Article

A Novel Binary QUasi-Affine TRansformation Evolutionary (QUATRE) Algorithm

Shu-Chuan Chu ^{1,2} , Zhongjie Zhuang ¹ , Junbao Li ³ and Jeng-Shyang Pan ^{1,4,*}

¹ College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; jan.chu@flinders.edu.au (S.-C.C.); zjzhuangsdust@gmail.com (Z.Z.)

² College of Science and Engineering, Flinders University, 1284 South Road, Clovelly Park, SA 5042, Australia

³ School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China; lijunbao@hit.edu.cn

⁴ Department of Information Management, Chaoyang University of Technology, Taichung 41349, Taiwan

* Correspondence: jspace@cc.kuas.edu.tw

Abstract: QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm generalized differential evolution (DE) algorithm to matrix form. QUATRE was originally designed for a continuous search space, but many practical applications are binary optimization problems. Therefore, we designed a novel binary version of QUATRE. The proposed binary algorithm is implemented using two different approaches. In the first approach, the new individuals produced by mutation and crossover operation are binarized. In the second approach, binarization is done after mutation, then cross operation with other individuals is performed. Transfer functions are critical to binarization, so four families of transfer functions are introduced for the proposed algorithm. Then, the analysis is performed and an improved transfer function is proposed. Furthermore, in order to balance exploration and exploitation, a new liner increment scale factor is proposed. Experiments on 23 benchmark functions show that the proposed two approaches are superior to state-of-the-art algorithms. Moreover, we applied it for dimensionality reduction of hyperspectral image (HSI) in order to test the ability of the proposed algorithm to solve practical problems. The experimental results on HSI imply that the proposed methods are better than Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

Keywords: binary; QUATRE; transfer function; dimension reduction; hyperspectral image

Citation: Chu, S.-C.; Zhuang, Z.; Li, J.; Pan, J.-S. A Novel Binary QUasi-Affine TRansformation Evolutionary (QUATRE) Algorithm. *Appl. Sci.* **2021**, *11*, 2251. <https://doi.org/10.3390/app11052251>

Academic Editor: Giancarlo Mauri

Received: 20 January 2021

Accepted: 26 February 2021

Published: 4 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The optimization problem refers to determine the value of decision variables under special constraints so that the objective functions can reach the optimal values. Traditional methods to solve the optimization problems include simplex method, steepest descent method, trust region method, penalty function method, etc. However, traditional optimization algorithms usually need to satisfy some specific conditions. For example, some algorithms require that the objective function must be continuous or differentiable, some require the problem to be solved is a convex optimization problem, and others require the constraint conditions to meet linear conditions. However, it is sometimes difficult to satisfy these conditions in practical applications. Therefore, in the 1940s, heuristic algorithms emerged, which constructed a model highly similar to the problem to be solved based on intuition or experience, and gave a feasible solution to the optimization problem at an acceptable cost. Furthermore, the deviation degree from the obtained feasible solution to the optimal solution could not be predicted. Heuristic algorithms reduce computational complexity at the expense of computational accuracy. In the 1960s, under the inspiration of bionics, meta-heuristic algorithms appeared, which obtained inspiration from random phenomena in nature. Meta-heuristic algorithms combined random algorithms with local

algorithms in order to jump out of the local optimal solution with a probability. Meta-heuristic algorithms have become one of the hottest spots in the optimization problems, for there is not any special condition that constrains the objective functions, whereas satisfactory solutions can often be obtained.

According to the mechanisms to be based on, meta-heuristic algorithms can be coarsely divided into two categories: one is inspired by biological processes, and the other is inspired by physical or mathematical models. Among them, meta-heuristic algorithms inspired by biological processes can be finely separated into swarm intelligence algorithms, which are based on animal social behaviors, and evolutionary algorithms, which are based on the theory of evolution. In the meta-heuristic algorithms inspired by physical or mathematical models, Simulated Annealing (SA) [1–3] was inspired by the internal molecular state and internal energy of solids in the process of changing from high temperature to low temperature. It is easy to compute but is hard to converge; Gravitational Search Algorithm (GSA) [4–6] was proposed based on Newtonian gravity and the laws of motion, and it has strong exploitation ability but weak exploration ability; Sine Cosine Algorithm (SCA) [7–9] created multiple initial random candidate solutions and then used a mathematical model based on sine and cosine functions to make these solutions fluctuate in the direction of the optimal solution or in the opposite direction. In Swarm intelligence algorithms, Particle Swarm Optimization (PSO) [10–13] simulated the foraging behaviour of birds to obtain the optimal solution. In particular, it was the first swarm intelligence algorithm; Ant Colony Optimization (ACO) [14–16] was inspired by the foraging behavior of ant colony. The parameter setting of ACO is complicated. If the parameter setting is improper, it is easy to deviate from the high-quality solution; Cat Swarm Optimization (CSO) [17,18] mimicked cats hunting behaviour to obtain the optimal solution; Bat Algorithm (BA) [19–21] was proposed based on the echolocation behavior of bats; Pigeon Inspired Optimization (PIO) [22,23] was designed by mimicking the homing behavior of the pigeon. It requires very few adjustment parameters and is easy to implement; Symbiotic Organism Search (SOS) [24,25] imitated the interactive population relationship between different organisms in nature to enhance their adaptability to the environment, so as to improve the survival ability of the population; Grey Wolf Optimizer (GWO) [26,27] simulated the hierarchy and predation behavior of wolves. However, it is easy to fall into local extremum and hard to converge; Cuckoo Search (CS) [28–31] mimicked brood parasitism of cuckoo and it has a strong exploration ability; Monkey King Evolutionary (MKE) [32,33] Algorithm was inspired by the action of the Monkey King, a character of a famous Chinese mythological novel, named Journey to the West. In evolutionary algorithms, Genetic Algorithm (GA) [34–37] was proposed based on the theory of natural selection and the principle of genetics, and the optimal solution was obtained by mutation, crossover and selection; DE [38–41] was proposed to improve GA. The difference was that DE generated the mutation vector by the difference vector of the parent generation, and performed crossover with the individual vector of the parent generation to generate the new individual vector.

