Binary QUATRE Using Time-varying Transfer Functions

Shu-Chuan Chu[1,2], Zhongjie Zhuang[1], Chia-Cheng Hu[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] College of Artificial Intelligence, Yango University, China
scchu0803@gmail.com, zhongjiezhuang@126.com, cchu.chiachenghu@gmail.com, jspan@cc.kuas.edu.tw

Abstract

QUasi-Affine TRansformation Evolutionary (QUATRE) is an excellent algorithm that generalized differential evolution algorithm to matrix form. There are two approaches of Binary QUATRE (BQUATRE) algorithm that convert the continuous search space to the binary one. The convergence process of meta-heuristic algorithms can be regarded as two stages: exploration and exploitation. Generally speaking, transfer function plays an important role in the binary meta-heuristic algorithms. Therefore, four families of transfer functions are converted into the time-varying form to balance the exploration and exploitation. In particular, a new time-varying formula for the V-shaped, U-shaped and Z-shaped transfer functions is designed in this manuscript. Then, two new approaches of BQUATRE algorithm using the time-varying transfer functions are proposed. In addition, a wrapper feature selection algorithm is designed based on the proposed time-varying BQUATRE. In order to verify the performance of the proposed algorithms, the UCI repository is used in the experiment and the results show that the proposed algorithms are superior to the original BQUATRE and the state-of-the-art algorithms.

Keywords: Binary, QUATRE, Time-varying transfer function, Feature selection

1 Introduction

In the past few decades, many researchers have developed various effective meta-heuristic optimization algorithms inspired by bionics. Particle Swarm Optimization (PSO) [1-4] simulated the foraging behavior of birds to obtain the optimal solution; Cat Swarm Optimization (CSO) [5-6] mimicked cats hunting behavior to get the optimal solution; Moth Flame Optimizer (MFO) [7] described an effective mechanism for moths to maintain a fixed Angle to the moon when they fly at night. Sine Cosine Algorithm (SCA) [8-9] creates multiple initial random candidate solutions and requires them to either wave outward or use a mathematical model based on Sine and Cosine functions to wave toward the optimal solution. Pigeon Inspired Optimization (PIO) [10-11] is designed by mimicking the homing behavior of the pigeon; Dragonfly Algorithm (DA) [12-13] is proposed to simulate the social interaction of dragonflies while navigating, finding food, and avoiding enemies. Fish Migration Optimization (FMO) [14-15] is inspired by the fish migration, the migration and the swimming model are integrated into the optimization process. Flower Pollination Algorithm (FPA) is generated by the pollination process of flowers [16-17].

In particular, QUATRE was proposed in paper [18] to improve the drawback of Differential Evolution (DE) that did not achieve equilibrium search in search space without prior knowledge. The binary version of QUATRE (B QUATRE) was proposed in paper [19] and used in hyperspectral image processing. In [20], a novel E-QUATRE algorithm in which an automatically generated evolution matrix with self-adaptive mechanism and an adaptive control parameter is proposed for the enhancement of the QUATRE algorithm. A hybrid algorithm WOA-QT based on the whale optimization (WOA) and the quasi-affine transformation evolutionary (QUATRE) algorithm is shown in the paper [21], which is used in Wireless Sensor Network (WSN) positioning. A surrogate-assisted quasi-affine transformation evolutionary (SA-QUATRE) algorithm was designed in the paper [22], which greatly reduced the running time. In Ref. [23], the O-QUATRE algorithm was actually implemented as a combination of both the QUATRE algorithm and the orthogonal array, both of which together secured an overall better performance on complex optimization problems.

For the binary version of the meta-heuristic algorithm, the role of the transfer function (TF) is very important. In recent years, scholars have proposed some time-varying transfer functions. A time-varying transfer function for balancing the exploration and exploitation ability was presented for binary PSO (BPSO) in paper [24] and a time-varying mirrored S-shaped transfer function was proposed for BPSO in paper [25]. In paper [13], the time-varying transfer function was used on Binary dragonfly optimization. In Ref. [26], the whale optimization algorithm (WOA) with time-varying transfer function was proposed for feature selection and obtained excellent performance. Furthermore, evolutionary algorithms can also be used for feature selection based on deep learning. A hybrid classification approach for COVID-19 images by combining the strengths of convolutional neural networks (CNN) to extract features and Marine Predators Algorithm (MPA) to select the most relevant features was proposed in paper [27]. Then, a bi-stage feature selection approach based on DA for deep features was designed in paper [28]. Therefore, it is necessary to apply the time-varying transfer function to binary QUATRE algorithm. In addition, the S-shaped transfer functions are converted into time-varying form by scholars so far, and the U-shaped and Z-shaped transfer functions have not.

*Corresponding Author: Jeng-Shyang Pan; E-mail: jspan@cc.kuas.edu.tw
DOI: 10.53106/160792642022052303001

So we designed a new time-varying formula for the V-shaped, U-shaped [29] and Z-shaped [30] transfer functions in this manuscript.

Dimensionality reduction is a very important preprocessing process in machine learning tasks [31-32]. Feature extraction and feature selection are both dimensionality reduction methods. The feature extraction method is mainly through the relationship between attributes, such as the combination of different attributes to get new attributes, so as to change the original feature space. The method of feature selection is to select a subset from the original feature data set, which is an inclusive relationship without changing the original feature space [33]. Feature selection is widely used in image recognition [34], text classification, and natural language processing and recommendation algorithms.

The main contributions of this paper are concluded as follows:

(1) A novel time-varying formula for the V-shaped, U-shaped and Z-shaped transfer functions is designed for the first time at this manuscript.
(2) A new binary QUATRE algorithm using time-varying transfer functions is proposed.
(3) This manuscript investigates the exploration and exploitation behavior of binary QUATRE with existing transfer functions and identifies their possible effects on the balance between exploration and exploitation.
(4) It implements feature selection based on the proposed algorithm.

The rest of the paper is organized as follows. Section 2 shows the related works include QUATRE and Binary QUATRE. Section 3 describes the time-varying transfer functions and the proposed algorithm. Section 4 is experimental results and analysis on feature selection. Section 5 depicts the main work of the paper and gives some suggestions for further work.

