm has been improved to varying degrees relative to other comparative algorithms in *Accuracy*, *Recall*, *Precision*, $F1-Score$, and *AUC* evaluation metrics.

6 Conclusion

In this manuscript, three different transfer functions are designed to convert the continuous RUN algorithm to the BRUN algorithm. The improved GSR mechanism is applied to BRUN to enhance the exploration capability. The GBRUN algorithm gains a stronger exploration capacity, allowing it to quickly eliminate a large number of redundant features when dealing with high-dimensional FS problems. In order to evaluate the performance of the proposed algorithm, five standard datasets were selected for testing in this manuscript. The experimental results show that the proposed GBRUN algorithm outperforms other state-of-the-art algorithms in terms of the number of features selected and the classification accuracy.

In addition, this manuscript combines the GBRUN algorithm with EfficientNet for the COVID-19 CT image classification problem. The GBRUN algorithm achieves the best results on *Accuracy*, *Recall*, *Precision*, $F1-Score$, *AUC*, and *ROC* compared with other algorithms. The GBRUN algorithm is not yet combined with other neural networks to test the performance. The neural networks used need to perform migration learning first, which consumes a long time. Later, we will consider how to use untrained neural networks to extract features and reduce resource consumption.

References

[1] D. L. Donoho, High-dimensional data analysis: The curses and blessings of dimensionality, *AMS math challenges lecture*, pp. 1-32, August, 2000.

[2] V. Kumar, S. Minz, Feature selection: a literature review, *Smart Computing Review*, Vol. 4, No. 3, pp. 211-229, June, 2014.

[3] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, *ACM computing surveys*, Vol. 50, No. 6, pp. 1-45, November, 2018.

[4] Z. Beheshti, S. M. H. Shamsuddin, A review of population-based meta-heuristic algorithm, *International Journal of Advances in Soft Computing and its Applications*, Vol. 5, No. 1, pp. 1-35, March, 2013.

[5] T. Bäck, H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, *Evolutionary computation,* Vol. 1, No. 1, pp. 1-23, March, 1993.

[6] A. Chakraborty, A. K. Kar, Swarm Intelligence: A Review of Algorithms, in: S. Patnaik, X. S. Yang, K. Nakamatsu (Eds.), *Nature-Inspired Computing and Optimization: Theory and Applications*, Springer International Publishing, 2017, pp. 475-494.

[7] T.-T. Nguyen, T. Dong-Nguyen, T.-G. Ngo, V.-T. Nguyen, An Optimal Thresholds for Segmenting Medical Images Using Improved Swarm Algorithm, *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 13, No. 1, pp. 12-21, March, 2022.

[8] J.-S. Pan, M. Gao, J.-P. Li, S.-C. Chu, A compact GBMO applied to modify DV-Hop based on layers in a wireless sensor network, *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 39, No. 1-2, pp. 20-36, February, 2022.

[9] K. V. Price, Differential Evolution, in: I. Zelinka, V. Snášel, A. Abraham (Eds.), *Handbook of optimization*, Springer, 2013, pp. 187-214.

[10] S. Mirjalili, *Evolutionary algorithms and neural networks*, Springer, 2019.

[11] D. Floreano, P. Dürr, C. Mattiussi, Neuroevolution: from architectures to learning, *Evolutionary intelligence*, Vol. 1, No. 1, pp. 47-62, March, 2008.

[12] N. Hansen, The CMA Evolution Strategy: A Comparing Review, in: J. A. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea (Eds.), *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, Springer Berlin Heidelberg, 2006, pp. 75-102.

[13] J. D. Knowles, D. W. Corne, M-PAES: A memetic algorithm for multiobjective optimization, *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, California, USA, 2000, pp. 325-332.

[14] R. A. Rutenbar, Simulated annealing algorithms: An overview, *IEEE Circuits and Devices Magazine*, Vol. 5, No. 1, pp. 19-26, January, 1989.

[15] J.-J. Yan, J.-S. Fang, J. S.-H. Tsai, C.-H. Huang, S.-M. Guo, Robust Digital-Redesign Tracking control for Uncertain Systems: PID sliding mode Control and PSO Algorithm, *Journal of Network Intelligence*, Vol. 6, No. 4, pp. 668-687, November, 2021.

[16] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Advances in engineering software*, Vol. 69, pp. 46-61, March, 2014.

[17] X.-S. Yang, S. Deb, Cuckoo search: recent advances and applications, *Neural Computing and Applications*, Vol. 24, No. 1, pp. 169-174, January, 2014.

[18] L. Kong, J.-S. Pan, P. Tsai, S. Vaclav, J. N. 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, March, 2015.

[19] J.-S. Pan, X.-F. Ji, A. Liang, K.-C. Huang, S.-C. Chu, Parallel Binary Cat Swarm Optimization with Communication Strategies for Traveling Salesman Problem, *Journal of Internet Technology*, Vol. 22, No. 7, pp. 1621-1633, December, 2021.