In particular, QUATRE was proposed by Khanesar, M.A. [42] to improve the drawback of DE that did not achieve equilibrium search in search space without prior knowledge. QUATRE generalized the crossover operation of DE from vector to matrix. Therefore, DE can be regarded as a special case of QUATRE. Subsequently, a series of algorithms based on QUATRE was proposed due to the good performance of QUATRE. The relations between QUATRE and other meta-heuristic algorithms are discussed by Meng, Z. [43], including PSO variants and DE variants and a framework of gesture recognition was proposed based on QUATRE also by Meng, Z. [44]. A compact QUATRE algorithm that used a pairwise competition mechanism to enhance the performance of QUATRE was proposed [45] and an enhanced mutation strategy with time stamp mechanism for the QUATRE algorithm was shown by Meng, Z. and Pan, J.S. [46]. The reason why the exploration bias still exists in the binomial crossover was discussed [47]; then, a novel QUATRE structure with some novel standards and adaptation schemes was proposed. In addition, a novel bi-population QUasi-Affine TRansformation Evolution algorithm (BP-QUATRE) algorithm was proposed, which

divided the populations into two subpopulations with different sorting and mutation strategies in each subpopulation [48] and a multi-group multi-choice communication strategy was proposed to solve the disadvantage that the original QUATRE was always easy to fall into local optimization and apply the proposed algorithm in wireless sensor network node localization [49]. A novel algorithm that combined QUATRE with PIO was proposed to avoid falling into local optimum by Sun, X.X. [50].

Practical optimization problems are various, some are high dimensional, some are many-objective optimization problems and others require minimal memory. Consequently, many excellent algorithms were proposed to apply the meta-heuristic algorithms for solving practical problems. Furthermore, a new surrogate-assisted model on PSO to deal with the high dimensional problem is used [12]; S. Qin et al. proposed a modified PSO based on a decomposition framework with different ideal points on each reference vector to solve many-objective optimization problems [13]; P.-C. Song et al. developed a compact scheme on CSO to effectively save the memory of the unmanned robot [30]. There are some other practical problems that are binary optimization problems, such as feature extraction and 0–1 knapsack problems, but most meta-heuristic algorithms are designed for a continuous search space. Therefore, it is essential to convert meta-heuristic algorithms in a binary version.

The continuous PSO was converted into binary version for the first time by Khanesar, M.A. [51]. A binary version of the CS was proposed by Rodrigues by Rodrigues, D. [52]. In Reference [53] a hyper learning binary Dragonfly Algorithm (DA) was designed for coronavirus disease 2019 (COVID-19). In Reference [54], an improved binary GWO was discussed and used for feature selection by Hu, P. A optimized binary BA was presented and applied for classification of white blood cells by Gupta, D. [55]. The GSA was converted into binary form based on mutual information by Bostani, H. [56] for intrusion detection systems. Almost all excellent meta-heuristic algorithms have variants of the binary version. However, QUATRE was designed for solving continuous optimization problems and no one has yet turned it into a binary version. Therefore, we discussed how to change the QUATRE algorithm into binary form in this manuscript for the first time.

Remote sensing is an important technology, which is widely used in the military and by civilians [57,58]. On the military side, sensing technology is widely used in military reconnaissance, missile early warning, military surveying and mapping, marine surveillance, and meteorological observation. On the civil side, remote sensing technology is widely used in earth resources survey, vegetation classification, crop pests, crop diseases and crop yield surveys, environmental pollution monitoring, marine development, and earthquake monitoring. At present, high resolution remote sensing has become the mainstream of remote sensing [59]. Hyperspectral remote sensing is an important aspect of high resolution remote sensing. It is a technique to obtain approximately continuous spectral data by imaging spectrometer within the range of visible, near-infrared, mid-infrared and thermal infrared bands of electromagnetic spectrum. Hyperspectral remote sensing can obtain hundreds of spectral bands at the same time, and the wavelength range of each spectral band is only about 10 nm. This many bands can provide extremely rich spectral information for ground object information extraction, so hyperspectral images are often used for fine object classification. Nevertheless, the phenomenon of "Hughes" is also introduced into many dimensions [60,61], that is, with a certain number of samples, the classification accuracy increases first and then decreases with the increase in data dimension. Hughes phenomenon has seriously affected the application and popularization of hyperspectral remote sensing technology. Therefore, dimensionality reduction is often needed before a fine classification.

The main contributions of this paper are concluded as follows:

- (1) Two approaches are presented to convert QUATRE to binary version and four families of transfer functions are used by the two approaches, respectively.
- (2) The search space of the proposed binary QUATRE is analyzed, then new transfer functions are introduced.

- (3) A new scaling factor is proposed in order to balance exploration and exploitation.
- (4) The proposed algorithm is performed on 23 benchmark functions and HSI.
- (5) A new fitness function for dimensionality reduction of HSI is proposed.

The rest of the paper is organized as follows. Section 2 describes related works include QUATRE and transfer functions. Section 3 describes the proposed two binary QUATRE approaches. Section 4 is experimental results and analysis on benchmark test functions and HSI. Section 5 depicts the main work of the paper and gives some suggestions for further work.

2. Related Works

2.1. QUATRE

The algorithm is named QUATRE because its evolution equation is in an affine-transformation like form. Affine transformation is the transformation of a vector space into another vector space by a linear transformation coupled with a translation in geometry. It can be written as $X = MX + B$. Set $X = [x_1, x_2, \dots, x_{ps}]^T$ denotes position matrix of the particle population, where ps is the population size. A particle can be written as $x_i = [x_{i1}, x_{i2}, \dots, x_{iD}]$, where $i = 1, 2, \dots, ps$, and D is the dimension of problem. The exact evolution equation used in QUATRE is shown in Equation (1). Where $\otimes$ means component-wise multiplication, same as the $.*$ operation in Matlab. X_G , X_{G+1} are the position matrixes at G th and $(G + 1)$ th generations.

$$X_{G+1} = M \otimes X_G + \bar{M} \otimes B \quad (1)$$

Matrix B denotes the mutation matrix of particles, which can be generated in six schemes shown in Table 1. Among them, $F \in (0, \infty)$ is a scaling factor but, usually, we constrain $F \in [0, 2]$. F means variation rate and the value of F determines the ability of exploration and exploitation. $X_{r1,G}$, $X_{r2,G}$, $X_{r3,G}$, $X_{r4,G}$ and $X_{r5,G}$ are random matrixes generated by permutating row vectors of position matrix X_G . $X_{gbest,G}$ is defined as Equation (2), where $x_{gbest,G}$ is a vector denotes the global best particle in G th generation. As shown in Table 1, the first mutation strategy is $B = X_{gbest,G} + F * (X_{r1,G} - X_{r2,G})$ which can be denoted as QUATRE/best/1, meaning that the vector to be perturbed is the global best solution $X_{gbest,G}$, and that only one difference vector ($X_{r1,G} - X_{r2,G}$) is included.

$$X_{gbest,G} = \begin{bmatrix} x_{gbest,G} \\ x_{gbest,G} \\ \dots \\ x_{gbest,G} \end{bmatrix} \quad (2)$$

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

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

M can be considered as the selection matrix, made up of 0 and 1. $\bar{M}$ means a binary inverse operation on the matrix elements of M . As shown in Equation (3), binary inversion

means that the element 0 inverts the contravariant to 1, element 1 inverts the contravariant to 0.