2. Related Works

2.1 The Standard QUATRE Algorithm

As it is known to all, the affine transformation can be defined by the following formula: $\mathrm{X} = MX + B$. The evolutionary framework used by the QUATRE algorithm is similar to affine transformation; the detailed evolution mode is given in Eq. (1).

$$\hat{X} = M \otimes \hat{X} + \bar{M} \otimes B \tag{1}$$

Where $\otimes$ is the bitwise multiplication of matrix elements. $\hat{X}$ is the position matrix, $\hat{X} = (X_1, X_2, \cdots, X_i, \cdots, X_{ps})^T$, where ps is the population size, $X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,d}, \cdots, x_{i,D})(i = 1, 2, \cdots ps; d = 1, 2, \cdots D)$ is the position vector of particle i , D is the individual dimension. B is the evolution guidance matrix, and its function is similar to the differential operation in differential evolution. B can be generated in many schemes, six of which are described in Table 1, where $\hat{X}_{r,1}, \hat{X}_{r,2}, \hat{X}_{r,3}, \hat{X}_{r,4}$ and $\hat{X}_{r,5}$ are random matrixes, F is the scaling factor, we usually constrain $F \in [0, 2]$. $\hat{X}_{gbest}$ is a matrix which can be obtained from Eq. (2), where X_{gbest} is the global best particle.

$$X_{gbest,G} = \begin{bmatrix} x_{gbest,G} \\ x_{gbest,G} \\ \cdots \\ x_{gbest,G} \end{bmatrix} \tag{2}$$

M is the binary evolutionary cooperation matrix, $\bar{M}$ is a matrix of binary inverse operations about M. Both M and $\bar{M}$ are made up of 0 and 1 . A special case of M and $\bar{M}$ is shown in Eq. (3).

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \bar{M} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3}$$

The evolutionary cooperation matrix M can be obtained from its initial matrix M_{init} which is a lower triangular matrix. Then, randomly swap each row and each column of M_{init}, the M can be obtain. In this way, QUATRE can achieve equilibrium search in search space. Eq. (4) describes the process of the transformation from M_{init} to M when the population size ps is equal to the dimension D.

$$M_{\mathrm{tmp}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} = M \tag{4}$$

If $ps > D$, the number of rows in evolutionary cooperation matrix M should be expanded according to ps Eq. (5) gives the expansion when the population size ps satisfies $ps = s * D + k$ and $ps\%D = k$ (% is the complementary operation). In detail, the first $s * D$ rows of M_{init} is consist of s times lower triangular matrices, and the last k rows is the first k rows of the triangular matrix. If $ps < D$, we can do similar operations to extend the columns of matrix M_{init} according to D.

$$M_{tmp} = \begin{bmatrix} 1 & & & \cdots & \\ 1 & 1 & & \cdots & \\ 1 & 1 & 1 & \cdots & \\ & & & \cdots & \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & & & \cdots & \\ 1 & 1 & & \cdots & \\ 1 & 1 & 1 & \cdots & \\ & & & \cdots & \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & & & \cdots & \\ 1 & 1 & & \cdots & \end{bmatrix} \sim \begin{bmatrix} 1 & & 1 & \cdots & \\ 1 & 1 & 1 & \cdots & 1 \\ & & 1 & \cdots & \\ 1 & & 1 & \cdots & 1 \\ & & & \cdots & \\ & 1 & & \cdots & 1 \\ & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ & & & \cdots & \\ & & 1 & \cdots & \\ 1 & & 1 & \cdots & \\ & 1 & & \cdots & \end{bmatrix} = M \tag{5}$$

2.2 The Binary QUATRE Algorithm

QUATRE algorithm is designed for continuous search space but many optimization problems are binary ones. Therefore, the binary QUATRE maps the continuous search space to the binary ones. There are two approaches of the Binary QUATRE algorithm. The first approach of Binary QUATRE algorithm (BQUATRE1) is described as follows. First of all, obtain the expanded cooperative search matrix M from M_{init} according to Eq. (5). $\bar{M}$ can be computed through binary inverse operation on the matrix M. Secondly, select one mutation strategy from Table 1 to calculate matrix B, and then the position $\hat{X}$ can be got by Eq. (1) that is a continuous matrix. Next, take the sigmoid transfer function as an example, the sigmoid function is used to convert every element of the continuous $\hat{X}$ to the binary matrix by Eq. (6) and Eq. (7), where $x_{i,d}^{cout}$ denotes the element $x_{i,d}$ is a continuous variable and $x_{i,d}^{bin}$ denotes the element is a binary one. The following steps are the same as the original QUATRE and will not be repeated.

Table 1. The six schemes of Matrix B calculation in QUATRE algorithm

Number	QUATRE/B	Equation
1	QUATRE/best/1	$B = X_{gbest,G} + F \times (X_{r1,G} - X_{r2,G})$
2	QUATRE/rand/1	$B = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G})$
3	QUATRE/target/1	$B = X_G + F \times (X_{r1,G} - X_{r2,G})$
4	QUATRE/target/2	$B = X_G + F \times (X_{r1,G} - X_{r2,G}) + F \times (X_{r3,G} - X_{r4,G})$
5	QUATRE/rand/2	$B = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G}) + F \times (X_{r4,G} - X_{r5,G})$
6	QUATRE/best/2	$B = X_{gbest,G} + F \times (X_{r1,G} - X_{r2,G}) + F \times (X_{r3,G} - X_{r4,G})$

$$T(x_{i,d}^{cont}) = \frac{1}{(1+e^{-x_{i,d}^{cont}})} \tag{6}$$

$$x_{i,d}^{bin} = \begin{cases} 0, rand < T(x_{i,d}^{cont}) \\ 1, rand \geq T(x_{i,d}^{cont}) \end{cases} \tag{7}$$

The second approach of binary QUATRE algorithm (BQUATRE2) only converts the mutation matrix B to the binary. The process is shown as follows. The first step is the same as the first one in BQUATRE1, that is, M and $\bar{M}$can be got according to Eq. (5). The mutation matrix B is selected from Table 1. The particle i of B in d dimension can be denoted as $b_{i,d}(i = 1, 2, \cdots, ps; d = 1, 2, \cdots, D)$. The sigmoid function is used on matrix B as shown by Eq. (8).

$$T(b_{i,d}^{cont}) = \frac{1}{(1+e^{-b_{i,d}^{cont}})} \tag{8}$$

The range of $T(b_{i,d}^{cont})$ is [0,1], which can be convert to binary according to Eq. (9).

$$b_{i,d}^{bin} = \begin{cases} 1, rand < T(b_{i,d}^{cont}) \\ 0, rand \geq T(b_{i,d}^{cont}) \end{cases} \tag{9}$$

For the formula $\hat{X} = M \otimes \hat{X} + \bar{M} \otimes B$, M and $\bar{M}$ are binary, and B can be convert to binary through Eq. (8) and Eq. (9). Therefore, $\hat{X}$ is a binary matrix.