[20] Q.-W. Chai, S.-C. Chu, J.-S. Pan, W.-M. Zheng, Applying Adaptive and Self Assessment Fish Migration Optimization on Localization of Wireless Sensor Network on 3-D Te rrain, *Information Hiding and Multimedia Signal Processing*, Vol. 11, No. 2, pp. 90-102, June, 2020.

[21] S.-C. Chu, X.-W. Xu, S.-Y. Yang, J.-S. Pan, Parallel fish migration optimization with compact technology based on memory principle for wireless sensor networks, *Knowledge-Based Systems*, Vol. 241, Article No. 108124, April, 2022.

[22] J.-S. Pan, T.-K. Dao, T.-S. Pan, T.-T. Nguyen, S.-C. Chu, J. F. Roddick, An Improvement of Flower Pollination Algorithm for Node Localization Optimization in WSN, *Information Hiding and Multimedia Signal Processing*, Vol. 8, No. 2, pp. 486-499, March, 2017.

[23] C.-Y. Ning, L.-C. Liao, T.-T. Nguyen, K.-C. Huang, J.-S. Pan, A Routing Optimization in Wireless Sensor Networks Based on Reverse Elite Sparrow Search Algorithm, *Journal of Network Intelligence*, Vol. 6, No. 4, pp. 763-775, November, 2021.

[24] P. Hu, S.-C. Chu, V. Snasel, J.-S. Pan, Binary Equilibrium Optimizer Algorithm, *Journal of Network Intelligence*, Vol. 7, No. 1, pp. 45-58, February, 2022.

[25] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, *Knowledge-Based Systems*, Vol. 191, Article No. 105190, March, 2020.

[26] M. Alshayeji, B. Behbehani, I. Ahmad, Spark-based parallel processing whale optimization algorithm, *Concurrency Computation: Practice Experience*, Vol. 34, No. 4, Article No. e6607, February, 2022.

[27] T. G. Crainic, M. Toulouse, Parallel Strategies for Meta-Heuristics, in: F. Glover, G. A. Kochenberger (Eds.), *Handbook of metaheuristics*, Springer, 2003, pp. 475-513.

[28] Y. Su, Y. Dai, Y. Liu, A hybrid parallel Harris hawks optimization algorithm for reusable launch vehicle reentry trajectory optimization with no-fly zones, *Soft Computing*, Vol. 25, No. 23, pp. 14597-14617,