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \overline{M} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (3)$$

The initial value of the M is set to the lower triangular matrix, shown as M_{tmp} in Equation (4). Then, randomly swap each row and each column of M_{tmp} , we can get M . In this way, QUATRE can achieve equilibrium search in the search space.

$$M_{tmp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} = M \quad (4)$$

The number of rows in M is the population size ps , while the number of columns in M is the dimension D of problem. If $ps > D$, we can extend M_{tmp} according to ps . As shown in Equation (5), if $ps = k \times D + i$, then the first $k \times D$ rows of M_{tmp} are made up of k lower triangular matrices of $D \times D$. The last i rows of M_{tmp} are the first i rows of $D \times D$ matrix. Then, we can get M by randomly swap each row and each column. If $ps < D$, we can do similar operations to extend the matrix M_{tmp} according to D .

$$M_{tmp} = \begin{bmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ 1 & 1 & 1 & & & \\ & & & & & \\ 1 & 1 & 1 & & & 1 \\ 1 & & & & & \\ 1 & 1 & & & & \\ 1 & 1 & 1 & & & \\ & & & & & \\ 1 & 1 & 1 & & & 1 \\ 1 & & & & & \\ 1 & 1 & & & & \\ 1 & 1 & & & & \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & & & & \\ 1 & 1 & 1 & & & 1 \\ & & 1 & & & \\ & & & & & \\ 1 & & 1 & & & 1 \\ & 1 & & & & 1 \\ & 1 & 1 & & & 1 \\ 1 & 1 & 1 & & & 1 \\ & & & & & \\ & & 1 & & & \\ 1 & & 1 & & & \\ & 1 & & & & \end{bmatrix} = M \quad (5)$$

The pseudo code of the QUATRE Algorithm is shown in Algorithm 1, where $X_{pbest,G} = [x_{1pb,G}, x_{2pb,G}, \dots, x_{ipb,G}, \dots, x_{pspb,G}]$ is the personal best position matrix. $x_{ipb,G}$, $i = 1, 2, \dots, ps$ is a vector which means the personal best position of particle i until G th generation.

Algorithm 1: Pseudo code of QUATRE.**input:**

The dimension D , popolation size ps , max iterations $MAXG$
 mutation schema to calculate B , and the fitness function $f(X)$.

Initialization:

Initialize searching space V , $G = 1$, position matrix X_G
 $X_{pbest,G} = X_G$ and calculation $X_{gbest,G}$.

Iteration:

```

1: while  $G < MAXG$  | stopCriterion do
2:   Generate matrix  $M$  by Equation (5), calculate  $\overline{M}$ 
3:   Calculation mutation Matrix  $B$  according to mutation schema.
4:    $X_{G+1} = M \otimes X_G + \overline{M} \otimes B$ 
5:   for  $i = 1 : ps$  do
6:     if  $f(x_{i,G+1})$  optimal than  $f(x_{ipb,G})$  then
7:        $x_{ipb,G+1} = x_{i,G+1}$ 
8:     else
9:        $x_{ipb,G+1} = x_{ipb,G}$ 
10:    end if
11:  end for
12:   $X_{G+1} = X_{pbest,G+1}$ 
13:   $x_{gbest,G+1} = opt\{X_{pbest,G+1}\}$ .
14:  Update  $X_{gbest,G+1}$  by Equation (2).
15:   $G = G + 1$ 
16: end while

```

output:

The global optima $X_{gbest,G}$, $f(X_{gbest,G})$.

2.2. Transfer Function

In the binary version of the meta-heuristic algorithm, the role of transfer function is very important because the value of the transfer function is the probability that the element of the position vector takes 0/1 or the probability that the element goes from 0 to 1. Therefore, the transfer function must be a bounded function of $[0,1]$. In this section, four families of transfer functions are introduced.

In Reference [62], the sigmoid transfer function was firstly proposed on binary PSO by Kennedy, J. The particles of binary PSO can only be 0 or 1 according to their position vector; as shown in Equation (6), where $v_i^k(t)$ is the velocity value of particle i at iteration t in k dimension.

$$T(v_i^k(t)) = \frac{1}{1 + e^{-v_i^k(t)}} \quad (6)$$

After converting the continuous value of velocity to a probability value, the position value of particle i at iteration $t + 1$ in k dimension $x_i^k(t + 1)$ can be updated with the probability value by Equation (7), where $rand \in [0,1]$ is a random variable. In this strategy, the sigmoid function forces particles to take values of 0 or 1 according to their velocity values.

$$x_i^k(t + 1) = \begin{cases} 0, & rand < T(v_i^k(t)) \\ 1, & rand \geq T(v_i^k(t)) \end{cases} \quad (7)$$

Subsequently, in Reference [63], S-shaped and V-shaped families of transfer functions are proposed by Mirjalili, S. The expressions are shown in Tables 2 and 3. The S-shaped families of transfer functions are extensions of the sigmoid function. Therefore, they used Equation (7) to update the position value. However, the V-shaped transfer functions used a different position update strategy as shown in Equation (8), where $x_i^k(t)^{-1}$ refers to the

binary inverse operation. In this strategy, particles stay in their current positions when their velocity values are low, and switch to their complements when the velocity values are high.

$$x_i^k(t+1) = \begin{cases} x_i^k(t)^{-1}, & rand < T(v_i^k(t)) \\ x_i^k(t), & rand \geq T(v_i^k(t)) \end{cases} \quad (8)$$

Table 2. The expressions of the S-shaped families of transfer functions.

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

Table 3. The expressions of the V-shaped families of transfer functions.

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

The new U-shaped families of transfer functions [63] and Z-shaped families of transfer functions [64] show good performance in binary PSO. The expresses are shown in Tables 4 and 5. The position value update strategy used in U-shaped and Z-shaped transfer functions is Equation (8), the same as V-shaped families of transfer functions.

Table 4. The expressions of the U-shaped families of transfer functions.

Name	Transfer Function
$U_1(x)$	$T_1(x) = \min(x^{1.5} , 1)$
$U_2(x)$	$T_2(x) = \min(x^2 , 1)$
$U_3(x)$	$T_3(x) = \min(x^3 , 1)$
$U_4(x)$	$T_4(x) = \min(x^4 , 1)$

Table 5. The expressions of the Z-shaped families of transfer functions.

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

3. Proposed Binary QUATRE Algorithm

In this section, a novel binary QUasi-Affine TRansformation Evolutionary (BQUATRE) algorithm is proposed for dimensionality reduction on HSI. First of all, mathematical analysis is performed on BQUATRE, then the improved four families of transfer functions are proposed. Furthermore, in order to balance the exploration and exploitation, a new linear increment scale factor is presented. Finally, two approaches of BQUATRE algorithm are described in detail.