Since the transfer function and scale factor F combined to determine the probability of conversion in the proposed binary QUATRE. A linearly increasing scale factor is used in BQUATRE as shown in Eq. (10), where $MAXITER$ is the maximum number of iteration, F_{max} and F_{min} are the predetermined maximum and minimum values of scale factor F. Usually, we set $F_{max} = 2$ and $F_{min} = 0$.

$$F = (F_{\max} - F_{\min}) \times \frac{G}{MAXITER}. \tag{10}$$

3. The Proposed Algorithm

3.1. The Time-varying Transfer Functions

The process of the meta-heuristic algorithms can be divided into two stages: exploration and exploitation. Generally speaking, more attention is paid to exploration at the early stage of iteration, so that more possibilities can be explored and local optimal solutions can be avoided. In the later stage of iteration, more attention is paid to exploitation, the approximate interval of the optimal solution has been found at this time, and the optimal solution needs to be found in this local area. Based on this, a dynamic time-varying transfer function is designed in paper [24] to balance the exploration and exploitation of BPSO.

The slope of the transfer function can be calculated to evaluate the velocity trend, because the slope indicates the switching speed of position. It is indicated that the slope is inverse to the value of $|x|$, while the slope is large when the value of $|x|$ is small [35].

Inspired by paper [24], an effective strategy to mitigate this problem in the BQUATRE is to update the S-shaped TF and design a new model of TFs as shown in Table 2, where τ is a time-varying variable that got by Eq. (11), where itr is the current iteration, itr_{max} is the maximum number of iterations, and φ_{max}, φ_{min} are the bounds on the control parameter φ.

$$\varphi = \varphi_{\max} - itr \times \frac{\varphi_{\max} - \varphi_{\min}}{itr_{\max}} \tag{11}$$

The curve of transfer function S_1 is shown in Figure 1(a), where $\varphi_{min} = 0.1$ and $\varphi_{max} = 10$. The transfer functions used in the algorithm vary from "Initial shapes" to "Final shapes" as the number of iterations increases. It clearly indicated that the curve $S(\varphi_{max})$ linearly increase as the velocity value increase, which is much slower than the other curves. For each given velocity value, the curve $S(\varphi_{max})$ provides the highest amount of bit flipping probability because the curve is the closest to the probability value of 0.5 than any other curves [24]. Therefore, $S(\varphi_{max})$ is beneficial for exploration and should be placed at the early stage of the iteration, which is denoted as 'Initial shapes' in Figure 1(a). On the contrary, $S(\varphi_{min})$ should be placed at the late stage of the iteration, which is denoted as 'Final shapes' in the same figure.

For the V-shaped, U-shaped, and Z-shaped transfer functions, BQUATRE1 uses Eq.(12) instead of Eq.(7) to determine the position of the particles, where $(x_{ik}^{bin})^{-1}$ is the inverse value of x_{ik}^{bin}. That is, if $x_{ik}^{bin} = 0$, then $(x_{ik}^{bin})^{-1} = 1$, and If x_{ik}^{bin}=1, then $(x_{ik}^{bin})^{-1}$=0. Similarly, BQUATRE2 uses Eq.(13) instead of Eq.(9) to determine the b value.

$$x_{i,d}^{bin} = \begin{cases} (x_{i,d}^{bin})^{-1}, rand < T(x_{i,d}^{cont}) \\ x_{i,d}^{bin}, rand \geq T(x_{i,d}^{cont}) \end{cases} \quad \textbf{(12)}$$

$$b_{i,d}^{bin} = \begin{cases} (b_{i,d}^{bin})^{-1}, rand < T(b_{i,d}^{cont}) \\ b_{i,d}^{bin}, rand \geq T(b_{i,d}^{cont}) \end{cases} \quad \textbf{(13)}$$

Figure 1(b) shows the curve of the transfer function V_1, where $\varphi_{min} = 0.6$ and $\varphi_{max} = 10$. For Eq. (12) is used to convert the probability value to binary, the probability of flipping the bits of position increase as the value of $T(x_{i,d}^{cont})$ increase. That is, given the same value of $x_{i,d}^{cont}$, the $V(\varphi_{\min}$) has a greater probability of flipping, which is useful to exploration, while $V(\varphi_{max})$ has a smaller probability of flipping, which is helpful for exploitation. Therefor, a new formula Eq.(14) is designed for computing φ for V-shaped transfer functions for the first time. Then, the V-shaped families of transfer functions are shown in Table 3.

$$\varphi = \varphi_{\min} + itr \times \frac{\varphi_{\max} - \varphi_{\min}}{itr_{\max}} \quad \textbf{(14)}$$

Table 2. The time-varying S-shaped families of transfer functions

Name	Transfer function
$\boldsymbol{S_1(x)}$	$T_1(x) = \frac{1}{(1+e^{-\frac{2x}{\varphi}})}$
$\boldsymbol{S_2(x)}$	$T_2(x) = \frac{1}{(1+e^{-\frac{x}{\varphi}})}$
$\boldsymbol{S_3(x)}$	$T_3(x) = \frac{1}{(1+e^{-x/2\varphi})}$
$\boldsymbol{S_4(x)}$	$T_4(x) = \frac{1}{(1+e^{-x/3\varphi})}$

Table 3. The time-varying V-shaped families of transfer functions

Name	Transfer function
$\boldsymbol{V_1(x)}$	$T_1(x) = \lvert erf(\frac{\sqrt{\pi}x}{2\varphi})\rvert$
$\boldsymbol{V_2(x)}$	$T_2(x) = \lvert tanh(x/\varphi)\rvert$
$\boldsymbol{V_3(x)}$	$T_3(x) = \lvert x/\varphi\sqrt{1+(x/\varphi)^2}\rvert$
$\boldsymbol{V_4(x)}$	$T_4(x) = \lvert\frac{2}{\pi}arctan(\frac{2}{\pi}x/\varphi)\rvert$

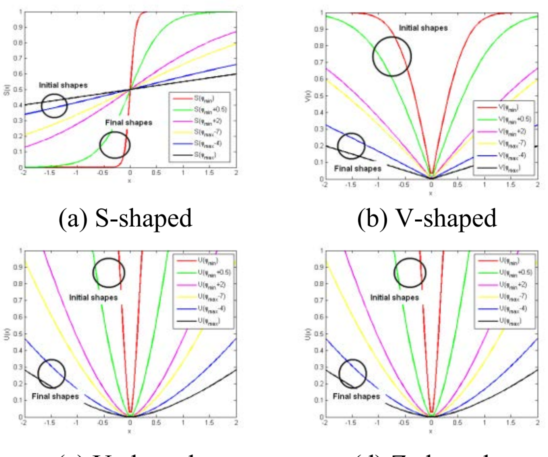
Similar to V-shaped families of transfer functions, the U-shaped and Z-shaped families of transfer functions are described in Table 4 and Table 5. The curves of the time-varying $U_1(\varphi_{min} = 0.1, \varphi_{max} = 10)$ and $Z_1(\varphi_{min} = 0.1, \varphi_{max} = 10)$ are shown in Figure 1(c) and Figure 1(d), respectively. Same as previously mentioned, $U(\varphi_{min})$ and $Z(\varphi_{min})$ are used at the early stage of the iteration while $U(\varphi_{max})$ and $Z(\varphi_{max})$are used at the late stage. Then, Eq. (14) is used for the U-shaped and Z-shaped families of transfer functions.