December, 2021.
[29] G. Dheemanth, V. Skanda, R. Nagpal, Parallel Antlion Optimisation (ALO) and Grasshopper Optimization (GOA) for Travelling Salesman Problem (TSP), In: H. S. Saini, R. Sayal, A. Govardhan, R. Buyya (Eds.), *Innovations in Computer Science and Engineering*, Springer, 2021, pp. 787-793.
[30] H.-J. Wang, J.-S. Pan, T.-T. Nguyen, S. Weng, Distribution network reconfiguration with distributed generation based on parallel slime mould algorithm, *Energy*, Vol. 244, Article No. 123011, April, 2022.
[31] I. Ahmadianfar, A. A. Heidari, A. H. Gandomi, X. Chu, H. Chen, RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method, *Expert Systems with Applications*, Vol. 181, Article No. 115079, November, 2021.
[32] J. C. Butcher, A history of Runge-Kutta methods, *Applied numerical mathematics*, Vol. 20, No. 3, pp. 247-260, March, 1996.
[33] I. Ahmadianfar, O. Bozorg-Haddad, X. Chu, Gradient-based optimizer: A new metaheuristic optimization algorithm, *Information Sciences*, Vol. 540, pp. 131-159, November, 2020.
[34] W. S. Noble, What is a support vector machine?, *Nature biotechnology*, Vol. 24, No. 12, pp. 1565-1567, December, 2006.
[35] L. E. Peterson, K-nearest neighbor, *Scholarpedia*, Vol. 4, No. 2, Article No. 1883, February, 2009.
[36] M. A. Awadallah, M. A. Al-Betar, A. I. Hammouri, O. A. Alomari, Binary JAYA algorithm with adaptive mutation for feature selection, *Arabian Journal for Science Engineering and Engineering*, Vol. 45, No. 12, pp. 10875-10890, December, 2020.
[37] A. G. Hussien, D. Oliva, E. H. Houssein, A. A. Juan, X. Yu, Binary whale optimization algorithm for dimensionality reduction, *Mathematics*, Vol. 8, No. 10, Article No. 1821, October, 2020.
[38] T. Thaher, A. A. Heidari, M. Mafarja, J. S. Dong, S. Mirjalili, Binary Harris Hawks Optimizer for High-Dimensional, Low Sample Size Feature Selection, in: S. Mirjalili, H. Faris, I. Aljarah (Eds.), *Evolutionary machine learning techniques*, Springer, 2020, pp. 251-272.
[39] Q. Al-Tashi, S. J. A. Kadir, H. M. Rais, S. Mirjalili, H. Alhussian, Binary optimization using hybrid grey wolf optimization for feature selection, *IEEE Access*, Vol. 7, pp. 39496-39508, March, 2019.
[40] R. M. Rizk-Allah, A. E. Hassanien, M. Elhoseny, M. Gunasekaran, A new binary salp swarm algorithm: development and application for optimization tasks, *Neural Computing Applications*, Vol. 31, No. 5, pp. 1641-1663, May, 2019.
[41] K. S. Reddy, L. K. Panwar, B. Panigrahi, R. Kumar, A new binary variant of sine–cosine algorithm: development and application to solve profit-based unit commitment problem, *Arabian Journal for Science and Engineering*, Vol. 43, No. 8, pp. 4041-4056, August, 2018.
[42] G. Muhammad, M. S. Hossain, COVID-19 and non-COVID-19 classification using multi-layers fusion from lung ultrasound images, *Information Fusion*, Vol. 72, pp. 80-88, August, 2021.
[43] S.-H. Wang, V. V. Govindaraj, J. M. Górriz, X. Zhang, Y.-D. Zhang, Covid-19 classification by FGCNet with deep feature fusion from graph convolutional network and convolutional neural network, *Information Fusion*, Vol. 67, pp. 208-229, March, 2021.
[44] S.-H. Wang, D. R. Nayak, D. S. Guttery, X. Zhang, Y.-D. Zhang, COVID-19 classification by CCSHNet with deep fusion using transfer learning and discriminant correlation analysis, *Information Fusion*, Vol. 68, pp. 131-148, April, 2021.
[45] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, *International conference on machine learning*, Long Beach, California, USA, 2019, pp. 6105-6114.
[46] T. Kattenborn, J. Leitloff, F. Schiefer, S. Hinz, Review on Convolutional Neural Networks (CNN) in vegetation remote sensing, *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 173, pp. 24-49, March, 2021.
[47] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: analysis, applications, and prospects, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 33, No. 12, pp. 6999-7019, December, 2022.
[48] Y. Pathak, P. K. Shukla, A. Tiwari, S. Stalin, S. Singh, P. K. Shukla, Deep transfer learning based classification model for COVID-19 disease, *Irbm*, Vol. 43, No. 2, pp. 87-92, April, 2020.
[49] J. Zhao, Y. Zhang, X. He, P. T. Xie, Covid-ct-dataset: a ct scan dataset about covid-19, *arXiv preprint arXiv:.13865*, March, 2020. https://arxiv.org/abs/2003.13865v1
[50] M. Barstugan, U. Ozkaya, S. Ozturk, Coronavirus (covid-19) classification using ct images by machine learning methods, *arXiv preprint arXiv:.09424*, March, 2020. https://arxiv.org/abs/2003.09424

Biographies

Zhi-Chao Dou received his B.S. degree. B.S. in Software Engineering from Yantai Nanshan College. Master degree in Software Engineering from Shandong University of Science and Technology. His current research interests include population intelligence and image processing

Shu-Chuan Chu received the Ph.D. degree from the School of Computer Science, Engineering and Mathematics, Flinders University, South Australia, in 2004. She joined Flinders University, Australia, in December 2009. After nine years, she was with Cheng Shiu University, Taiwan. She has been a Research Fellow with the College of Science and Engineering, Flinders University, since December 2009. Currently she is a Ph.D. Advisor with

the College of Computer Science and Engineering, Shandong University of Science and Technology, since September 2019. Her research interests are mainly in swarm intelligence, intelligent computing, and data mining.

Zhongjie Zhuang received the B.S. degree in network engineering from Shandong Agricultural University, the M.S. degree in Computer application technology from Shandong University of Science and Technology, where she is currently pursuing the Ph.D. degree. Her current research interests include swarm intelligence, pattern recognition and image processing.

Ali Riza Yildiz is a Professor in the Department of Mechanical Engineering, Bursa Uludağ University, Bursa, Turkey. His research interests are the finite element analysis of structural components, lightweight design, vehicle design, vehicle crashworthiness, shape and topology optimization of vehicle components, meta-heuristicoptimization techniques, and additive manufacturing.

Jeng-Shyang Pan received the B.S. degree in electronic engineering from the National Taiwan University of Science and Technology in 1986, the M.S. degree in communication engineering from National Chiao Tung University, Taiwan, in 1988, and the Ph.D. degree in electrical engineering from the University of Edinburgh, U.K., in 1996. He is currently the Professor of Shandong University of Science and Technology. He is the IET Fellow, U.K., and has been the Vice Chair of the IEEE Tainan Section.