3.1. Mathematical Analysis

Suppose that the position matrix at G generation X_G of BQUATRE is made up of 0 or 1. So the $X_{r1,G}$, $X_{r2,G}$, $X_{r3,G}$, $X_{r4,G}$, $X_{r5,G}$ and $X_{gbest,G}$ at G th generation are also binary matrixes. The mutation matrix B is calculated according to the schemes shown in Table 1. However, the range of scale factor F is $[0, 2]$. Therefore, the mutation matrix B will not be a binary matrix, and the position matrix X_{G+1} at $(G + 1)$ th generation obtained by Equation (1) will also not be a binary matrix. In order to further analyse, we choose the first mutation strategy in Table 1 as an example, which is Equation (9).

$$B = X_{gbest,G} + F * (X_{r1,G} - X_{r2,G}) \quad (9)$$

There are eight combinations about the values of $X_{gbest,G}$, $X_{r1,G}$ and $X_{r2,G}$, which are described as follows:

- (1) if $X_{gbest,G} = 0$, $X_{r1,G} = 0$ and $X_{r2,G} = 0$.
 $B = 0 + F * (0 - 0) = 0$.
- (2) if $X_{gbest,G} = 0$, $X_{r1,G} = 0$ and $X_{r2,G} = 1$.
 $B = 0 + F * (0 - 1) = -F$, thereby $B \in [-2, 0]$.
- (3) if $X_{gbest,G} = 0$, $X_{r1,G} = 1$ and $X_{r2,G} = 0$.
 $B = 0 + F * (1 - 0) = F$, thereby $B \in [0, 2]$.
- (4) if $X_{gbest,G} = 0$, $X_{r1,G} = 1$ and $X_{r2,G} = 1$.
 $B = 0 + F * (1 - 1) = 0$.
- (5) if $X_{gbest,G} = 1$, $X_{r1,G} = 0$ and $X_{r2,G} = 0$.
 $B = 1 + F * (0 - 0) = 1$.
- (6) if $X_{gbest,G} = 1$, $X_{r1,G} = 0$ and $X_{r2,G} = 1$.
 $B = 1 + F * (0 - 1) = 1 - F$, thereby $B \in [-1, 1]$.
- (7) if $X_{gbest,G} = 1$, $X_{r1,G} = 1$ and $X_{r2,G} = 0$.
 $B = 1 + F * (1 - 0) = 1 + F$, thereby $B \in [1, 3]$.
- (8) if $X_{gbest,G} = 1$, $X_{r1,G} = 1$ and $X_{r2,G} = 1$.
 $B = 1 + F * (1 - 1) = 1$.

From the above analysis, it can be concluded that $B \in [-2, 3]$. The same conclusion can be drawn from mutation strategies QUATRE/target/1 and QUATRE/best/1. In the last three mutation strategies, QUATRE/target/2, QUATRE/rand/2 and QUATRE/best/2, it can be concluded that $B \in [-4, 5]$ by the same way.

3.2. Improved Four Families of Transfer Functions

In this section, the results concluded from Section 3.1 will continued to be discussed. The same as in Section 3.1, we choose Equation (9) as an example, then $B \in [-2, 3]$. As shown in Figure 1, the four dotted lines represent the original S-shaped families of transfer functions define in Table 2, we can find that they do not fit the search space $[-2, 3]$. The center of search space is 0.5, while the center of S-shaped families of transfer functions is 0. Another problem is that the maximal and minimal value of these functions in $[-2, 3]$ does not reach 1 and 0. However, we update the position matrix by Equation (7), in which the *rand* is generated in $[0, 1]$. Therefore, we shift the functions to the center of the search space, then stretch the value of them to $[0, 1]$. The exact expression of the four improved S-shaped families of transfer functions are described in Table 6. The solid lines in Figure 1 show the curve of the improved four functions.

Table 6. The expressions of the improved S-shaped transfer functions.

Name	Transfer Function
$S_{1i}(x)$	$T_{1i}(x) = \begin{cases} \frac{1/(1+e^{-2(x-0.5)})-0.5}{1/(1+e^{-5})-0.5} + 0.5 & x \geq 0.5 \\ \frac{1/(1+e^{-2(x-0.5)})-0.5}{0.5-1/(1+e^5)} + 0.5 & x < 0.5 \end{cases}$
$S_{2i}(x)$	$T_{2i}(x) = \begin{cases} \frac{1/(1+e^{-(x-0.5)})-0.5}{1/(1+e^{-2.5})-0.5} + 0.5 & x \geq 0.5 \\ \frac{1/(1+e^{-(x-0.5)})-0.5}{0.5-1/(1+e^{2.5})} + 0.5 & x < 0.5 \end{cases}$
$S_{3i}(x)$	$T_{3i}(x) = \begin{cases} \frac{1/(1+e^{-(x-0.5)/2})-0.5}{1/(1+e^{-1.25})-0.5} + 0.5 & x \geq 0.5 \\ \frac{1/(1+e^{-(x-0.5)/2})-0.5}{0.5-1/(1+e^{1.25})} + 0.5 & x < 0.5 \end{cases}$
$S_{4i}(x)$	$T_{4i}(x) = \begin{cases} \frac{1/(1+e^{-(x-0.5)/3})-0.5}{1/(1+e^{-2.5/3})-0.5} + 0.5 & x \geq 0.5 \\ \frac{1/(1+e^{-(x-0.5)/3})-0.5}{0.5-1/(1+e^{2.5/3})} + 0.5 & x < 0.5 \end{cases}$

Figure 1. The original (the four dotted lines) and improved (the four solid lines) S-shaped families of transfer functions.

Similar operations are performed on the V-shaped and Z-shaped families of transfer functions. Since the maximal and minimal value of U-shaped families of transfer functions are 1 and 0, we do not need to stretch the value of them. Tables 7–9 are the exact formulas while Figures 2–4 describe the curves of the improved three families of transfer functions.

Figure 2. The original (the four dotted lines) and improved (the four solid lines) V-shaped families of transfer functions.

Figure 3. The original (the four dotted lines) and improved (the four solid lines) U-shaped families of transfer functions.

Figure 4. The original (the four dotted lines) and improved (the four solid lines) Z-shaped families of transfer functions.

Table 7. The expressions of the improved V-shaped transfer functions.

Name	Transfer Function
$V_{1i}(x)$	$T_{1i}(x) = \operatorname{erf}(\frac{\sqrt{\pi}}{2}(x - 0.5)) / \operatorname{erf}(\frac{\sqrt{\pi}}{2}(2.5))$
$V_{2i}(x)$	$T_{2i}(x) = \tanh(x - 0.5) / \tanh(2.5)$
$V_{3i}(x)$	$T_{3i}(x) = (x - 0.5) / \sqrt{1 + (x - 0.5)^2} / (2.5) / \sqrt{1 + (2.5)^2}$
$V_{4i}(x)$	$T_{4i}(x) = \frac{2}{\pi} \arctan(\frac{2}{\pi}(x - 0.5)) / \frac{2}{\pi} \arctan(\frac{2}{\pi}(2.5))$

Table 8. The expressions of the improved U-shaped transfer functions.