Table 4. The time-varying U-shaped families of transfer functions

Name	Transfer function
$\boldsymbol{U_1(x)}$	$T_1(x) = min(\lvert(x/\varphi)^{1.5}\rvert,1)$
$\boldsymbol{U_2(x)}$	$T_2(x) = min(\lvert(x/\varphi)^{2}\rvert,1)$
$\boldsymbol{U_3(x)}$	$T_3(x) = min(\lvert(x/\varphi)^{3}\rvert,1)$
$\boldsymbol{U_4(x)}$	$T_4(x) = min(\lvert(x/\varphi)^{4}\rvert,1)$

Table 5. The time-varying Z-shaped families of transfer functions

Name	Transfer function
$\boldsymbol{Z_1(x)}$	$T_1(x) = \sqrt{1-2^{(x/\varphi)}}$
$\boldsymbol{Z_2(x)}$	$T_2(x) = \sqrt{1-5^{(x/\varphi)}}$
$\boldsymbol{Z_3(x)}$	$T_3(x) = \sqrt{1-8^{(x/\varphi)}}$
$\boldsymbol{Z_4(x)}$	$T_4(x) = \sqrt{1-20^{(x/\varphi)}}$



(a) S-shaped (b) V-shaped

(c) U-shaped (d) Z-shaped

Figure 1. The (a), (b), (c), and (d) are the time-varying S-shaped, V-shaped, U-shaped, and Z-shaped transfer functions, respectively

3.2. The Time-varying Binary QUATRE Algorithm -Approach 1

In this section, the first approach of time-varying binary algorithm (BQUATRE1-Tv) that is based on BQUATRE1 will be described in detail. The same as BQUATRE1, BQUATRE1-Tv converts the continuous position matrix at $t+1$ generation $\hat{X}(t+1)$ to the binary.

First of all, we calculate the generate matrix M by Eq. (5), therewith $\bar{M}$ can be computed very easily. Secondly, use Eq. (10) to calculate F, subsequently select a mutation strategy from Table 1, the mutation matrix B can be obtained. Thirdly, calculate φ by Eq. (11) (S-shaped) or Eq. (14) (V-shaped, U-shaped, and Z-shaped)according to the selected transfer function. Then, convert every $x_{i,d}^{cont}(t+1)$ to probability value $T_{tv}(x_{i,d}^{cont}(t+1))$ by the time-varying transfer function $T_{tv}(x)$. Finally, the position matrix $\hat{X}(t+1)$ can be got. The rest of the steps are similar to BQUATRE1.

The pseudo code of BQUATRE1-Tv is described in Table 6.

3.3. The Time-varying Binary QUATRE Algorithm -Approach 2

The second approach of time-varying binary QUATRE algorithm (BQUATRE2-Tv) only converts the mutation matrix B to the binary which is based on BQUATRE2.

Suppose the factor $F=[0,2]$, then $B\in[-2,3]$. We can find that they are not fit the search space $[-2,3]$. The center of search space is 0.5, while the center of four families of time-varying transfer functions is 0. Therefore, we shift the time-varying transfer functions to the center of the search space. That is, all x in Table 5 to Table 8 will be replaced by $(x-0.5)$.

Table 6. Pseudo code of time-varying binary QUATRE algorithm-Approach 1 (BQUATRE1- Tv)

Algorithm 1. Pseudo code of BQUATRE1-Tv

```
input:
The dimension $D$, population size $ps$, max iteration $MAXITER$, the fitness function $f(\hat{X})$,
the time-varying transfer function $T_{tv}(x)$, and the mutation schema to calculate $B$.
Initialization:
Position matrix $\hat{X}(t)$, $\hat{X}_{pbest}(t) = \hat{X}(t)$ , $t = 1$, $\hat{X}_{gbest}(t)$.
Iteration:
1: while $G < MAXITER \| ! stopCriterion$ do
2:   Generate matrix $M$ by Eq. (5), calculate $\bar{M}$.
3:       $F = (F_{max} - F_{min}) \times t / MAXITER$
4:     Calculation continues mutation Matrix $B$ according to mutation schema.
5:       $\hat{X}^{cont}(t+1) = M \otimes \hat{X}^{bin}(t) + \bar{M} \otimes B^{cont}$
6:         Calculate $\varphi$ by Eq. (11) or Eq.(14) according to the selected transfer function
$f(\hat{X})$.
7:     Convert every element $x_{i,d}^{cont}$ to probability value $T_{tv}(x_{i,d}^{cont})$ by the $T_{tv}(x)$.
8:     Convert $T_{tv}(x_{i,d}^{cont})$ to binary $x_{i,d}^{bin}$ by Eq. (9) or Eq. (13) according to
the selected transfer function $f(\hat{X})$.
9:     for $i = 1:ps$ do
10:    if $f(X_i(t+1))$ optimal than $f(X_{pbest,i}(t))$ then
11:          $X_{pbest,i}(t+1) = X_i(t+1)$
12:    else
13:          $X_{pbest,i}(t+1) = X_{pbest,i}(t)$
14:    end if
15: end for
16: $\hat{X}(t+1) = \hat{X}_{pbest}(t+1)$, $X_{gbest}(t+1) = opt\{\hat{X}_{pbest}(t+1)\}$.
17:   Update $\hat{X}_{gbest}(t+1)$ by Eq. (2)
18: $t = t + 1$
19: end while
output:
The global optima $X_{gbest}(t)$, $f(X_{gbest}(t))$.
```

The following is the main step of BQUATRE2-Tv. To begin with, the generate matrix M and its inverse matrix $\bar{M}$ can be got in the same way with BQUATRE1-Tv. After that, the scale factor F can be obtained by Eq. (10). The mutation matrix B is calculated according to a mutation strategy in Table 1. In the next part, calculate φ by Eq. (11) or Eq. (14), convert every $b_{i,d}^{cont}(t+1)$ in B^{cont} to probability value $T_{tv}(b_{i,d}^{cont}(t+1))$ by the time-varying transfer function $T_{tv}(b-0.5)$. Then, the binary mutation matrix B^{bin} can be got by Eq. (9) or Eq. (13). Finally, $\hat{X}^{bin}(t+1) = M \otimes \hat{X}^{bin}(t) + \bar{M} \otimes B^{bin}$is used to calculate the binary position matrix $\hat{X}^{bin}(t+1)$.

The pseudo code of BQUATRE2-Tv is described in Table 7.

Table 7. Pseudo code of time-varying binary QUATRE algorithm-Approach 2 (BQUATRE2- Tv)

Algorithm 2. Pseudo code of BQUATRE2-Tv

```
input:
The dimension $D$, population size $ps$, max iteration $MAXITER$, the fitness function $f(\hat{X})$,
the time-varying transfer function $T_{tv}(x)$, and the mutation schema to calculate $B$ .
Initialization:
Position matrix $\hat{X}(t)$, $\hat{X}_{pbest}(t) = \hat{X}(t)$ , $t = 1$, $\hat{X}_{gbest}(t)$.
Iteration:
1:  while $G < MAXITER | !stopCriterion$ do
2:    Generate matrix $M$ by Eq. (5), calculate $\bar{M}$
3:        $F = (F_{max} - F_{min}) \times t/MAXITER$
4:        Calculation continues mutation Matrix $B^{cont}$ according to mutation schema.
5:        Calculate $\varphi$ by Eq. (11) or Eq. (14) according to the selected transfer function
$f(\hat{X})$.
6:        Convert every element $b_{i,d}^{cont}$ to probability value $T_{tv}(b_{i,d}^{cont})$ by the $T_{tv}(x)$.
7:        Convert $T_{tv}(b_{i,d}^{cont})$ to binary $b_{i,d}^{bin}$ by Eq. (9) or Eq. (13) according to
the selected transfer function $f(\hat{X})$.
8:        $\hat{X}^{bin}(t+1) = M \otimes \hat{X}^{bin}(t) + \bar{M} \otimes B^{bin}$
9:        for $i = 1{:}ps$ do
10:       if $f(X_i(t+1))$ optimal than $f(X_{pbest,i}(t))$ then
11:               $X_{pbest,i}(t+1) = X_i(t+1)$
12:       else
13:               $X_{pbest,i}(t+1) = X_{pbest,i}(t)$
14:       end if
15: end for
16: $\hat{X}(t+1) = \hat{X}_{pbest}(t+1)$, $X_{gbest}(t+1) = opt\{\hat{X}_{pbest}(t+1)\}$.
17:   Update $\hat{X}_{gbest}(t+1)$ by Eq. (2)
18: $t = t + 1$
19: end while
output:
The global optima $X_{gbest}(t)$, $f(X_{gbest}(t))$.
```

4. Experimental Results and Analysis

When we perform various analyses on data, we often need to extract the most representative features from a large number of features to analyze useful information. Feature selection is to select d features from D features, where $d \leq D$, and it is widely used in data mining, image recognition, speech recognition, text classification, and other fields.

In order to verify the effect of feature selection, we use the K nearest neighbor classifier (KNN) to classify the data after feature selection. The KNN classifier is one of the simplest classifiers. The process is to calculate the k training samples closest to the test sample, and then select the category label with the largest vote. That is, the category with the largest number of k closest training samples is used as the category of the test sample. K-fold cross-validation is used in the experiment.

Fitness function is one of the key technologies for applying meta-heuristic algorithms. Generally speaking, there are two fitness functions are used by meta-heuristic algorithms for feature extraction. Eq. (15) and Eq. (16) are the two fitness functions, where $kfoldloss$ is the classification error of k-fold cross-validation, $|S|$ is the number of the selected features and $|C|$ is the number of the total features. Eq. (15) means the evaluation criteria (fitness function) is defined as the average classification error of k-fold cross validation. Eq. (16) means the evaluation criteria is determined by the average classification error and the number of the selected features. The parameter m is the weight of the classification error of k-fold cross-validation.

$$fitness = kfoldloss \quad \textbf{(15)}$$

$$fitness = m \times kfoldloss + (1 - m) \times |S| / |C| \quad \textbf{(16)}$$

In this section, we test the proposed algorithm on feature selection with UCI repository. UCI is a repository for machine learning which is proposed by the University of California Irvine. Nine datasets of UCI are chosen in this experiment to validate the performance of the proposed time-varying binary QUATRE algorithm. Table 8 described the nine datasets in detail. The “N/A” in Table 8 means “Not Available”.

We choose the second proposed approach BQUATE2-Tv in this experiment and the four families of transfer functions are used. Multiple Dimensional Scaling (MDS), Principal component analysis (PCA), Binary Gray Wolf Optimization (BGWO) [36], and BQUATRE2 are used as the compared algorithms. MDS and PCA are both classical methods for feature extraction [37]. The number of features selected using PCA is got by the ’drtoolbox’ that is a famous dimension reduction toolbox on MATLAB.

Table 8. The datasets of UCI repository

No.	Dataset	Attribute Types	Instances	Attributes	Year
1	Wine	Integer, Real	178	13	1991
2	Soybean	Categorical	47	35	1987
3	Connectionist Bench	Real	208	60	N/A
4	Parkinsons	Real	197	23	2008
5	Molecular Biology	Categorical	106	58	1990
6	Libras Movement	Real	360	91	2009
7	Statlog	Categorical, Real	270	13	N/A
8	Glass Identification	Real	214	10	1987
9	Breast Cancer	Categorical	286	9	1988
10	Seeds	Clustering,Real	210	7	2012
11	Liver Disorders	Categorical, Integer, Real	345	7	1990
12	Zoo	Categorical, Integer	101	17	1990
13	Image Segmentation	Real	2310	19	1990
14	Ceramic Samples	Real	88	19	2019

Table 9. The results of the compared algorithms base on Eq. (15)