Name	Transfer Function
$U_{1i}(x)$	$T_{1i}(x) = \min((x - 0.5)^{1.5} , 1)$
$U_{2i}(x)$	$T_{2i}(x) = \min((x - 0.5)^2 , 1)$
$U_{3i}(x)$	$T_{3i}(x) = \min((x - 0.5)^3 , 1)$
$U_{4i}(x)$	$T_{4i}(x) = \min((x - 0.5)^4 , 1)$

Table 9. The expressions of the improved Z-shaped transfer functions.

Name	Transfer Function
$Z_{1i}(x)$	$T_{1i}(x) = \sqrt{1 - 2^{x-0.5}} / \sqrt{1 - 2^{-2.5}}$
$Z_{2i}(x)$	$T_{2i}(x) = \sqrt{1 - 5^{x-0.5}} / \sqrt{1 - 5^{-2.5}}$
$Z_{3i}(x)$	$T_{3i}(x) = \sqrt{1 - 8^{x-0.5}} / \sqrt{1 - 8^{-2.5}}$
$Z_{4i}(x)$	$T_{4i}(x) = \sqrt{1 - 20^{x-0.5}} / \sqrt{1 - 20^{-2.5}}$

3.3. New Scaling Factor Based on Exploration and Exploitation

The convergence process of meta-heuristic algorithm can be regarded as two stages: exploration and exploitation. In the early stage of the algorithm, the exploration ability is conducive to searching more space and jumping out of the local optimal solution. In the later stage of the algorithm, the approximate optimal solution has been found, and the development ability can help to find the optimal solution. Similarly, for the binary version, the position needs to be able to switch between 0 and 1 quickly in the beginning and slowly in the end. In continuous QUATRE, Liu, N. proposed a linear decrease scale factor shown as Equation (10) [48], where F_{max} and F_{min} are the predetermined maximum and minimum values of scale factor F . Usually, we set $F \in [0, 2]$, then $F_{max} = 2$ and $F_{min} = 0$. G is the current generation number, and $MAXG$ is the maximum generation number.

$$F = F_{max} - (F_{max} - F_{min}) \times \frac{G}{MAXG} \quad (10)$$

However, in the binary version, things are different, as proved by Hu, P. [54] and Liu, J. [65], as the transfer function also determines exploration and exploitation. The slope of the transfer function can be calculated to evaluate the velocity trend, for the slope indicates the switching speed of position. Figure 5 shows the curve of original $S_1(x)$, $V_1(x)$, $U_1(x)$, and $Z_1(x)$ functions described in Tables 2–5. We can find that the slope is small when the value of $|x|$ is large, while the slope is large when the value of $|x|$ is small.

Since the transfer function and scale factor F combined determine the probability of conversion in the proposed binary QUATRE, a linearly increasing scale factor is presented in this manuscript as shown in Equation (11).

$$F = (F_{max} - F_{min}) \times \frac{G}{MAXG} \quad (11)$$

Figure 5. The curve of $S1(x)$, $V1(x)$, $U1(x)$, and $Z1(x)$ transfer functions.

3.4. Binary QUATRE Algorithm—Approach 1 (BQUATRE1)

The first approach of binary QUATRE algorithm (BQUATRE1) converts the continuous position matrix at $(G + 1)$ th generation X_{G+1} to the binary. In this section, BQUATRE1 will be described in detail.

Suppose that the position matrix at G th generation X_G is obtained. First of all, the cooperative search matrix is generated at this generation M_{tmp} and M is calculated according to Equation (5). $\bar{M}$ can be obtained through binary inverse operation on the matrix M .

After that, the scale factor F can be obtained by Equation (11), that is $F = (F_{max} - F_{min}) \times \frac{G}{MAXG}$. The mutation matrix B is calculated according to a mutation strategy in Table 1. For example, we choose the first mutation strategy Equation (9), that is $B = X_{gbest,G} + F * (X_{r1,G} - X_{r2,G})$. Then X_{G+1} can be calculated by Equation (1), that is $X_{G+1} = M \otimes X_G + \bar{M} \otimes B$.

The element in X_{G+1} denoted as $x_{ik,G+1} \in [-2, 3]$ refers to the i th particle in k dimension. In order to distinguish, the continuous position matrix at $(G + 1)$ th generation is denoted as X_{G+1}^{cont} , the binary matrix at $(G + 1)$ th generation is denoted as X_{G+1}^{bin} . Then, Equation (1) can be rewritten as Equation (12).

$$X_{G+1}^{cont} = M \otimes X_G^{bin} + \bar{M} \otimes B \quad (12)$$

The particle i at k dimension of X_{G+1}^{cont} denoted as $x_{ik,G+1}^{cont}$. Convert every element $x_{ik,G+1}^{cont}$ to probability value by a specified transfer function, then whole matrix X_{G+1}^{cont} can be obtained. In detail, if S-shaped families of transfer functions are selected, the element of matrix X_{G+1}^{cont} can be updated with the probability value by Equation (13).

$$x_{ik,G+1}^{bin} = \begin{cases} 0, & rand < T(x_{ik,G+1}^{cont}) \\ 1, & rand \geq T(x_{ik,G+1}^{cont}) \end{cases} \quad (13)$$

However, if V-shaped, U-shaped or Z-shaped families of transfer functions are selected, the position matrix X_{G+1}^{bin} can be updated with the probability value by Equation (14).

$$x_{ik,G+1}^{bin} = \begin{cases} (x_{ik,G}^{bin})^{-1}, & rand < T(x_{ik,G+1}^{cont}) \\ x_{ik,G}^{bin}, & rand \geq T(x_{ik,G+1}^{cont}) \end{cases} \quad (14)$$

As shown in Equation (12), some elements of matrix X_{G+1}^{cont} come from X_G^{bin} , while the others come from B . Since the elements come from X_G^{bin} is inherently binary, the improved four families of transfer functions do not fit X_{G+1}^{cont} . In this situation, the original four families of transfer functions $T(x)$ are used to convert X_{G+1}^{cont} from a continuous matrix to binary one. The pseudo code of BQUATRE1 is described in Algorithm 2. Since most

matrices are binary, we only denote the continuous matrix, vector or number, for example, X^{cont} means X is a continuous matrix.

Algorithm 2: Pseudo code of binary QUATRE algorithm-Approach 1(BQUATRE1).

input:

The dimension D , population size ps , max iterations $MAXG$, fitness function $f(X)$, mutation schema to calculate B , and original transfer function $T(x)$.