Dataset	BGWO		MDS		PCA		BQUATRE2		BQUATE2-Tv(S1)		BQUATE2-Tv(V1)		BQUATE2-Tv(U1)		BQUATE2-Tv(Z1)	
	Error	Number	Error	Number	Error	Number	Error	Number	Error	Number	Error	Number	Error	Number	Error	Number
1	0.0618	0.6944	0.2371	0.3077	0.2247	0.3333	0.2438	0.3667	0.2146	0.4000	0.0933	0.5000	0.3663	0.4833	0.2910	0.6333
2	0.2536	0.7810	0.2348	0.0833	0.4348	0.0857	0.3348	0.3657	0.3130	0.4800	0.1696	0.5200	0.3000	0.5143	0.1957	0.5657
3	0.2628	0.8389	0.3327	0.1311	0.3125	0.1333	0.2548	0.3000	0.2442	0.3600	0.2808	0.4400	0.2538	0.4967	0.3000	0.4733
4	0.1460	0.5000	0.2031	0.1304	0.1856	0.1364	0.2000	0.2727	0.1825	0.5091	0.1722	0.5636	0.1753	0.5455	0.1825	0.4364
5	0.3098	0.6667	0.2962	0.3793	0.3585	0.3860	0.2642	0.4737	0.2736	0.5053	0.2943	0.5263	0.2453	0.5439	0.1830	0.6035
6	0.4687	0.7640	0.5783	0.0444	0.5694	0.0449	0.4522	0.3618	0.4622	0.4697	0.4533	0.4584	0.4678	0.5438	0.4361	0.5438
7	0.1864	0.6111	0.3600	0.2308	0.3259	0.2500	0.2496	0.3333	0.3748	0.4000	0.3163	0.5833	0.3163	0.5500	0.3200	0.6500
8	0.3536	0.5926	0.3944	0.4000	0.3804	0.4444	0.4290	0.2889	0.3935	0.4889	0.3514	0.4444	0.3617	0.6889	0.3262	0.6444
9	0.5000	0.6667	0.5710	0.4000	0.5913	0.4000	0.5746	0.3333	0.6174	0.4444	0.5891	0.4889	0.5486	0.5333	0.5290	0.6000
10	0.0873	0.7143	0.1067	0.2500	0.1286	0.1250	0.0667	0.4286	0.0714	0.4286	0.0810	0.4286	0.0524	0.4286	0.0714	0.4762
11	0.3241	0.5556	0.5767	0.7143	0.5767	0.7143	0.2945	0.5000	0.3190	0.4444	0.2914	0.5000	0.3006	0.4444	0.3190	0.5000
12	0.1533	0.8750	0.2480	0.5882	0.4629	0.1053	0.1033	0.7500	0.0933	0.6042	0.0900	0.6667	0.0900	0.5833	0.0833	0.6250
13	0.2000	0.7778	0.4476	0.1053	0.4914	0.1053	0.2000	0.5741	0.2000	0.5741	0.1810	0.6481	0.1667	0.6111	0.2000	0.6296
14	0.5341	0.5741	0.7023	0.1053	0.6886	0.1053	0.5114	0.6556	0.5000	0.5185	0.5227	0.4444	0.5000	0.5370	0.5227	0.5926
Compared with PCA(W|T|L)	14|0|0		4|1|9		------		11|0|3		11|0|3		14|0|0		13|0|1		13|0|1	
Compared with MDS(W|T|L)	12|0|2		------		9|1|4		10|0|4		11|0|3		13|0|1		12|0|2		13|0|1	
Compared with BQUATRE2(W|T|L)	6|0|8		4|0|10		3|0|11		------		7|1|6		7|0|7		10|0|4		7|0|7	
Compared with BGWO(W|T|L)	------		2|0|12		0|0|14		8|0|6		8|0|6		9|0|5		8|0|6		9|0|5	

Table 10. The results of the compared algorithms base on Eq. (16)

Dataset	BGWO		MDS		PCA		BQUATRE2		BQUATE2-Tv(S1)		BQUATE2-Tv(V1)		BQUATE2-Tv(U1)		BQUATE2-Tv(Z1)	
	Error	Number	Error	Number	Error	Number	Error	Number	Error	Number	Error	Number	Error	Number	Error	Number
1	0.0573	0.6667	0.2371	0.3077	0.2247	0.3333	0.2281	0.2500	0.2764	0.4500	0.1910	0.4667	0.1730	0.3500	0.1157	0.5500
2	0.3261	0.6800	0.2348	0.0833	0.4348	0.0857	0.3000	0.3029	0.2522	0.4800	0.2130	0.3943	0.2826	0.3829	0.1000	0.4514
3	0.2760	0.8033	0.3327	0.1311	0.3125	0.1333	0.2981	0.3667	0.3144	0.4500	0.2510	0.3600	0.2750	0.3733	0.2712	0.4333
4	0.1361	0.6091	0.2031	0.1304	0.1856	0.1364	0.1722	0.2545	0.1876	0.6091	0.1784	0.4364	0.1557	0.3727	0.1990	0.4364
5	0.2170	0.8772	0.2962	0.3793	0.3585	0.3860	0.2962	0.3333	0.2528	0.4842	0.2755	0.4386	0.2736	0.5368	0.2038	0.5579
6	0.4856	0.7730	0.5783	0.0444	0.5694	0.0449	0.4556	0.3528	0.4289	0.4180	0.4406	0.4517	0.4561	0.4629	0.4311	0.4607
7	0.1763	0.5667	0.3600	0.2308	0.3259	0.2500	0.3193	0.2833	0.3148	0.5500	0.3044	0.4000	0.2867	0.6167	0.3267	0.7000
8	0.3421	0.6889	0.3944	0.4000	0.3804	0.4444	0.4280	0.3778	0.3570	0.4000	0.4084	0.4444	0.3841	0.5333	0.3542	0.7778
9	0.5058	0.6444	0.5710	0.4000	0.5913	0.4000	0.6319	0.3556	0.5913	0.5556	0.6246	0.4444	0.6058	0.4667	0.5101	0.6222
10	0.0867	0.4571	0.1067	0.2500	0.1286	0.1250	0.0778	0.3333	0.0667	0.4286	0.0762	0.2857	0.0714	0.2857	0.0667	0.4286
11	0.3190	0.5000	0.5767	0.7143	0.5767	0.7143	0.3190	0.3333	0.3129	0.3333	0.3160	0.8333	0.3037	0.5000	0.3252	0.3333
12	0.1220	0.5750	0.2480	0.5882	0.4629	0.1053	0.1160	0.7125	0.0800	0.5625	0.1000	0.8250	0.0800	0.5125	0.0800	0.5250
13	0.2095	0.7778	0.4476	0.1053	0.4914	0.1053	0.1905	0.4444	0.1714	0.5000	0.1714	0.4556	0.1714	0.4556	0.2000	0.5444
14	0.5758	0.5556	0.7023	0.1053	0.6886	0.1053	0.5227	0.2407	0.5114	0.2778	0.4841	0.2111	0.5000	0.3333	0.4773	0.1667
Compared with PCA(W|T|L)	14|0|0		4|1|9		------		11|0|3		10|0|4		12|0|2		12|0|2		12|0|2	
Compared with MDS(W|T|L)	13|0|1		------		9|1|4		11|0|3		11|0|3		12|0|2		12|0|2		14|0|0	
Compared with BQUATRE2(W|T|L)	7|0|7		3|0|11		3|0|11		------		11|0|3		13|0|1		13|0|1		10|0|4	
Compared with BGWO(W|T|L)	------		1|0|13		0|0|14		7|0|7		7|0|7		8|0|6		8|0|6		8|0|6	

The fitness function (15) is used in the first experiment and the result is shown in Table 8. The MDS, PCA, BGWO, and BQUATRE2 algorithms are compared with the four time-varying binary QUATRE algorithms, respectively. The fourth from last row in Table 9 is the result compared with the PCA algorithm. The third line from last is the result compared with the MDS algorithm. It can be got that all four proposed time-varying binary QUATRE algorithms are better than PCA and MDS. In detail, BQUATE2-Tv (S1), BQUATE2-Tv (V1), BQUATE2-Tv(U1), and BQUATE2-Tv (Z1) obtain better results on 11, 14, 13, and 13 datasets than PCA, and got better results on 11, 13, 12, and 13 datasets than MDS, respectively. This means that the proposed algorithm is better than the classical algorithm. The penultimate line is the result compared with the BQUATRE2 algorithm and the last line is the results compared with BGWO. They obtain better results on 7, 7, 10, and 7 datasets than BQUATRE2, and got 8, 9, 8, and 9 better results than BGWO. Therefor, the proposed algorithm is superior to the state-of-the-art methods. For the four time-varying transfer functions, the V-shaped and U-shaped functions obtain better performance for fitness function (15).

In the second experiment, we use Eq. (16) as the fitness function, and set m = 0.99 because many researchers have adopted this value in their works and get good performance. Table 10 listed the results in detail. The algorithms used in this experiment are the same as the last one.

The same to Table 9, the fourth from last row in Table 9 is the result compared with the PCA algorithm. BQUATE2-Tv (S1), BQUATE2-Tv (V1), BQUATE2-Tv (U1), and BQUATE2-Tv (Z1) obtained 10, 12, 12, and 12 better results than PCA, respectively. Similarly, they are superior to MDS. The penultimate line is the results compared with BQUATRE2. The experimental results are shown that BQUATE2-Tv (S1), BQUATE2-Tv (V1), BQUATE2-Tv (U1), and BQUATE2-Tv (Z1) got better results on 11, 13, 13, and 10 datasets. It can be seen from the last line that the proposed algorithm with four time-varying transfer functions got better results on 7, 8, 8, and 8 datasets than BGWO. Therefor, the proposed algorithm is superior to the state-of-the-art methods based on the fitness function (16).

To further compare the results, t-test is used for a significance test. It is suitable for a normal distribution with a small sample size and unknown population standard deviation. In the experiments, a two tailed t-test is used with significant level of 0.05, which means significant. The results are shown in Table 11, where "+" implies that the compared algorithm is superior to BQUATE2, and "–" indicates that the algorithm is inferior to BQUATE2. "=" means that the performance of the algorithms is consistent.

It can be seen from Table 11, The method that performed best was BQUATE2-Tv(U1), which got 12 "+". It was followed by BQUATE2-Tv (U1), which got 10 "+". Then, BQUATE2-Tv (Z1) got 9 "+" and 1 "-". MDS and PCA performed worst in this experiment.

Table 11. Thet-test results of the compared algorithms on BQUATE2 base on Eq. (16)

Dataset	BGWO	MDS	PCA	BQUATE2-Tv(S1)	BQUATE2-Tv(V1)	BQUATE2-Tv(U1)	BQUATE2-Tv(Z1)
1	+	=	=	-	+	+	+
2	-	-	+	+	+	+	+
3	+	-	-	-	+	+	+
4	+	-	-	-	=	+	-
5	+	-	=	+	+	+	+
6	-	-	-	+	+	=	+
7	+	=	-	=	+	+	=
8	+	+	+	+	+	+	+
9	+	+	+	+	=	+	+
10	=	-	-	=	=	=	=
11	=	-	-	=	=	+	=
12	=	-	-	+	+	+	+
13	-	-	-	+	+	+	=
14	-	-	-	=	+	+	+

5 Conclusion

In this paper, four families of transfer functions are converted to time-varying form to balance exploration and exploitation. Furthermore, two new approaches of binary QUATRE algorithm using the time-varying transfer functions are proposed. In order to test the performance of the proposed algorithm, the experiments are performed on feature selection problem which plays an important role in many areas, including data mining, image recognition, speech recognition, text classification, and so on. The results of the experiments on the UCI repository indicated that the proposed algorithms are superior to the compared PCA, MDS, and the original BQUATRE algorithms. The time-varying transfer function can be designed in other forms. In addition, the transfer function can also be designed to vary with other factors. This will be our future work.

References

[1] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, *Swarm intelligence*, Vol. 1, No. 1, pp. 33-57, June, 2007.

[2] J. Wang, J. Liu, J.-S. Pan, X. Xue, L. Huang, A hybrid bpso-ga algorithm for 0-1 knapsack problems, *The Euro-China Conference on Intelligent Data Analysis and Applications*, Bulevar Louis Pasteur, Spain, 2017, pp. 344-351.

[3] B. Du, J. Zhu, Q. Ding, Optimization of multi-scale kernel chaotic time series prediction method based on the joint parameters were optimized with variable particle swarm, *Journal of Network Intelligence*, Vol. 3, No. 4, pp. 291-304, December, 2018.

[4] J. Tian, Y. Tan, J. Zeng, C. Sun, Y. Jin, Multiobjective infill criterion driven gaussian process-assisted particle swarm optimization of high-dimensional expensive problems, *IEEE Transactions on Evolutionary Computation*, Vol. 23, No. 3, pp. 459-472, June, 2019.

[5] S.-C. Chu, P.-W. Tsai, J.-S. Pan, Cat swarm optimization, *Pacific Rim international conference on artificial intelligence*, Guilin, China, 2006, pp. 854- 858.

[6] Y. Kumar, P. K. Singh, Improved cat swarm optimization algorithm for solving global optimization problems and its application to clustering, *Applied Intelligence*, Vol. 48, No. 9, pp. 2681-2697, September, 2018.

[7] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, *Knowledge-based systems*, Vol. 89, pp. 228-249, November, 2015.

[8] S. Mirjalili, Sca: a sine cosine algorithm for solving optimization problems, *Knowledge-based systems*, Vol. 96, pp. 120-133, March, 2016.

[9] M. A. Elaziz, D. Oliva, S. Xiong, An improved opposition-based sine cosine algorithm for global optimization, *Expert Systems with Applications*, Vol. 90, pp. 484-500, December, 2017.

[10] H. Duan, P. Qiao, Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning, *International journal of intelligent computing and cybernetics*, Vol. 7, No. 1, pp. 24-37, March, 2014.

[11] T.-T. Nguyen, J.-S. Pan, T.-K. Dao, T.-W. Sung, T.-G. Ngo, Pigeon-inspired optimization for node location in wireless sensor network, *International Conference on Engineering Research and Applications*, Thai Nguyen, Vietnam, 2019, pp. 589-598.