Initialization:

Initialize searching space V , $G = 1$, position matrix X_G

$X_{pbest,G} = X_G$ and calculation $X_{gbest,G}$.

Iteration:

```

1: while  $G < MAXG$  |  $!stopCriterion$  do
2:   Generate matrix  $M$  by Equation (5), calculate  $\overline{M}$ 
3:    $F = (F_{max} - F_{min}) \times \frac{G}{MAXG}$ 
4:   Calculation mutation Matrix  $B$  according to mutation schema.
5:    $X_{G+1}^{cont} = M \otimes X_G + \overline{M} \otimes B$ 
6:   Convert every  $x_{ik,G+1}^{cont}$  to probability value  $T(x_{ik,G+1}^{cont})$  by original transfer function.
7:   Convert  $T(x_{ik,G+1}^{cont})$  to binary according to Equation (13) or Equation (14).
8:   for  $i = 1 : ps$  do
9:     if  $f(x_{i,G+1})$  optimal than  $f(x_{ipb,G})$  then
10:       $x_{ipb,G+1} = x_{i,G+1}$ 
11:     else
12:       $x_{ipb,G+1} = x_{ipb,G}$ 
13:     end if
14:   end for
15:  $X_{G+1} = X_{pbest,G+1}$ 
16:  $x_{gbest,G+1} = opt\{X_{pbest,G+1}\}$ .
17: Update  $X_{gbest,G+1}$  by Equation (2).
18:  $G = G + 1$ 
19: end while
output:
The global optima  $X_{gbest,G}, f(X_{gbest,G})$ .

```

3.5. Binary QUATRE Algorithm—Approach 2 (BQUATRE2)

The second approach of binary QUATRE algorithm (BQUATRE2) only converts the mutation matrix B to the binary. In this section, BQUATRE2 will be described in detail.

Again, we suppose that the position matrix at G th generation X_G is got. First of all, M and $\overline{M}$ can be obtained in the same way.

The mutation matrix B is calculated according to Equations (9) and (11). The particle i of B in k dimension can be denoted as b_{ik} , where $i = 1, 2, \dots, ps$ and $k = 1, 2, \dots, D$. Since all elements in B are continuous values, the improved families of transfer functions $T_i(x)$ are used in this approach. In order to distinguish, the continuous mutation matrix is denoted as B^{cont} , while the binary mutation matrix denoted as B^{bin} .

Then, we should convert the probability value $T_i(x)$ to binary. If the S-shaped families of transfer functions are selected, consider the second case in Section 3.1, that is if $X_{gbest,G} = 0$, $X_{r1,G} = 0$ and $X_{r2,G} = 1$, then $B \in [-2, 0]$. In this situation, the b_{ik} should have a high probability to be 0 for only the value of $X_{r2,G}$ is not 0. Similarly, consider the seventh case in Section 3.1, if $X_{gbest,G} = 1$, $X_{r1,G} = 1$ and $X_{r2,G} = 0$, then $B \in [1, 3]$. If the value of b_{ik} have a high probability to be 0, the design is reasonable. Therefore, if the S-shaped functions are selected, the Equation (7) is replaced by Equation (15) to updated

with the probability value in BQUATRE2. Among them, b_{ik}^{bin} is the element of B^{bin} , b_{ik}^{cont} is the element of B^{cont} , where $i = 1, 2, \dots, ps$ and $k = 1, 2, \dots, D$.

$$b_{ik}^{bin} = \begin{cases} 1, & rand < T_i(b_{ik}^{cont}) \\ 0, & rand \geq T_i(b_{ik}^{cont}) \end{cases} \quad (15)$$

If the improved V-shaped, U-shaped or Z-shaped families of functions, which are described in Tables 7–9 are chosen, we still adopt the strategy shown as Equation (8) to update the probability value. In this approach, the update function is shown in Equation (16), where $x_{ik,G}$ is a binary value that shows the position of particle i in k dimension at G generation. Then, the whole binary mutation matrix B^{bin} can be obtained. Finally, the position matrix X_{G+1} can be obtained.

$$b_{ik}^{bin} = \begin{cases} (x_{ik,G})^{-1}, & rand < T_i(b_{ik}^{cont}) \\ x_{ik,G}, & rand \geq T_i(b_{ik}^{cont}) \end{cases} \quad (16)$$

The pseudo code of BQUATRE2 described in Algorithm 3.

Algorithm 3: Pseudo code of binary QUATRE algorithm-Approach 2(BQUATRE2).

input:

The dimension D , population size ps , max iterations $MAXG$, fitness function $f(X)$ mutation schema to calculate B , and improved transfer function $T_i(x)$.

Initialization:

Initialize searching space V , $G = 1$, position matrix X_G

$X_{pbest,G} = X_G$ and calculation $X_{gbest,G}$.

Iteration:

```

1: while  $G < MAXG$  |  $stopCriterion$  do
2:   Generate matrix  $M$  by Equation (5), calculate  $\overline{M}$ 
3:    $F = (F_{max} - F_{min}) \times \frac{G}{MAXG}$ 
4:   Calculation continues mutation Matrix  $B^{cont}$  according to mutation schema.
5:   Convert every element  $b_{ik}^{cont}$  to probability value  $T_i(b_{ik}^{cont})$  by improved
     transfer function .
6:   Convert  $T_i(b_{ik}^{cont})$  to binary  $b_{ik}^{bin}$  by Equation (14) or Equation (15).
7:    $X_{G+1} = M \otimes X_G + \overline{M} \otimes B^{bin}$ 
8:   for  $i = 1 : ps$  do
9:     if  $f(x_{i,G+1})$  optimal than  $f(x_{ipb,G})$  then
10:       $x_{ipb,G+1} = x_{i,G+1}$ 
11:     else
12:       $x_{ipb,G+1} = x_{ipb,G}$ 
13:     end if
14:   end for
15:    $X_{G+1} = X_{pbest,G+1}$ 
16:    $x_{gbest,G+1} = opt\{X_{pbest,G+1}\}$ .
17:   Update  $X_{gbest,G+1}$  by Equation (2).
18:    $G = G + 1$ 
19: end while

```

output:

The global optima $X_{gbest,G}, f(X_{gbest,G})$.

4. Experimental Results and Analysis

4.1. Benchmark Function

In this section, the proposed BQUATRE1 and BQUATRE2 are examined by 23 benchmark functions. The mathematical formulas and properties of these functions are described in Tables 10–12; especially, D means the dimension of function and f_{min} represents the optimum. In detail, Table 10 contains the unimodal functions (denoted as f1–f7), Table 11 shows

the details of common multimodal functions (denoted as f8–f13) and Table 12 displays the details of multimodal functions in low dimension (denoted as f14–f23).