[12] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, *Neural Computing and Applications*, Vol. 27, No. 4, pp. 1053-1073, May, 2016.

[13] M. Mafarja, I. Aljarah, A. A. Heidari, H. Faris, P. Fournier-Viger, X. Li, S. Mirjalili, Binary dragonfly optimization for feature selection using time-varying transfer functions, *Knowledge-Based Systems*, Vol. 161, pp. 185-204, December, 2018.

[14] J.-S. Pan, P.-W. Tsai, Y.-B. Liao, Fish migration optimization based on the fishy biology, *2010 Fourth International Conference on Genetic and Evolutionary Computing*, Shenzhen, China, 2010, pp. 783-786.

[15] 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 terrain, *Journal of Information Hiding and*

Multimedia Signal Processing, Vol. 11, No. 2, pp. 90-102, June, 2020.
[16] X.-S. Yang, M. Karamanoglu, X. He, Flower pollination algorithm: a novel approach for multi-objective optimization, *Engineering optimization*, Vol. 46, No. 9, pp. 1222-1237, 2014.
[17] 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, *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 8, No. 2, pp. 486-499, March, 2017.
[18] J.-S. Pan, Z. Meng, H. Xu, X. Li, Quasi-affine transformation evolution (quatre) algorithm: A new simple and accurate structure for global optimization, *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, Morioka, Japan, 2016, pp. 657-667.
[19] S.-C. Chu, Z.-J. Zhuang, J.-B. Li, J.-S. Pan. A Novel Binary QUasi-Affine TRansformation Evolutionary (QUATRE) Algorithm. *Applied Sciences*, Vol. 11, No. 5, 2251, March, 2021.
[20] Z. Meng, Y. Chen, X. Li, C. Yang, Y. Zhong, Enhancing quasi-affine transformation evolution (quatre) with adaptation scheme on numerical opti- mization, *Knowledge-Based Systems*, Vol. 197, Article No. 105908, June, 2020.
[21] J.-S. Pan, F. Fan, S.-C. Chu, Z. Du, H. Zhao, A node location method in wireless sensor networks based on a hybrid optimization algorithm, *Wireless Communications and Mobile Computing*, Vol. 2020, Article No. 8822651, October, 2020.
[22] N. Liu, J.-S. Pan, C. Sun, S.-C. Chu, An efficient surrogate-assisted quasi-affine transformation evolutionary algorithm for expensive optimization problems, *Knowledge-Based Systems*, Vol. 209, Article No. 106418, December, 2020.
[23] N. Liu, J.-S. Pan, J. Y. Xue, An orthogonal quasi-affine transformation evolution (o-quatre) algorithm for global optimization, *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, Jilin, China, 2020, pp. 57-66.
[24] M. J. Islam, X. Li, Y. Mei, A time-varying transfer function for balancing the exploration and exploitation ability of a binary pso, *Applied Soft Computing*, Vol. 59, pp. 182-196, October, 2017.
[25] Z. Beheshti, A time-varying mirrored s-shaped transfer function for binary particle swarm optimization, *Information Sciences*, Vol. 512, pp. 1503-1542, February, 2020.
[26] M. A. Kahya, S. A. Altamir, Z. Y. Algamal, Improving whale optimization algorithm for feature selection with a time-varying transfer function, *Numerical Algebra, Control and Optimization*, Vol. 11, No. 1, pp. 87- 98, March, 2021.
[27] A. T. Sahlol, D. Yousri, A. A. Ewees, M. A. A. Al-qaness, R. Damasevicius, M. A. Elaziz, COVID-19 image classification using deep features and fractional-order marine predators algorithm, *Scientific reports*, Vol. 10, No. 1, pp. 1-15, September, 2020.
[28] S. Sen, S. Saha, S. Chatterjee, S. Mirjalili, R. Sarkar, A bi-stage feature selection approach for COVID-19 prediction using chest CT images, *Applied Intelligence*, Vol. 51, No. 12, pp. 8985-9000, December, 2021.
[29] S. Mirjalili, H. Zhang, S. Mirjalili, S. Chalup, N. Noman, A Novel U-Shaped Transfer Function for Binary Particle Swarm Optimisation, *9th International Conference on Soft Computing for Problem Solving*, Liverpool, UK, 2019, pp. 241-259.
[30] S.-S. Guo, J.-S. Wang, M.-W. Guo, Z-shaped transfer functions for binary particle swarm optimization algorithm, *Computational Intelligence and Neuroscience*, Vol. 2020, Article No. 6502807, June, 2020.
[31] X.-D. Wang, R.-C. Chen, F. Yan, High-dimensional data clustering using k-means subspace feature selection, *Journal of Network Intelligence*, Vol. 4, No. 3, pp. 80-87, August, 2019.
[32] L.-C. Ungureanu, T. Hartmann, Inside the collective mind: Features extraction to support automated design space explorations, in: I. Mutis, T. Hartmann (Eds.), *Advances in Informatics and Computing in Civil and Construction Engineering*, Springer, 2019, pp. 199-205.
[33] G. Fu, C. Sun, Y. Tan, G. Zhang, Y. Jin, A surrogate-assisted evolutionary algorithm with random feature selection for large-scale expensive problems, *International Conference on Parallel Problem Solving from Nature*, Leiden, The Netherlands, 2020, pp. 125-139.
[34] K.-W. Huang, C.-C. Lin, Y.-M. Lee, Z.-X. Wu, A deep learning and image recognition system for image recognition, *Data Science and Pattern Recognition*, Vol. 3, No. 2, pp. 1-11, May, 2019.
[35] P. Hu, J.-S. Pan, S.-C. Chu, Improved binary grey wolf optimizer and its application for feature selection, *Knowledge-Based Systems*, Vol. 195, Article No. 105746, May, 2020.
[36] L. K. Panwar, S. Reddy, A. Verma, B. K. Panigrahi, R. Kumar, Binary grey wolf optimizer for large scale unit commitment problem, *Swarm and Evolutionary Computation*, Vol. 38, pp. 251-266, February, 2018.
[37] F. Siddiqui, P. Sargent, G. Montague, The use of pca and signal processing techniques for processing time-based construction settlement data of road embankments, *Advanced Engineering Informatics*, Vol. 46, Article No. 101181, October, 2020.

Biographies

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.

Chia-Cheng Hu received the M.S. degree in Engineer Science from National Cheng Kung University in 1995. He received his Ph.D. in department of Computer Science and Information Engineering, National Taiwan University, Taiwan in 2005. He is currently a Professor with the College of Artificial Intelligence, Yango University.

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.