Unimodal functions have only a global optimal solution and no local optimal solution, which can verify whether the algorithms can find the global optimal solution within the finite population size and iterations. Multimodal functions have many local optima, which can verify whether the algorithms can fall into the local optimal solutions or not. Since the function dimension is too low, the algorithms are prone to precocity and fall into local optimum. Therefore, the multimodal functions in low dimension can verify the convergence results under more stringent conditions.

Table 10. Unimodal benchmark functions.

Name	Function	Search Space	D	f_{min}
Shpere	$f_1(x) = \sum_{i=1}^n x_i^2$	$[-100,100]$	30	0
Schwefel's function 2.21	$f_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	$[-10,10]$	30	0
Schwefel's function 1.2	$f_3(x) = \sum_{i=1}^n (\sum_{j=1}^i x_j)^2$	$[-100,100]$	30	0
Schwefel's function 2.22	$f_4(x) = \max_i \{ x_i , 1 \leq i \leq n\}$	$[-100,100]$	30	0
Rosenbroke	$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30,30]$	30	0
Step	$f_6(x) = \sum_{i=1}^n ([x_i + 0.5])^2$	$[-100,100]$	30	0
Dejong's noisy	$f_7(x) = \sum_{i=1}^n ix_i^4 + rand[0,1)$	$[-1.28,1.28]$	30	0

Table 11. Common multimodal benchmark functions.

Name	Function	Search Space	D	f_{min}
Schwefel	$f_8(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	$[-500,500]$	30	-12,569
Rastrigin	$f_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$[-5.12,5.12]$	30	0
Ackley	$f_{10}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)) + 20 + e$	$[-32,32]$	30	0
Griewank	$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600,600]$	30	0
Generalized penalized 1	$f_{12} = \frac{\pi}{n} \{10 \sin(\pi y_1) + \sum_{i=1}^n (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^n u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50,50]$	30	0
Generalized penalized 2	$f_{13}(x) = 0.1 \{\sin^2(3\pi x_1) + \sum_{i=1}^n (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^n u(x_i, 10, 100, 4)$	$[-50,50]$	30	0

Table 12. Multimodal benchmark functions in low dimension.

Name	Function	Search Space	D	f_{min}
Fifth of Dejong	$f_{14}(x) = (\frac{1}{500} \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6})^{-1}$	$[-65, 65]$	2	1
Kowalik	$f_{15}(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	$[-5, 5]$	4	0.00030
Six-hump camel back	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]$	2	-1.0316
Branins	$f_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	$[-5, 5]$	2	0.398
Goldstein-Price	$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$[-2, 2]$	2	3
Hartman 1	$f_{19}(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2)$	$[1, 3]$	3	-3.86
Hartman 2	$f_{20}(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2)$	$[0, 1]$	6	-3.32
Shekel 1	$f_{21}(x) = -\sum_{i=1}^5 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]$	4	-10.1532
Shekel 2	$f_{22}(x) = -\sum_{i=1}^7 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]$	4	-10.4028
Shekel 3	$f_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]$	4	-10.5363

Two approaches of BQUATRE algorithm with four families of transfer functions are examined in the experiment. Due to space limitation, we choose the second function of these families of transfer functions as an example, that is S_2 , V_2 , U_2 , Z_2 , S_{2i} , V_{2i} , U_{2i} and Z_{2i} . First of all, the S-shaped and V-shaped families of transfer functions are compared, that is the first approach of BQUATRE algorithm with S_2 transfer function (record as BQUATRE1-S2), the first approach of BQUATRE algorithm with V_2 transfer function (record as BQUATRE1-V2), the second approach of BQUATRE algorithm with the improved transfer function S_{2i} (record as BQUATRE2-S2i) and the second approach of BQUATRE algorithm with the improved transfer function V_{2i} (record as BQUATRE2-V2i). Since QUATRE is the improved DE, the proposed binary QUATRE is compared with Binary DE (record as BDE). The binary PSO is the most primitive binary algorithm, so we also used it in our experiment (record as BPSO). The third compared algorithm is the Advanced Binary Grey Wolf Optimizer algorithm with the V3a transfer function (record as ABGWO-V3a), which is proposed in Reference [54] by Hu, P. ABGWO-V3a can obtain better performance than many other binary meta-heuristic algorithms. Table 13 shows the experimental results of BDE, ABGWO-V3a, BQUATRE1-S2, BQUATRE1-V2, BQUATRE2-S2i, BQUATRE2-V2i, and BPSO. In detail, all algorithms in the experiment run 30 times, 500 iterations and 30 individuals on each benchmark function. Among them, AVG and STD are the mean and standard deviation of the results of 30 times, respectively. The results in red and in blue are the best result of the seven algorithms. The blue results mean that all algorithms obtained the optimal solution. The last line refers to the number of times the algorithm gets the optimal result.

From Table 13, we can see that the seven algorithms get the same solutions on 6 benchmark functions, which are all multimodal functions in low dimensions. BQUATRE1-V2, BQUATRE2-S2i and BQUATRE2-V2i obtained 14, 15 and 14 red results, respectively. However, BDE, ABGWO-V3a, and BPSO only obtained 6, 3 and 1 red results, respectively. Therefore, BQUATRE1-V2, BQUATRE2-S2i and BQUATRE2-V2i are superior to BDE, ABGWO-V3a, and BPSO. At the same time, we can see that BQUATRE1-S2 does not perform well. However, BQUATRE2-S2i obtained the best results compared to the other five algorithms. This result shows Equation (14) is effective for BQUATRE2. Figures 6–8 are the visualization results of Table 13.

Table 13. The statistical results of BDE, ABGWO-V3a, BQUATE1-S2, BQUATE1-V2, BQUATE2-S2i, BQUATE2-V2i, and BPSO.

Function	BDE		ABGWO-V3a		BQUATE1-S2		BQUATE1-V2		BQUATE2-S2i		BQUATE2-V2i		BPSO	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
f1	3.7333	1.8742	3.4333	1.3047	3.8333	0.5307	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	7.8333	1.1769
f2	3.5333	1.4559	2.7000	1.3170	3.6000	0.8137	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	8.1667	1.0532
f3	93.5333	69.4057	73.6667	63.7264	116.7000	38.7220	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	652.1333	178.1527
f4	1.0000	0.0000	1.0000	0.0000	0.9667	0.1826	0.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000
f5	0.0000	0.0000	543.4667	140.4708	333.1667	79.2421	29.0000	0.0000	0.0000	0.0000	47.2000	63.5227	172.7667	233.0352
f6	13.3000	3.0783	13.9000	2.3134	14.6333	1.8705	7.5000	0.0000	7.5000	0.0000	7.5000	0.0000	24.2333	1.9989
f7	45.0334	18.4923	32.2667	16.4231	43.7976	7.8121	0.0001	0.0001	0.0022	0.0021	0.0376	0.0306	108.7332	13.3579
f8	−25.2441	0.0000	−22.6075	0.8765	−18.2038	0.7808	−19.5502	0.9293	−25.2441	0.0000	−25.2441	0.0000	−17.1941	1.0525
f9	3.8667	1.5916	2.9000	1.2134	4.0000	0.5252	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	7.9667	0.9279
f10	1.2908	0.2408	1.2618	0.2606	1.3185	0.1427	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.9306	0.1250
f11	0.1725	0.0507	0.1254	0.0504	0.1020	0.0218	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2978	0.0443
f12	2.3200	0.2834	2.1036	0.2164	2.2528	0.1022	1.6690	0.0000	1.6690	0.0000	1.6690	0.0000	3.0290	0.2758
f13	0.0000	0.0000	0.3133	0.1383	0.8500	0.0900	0.6867	0.0860	0.0000	0.0000	0.0000	0.0000	0.9300	0.1264
f14	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000
f15	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000
f16	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
f17	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000
f18	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000
f19	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000
f20	−0.1556	0.0280	−0.1440	0.0433	−0.1657	0.0000	−0.1657	0.0000	−0.1657	0.0000	−0.1657	0.0000	−0.1657	0.0000
f21	−5.0552	0.0000	−5.0552	0.0000	−5.0552	0.0000	−5.0552	0.0000	−5.0552	0.0000	−5.0552	0.0000	−1.9948	1.8770
f22	−5.0877	0.0000	−5.0877	0.0000	−5.0877	0.0000	−5.0877	0.0000	−5.0877	0.0000	−5.0877	0.0000	−3.1387	2.1192
f23	−5.1285	0.0000	−5.1285	0.0000	−5.1285	0.0000	−5.1285	0.0000	−5.1285	0.0000	−5.1285	0.0000	−2.6071	2.0951
count	6 + 6		3 + 6		4 + 6		14 + 6		15 + 6		14 + 6		1 + 6

Figure 6. The optimal values of BDE, ABGWO-V3a, BQUATE1-S2, BQUATE1-V2, BQUATE2-S2i, BQUATE2-V2i, and BPSO on f1-f7.

Figure 7. The optimal values of BDE, ABGWO-V3a, BQUATE1-S2, BQUATE1-V2, BQUATE2-S2i, BQUATE2-V2i, and BPSO on f8-f13.

Figure 8. The optimal values of BDE, ABGWO-V3a, BQUATE1-S2, BQUATE1-V2, BQUATE2-S2i, BQUATE2-V2i, and BPSO on f14-f23.

Moreover, the U-shaped and Z-shaped families of transfer functions are compared to BDE, ABGWO-V3a, and BPSO, that is the first approach of BQUATRE algorithm with U_2 transfer function (record as BQUATRE1-U2), the first approach of BQUATRE algorithm with Z_2 transfer function (record as BQUATRE1-Z2), the second approach of BQUATRE algorithm with the improved transfer function U_{2i} (record as BQUATRE2-U2i) and the second approach of BQUATRE algorithm with the improved transfer function Z_{2i} (record as BQUATRE2-Z2i). The results are shown in Table 14.

From Table 14, it can be seen that the seven algorithms get the same solutions on six benchmark functions. However, BQUATRE1-U2 gets the best results on other 12 functions. Furthermore, BQUATRE1-Z2, BQUATRE2-U2i and BQUATRE2-Z2i obtained 16, 15 and 7 red results, respectively. However, BDE, ABGWO-V3a, and BPSO only obtained 6, 3 and 1 red results, respectively. On the whole, BQUATRE1-U2, BQUATRE1-Z2, BQUATRE2-U2i and BQUATRE2-Z2i are superior to BDE, ABGWO-V3a, and BPSO. From Tables 13 and 14, we can conclude that V-shaped, U-shaped and Z-shaped functions in BQUATRE1 and BQUATRE2 are superior to BDE, ABGWO-V3a, and BPSO. The BQUATRE2 with S-shaped transfer function can also perform well. The proposed second approach of BQUATRE can perform well on the unimodal functions and common multimodal functions. However, at the same time, we can see the improved algorithm does not improve much on multimodal functions in low dimensions. Figure 9–11 are the visualization results of Table 14.

To further compare the results, t-test is used for a significance test. The t-test is used to compare the mean value of two groups of data. 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 0.01, which means very significant. Table 15 shows the results between ABGWO-3Va and QUATRE1-S2, QUATRE1-V2, QUATRE1-U2, QUATRE1-Z2, QUATRE2-S2i, QUATRE2-V2i, QUATRE2-U2i, QUATRE2-Z2i. “+” appears that the compared algorithm is superior to ABGWO-3Va, and “−” indicates that the algorithm is inferior to ABGWO-3Va. “=” implies that the performance of the algorithms is consistent.

Table 14. The statistical results of BDE, ABGWO-V3a, BQUATE1-U2, BQUATE1-Z2, BQUATE2-U2i, BQUATE2-Z2i, and BPSO.

Function	BDE		ABGWO-V3a		BQUATE1-U2		BQUATE1-Z2		BQUATE2-U2i		BQUATE2-Z2i		BPSO	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
f1	3.7333	1.8742	3.4333	1.3047	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1000	0.3051	7.8333	1.1769
f2	3.5333	1.4559	2.7000	1.3170	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1000	0.3051	8.1667	1.0532
f3	93.5333	69.4057	73.6667	63.7264	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6000	1.0700	652.1333	178.1527
f4	1.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000
f5	0.0000	0.0000	543.4667	140.4708	29.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	172.7667	233.0352
f6	13.3000	3.0783	13.9000	2.3134	7.5000	0.0000	7.5000	0.0000	7.5000	0.0000	7.5667	0.3651	24.2333	1.9989
f7	45.0334	18.4923	32.2667	16.4231	0.0001	0.0001	0.0001	0.0001	0.0031	0.0031	1.2172	1.1935	108.7332	13.3579
f8	−25.2441	0.0000	−22.6075	0.8765	−19.1294	0.9358	−25.2441	0.0000	−25.2441	0.0000	−25.2441	0.0000	−17.1941	1.0525
f9	3.8667	1.5916	2.9000	1.2134	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1000	0.3051	7.9667	0.9279
f10	1.2908	0.2408	1.2618	0.2606	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0717	0.2188	1.9306	0.1250
f11	0.1725	0.0507	0.1254	0.0504	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0078	0.0092	0.2978	0.0443
f12	2.3200	0.2834	2.1036	0.2164	1.6690	0.0000	1.6690	0.0000	1.6690	0.0000	1.7043	0.0607	3.0290	0.2758
f13	0.0000	0.0000	0.3133	0.1383	0.7500	0.1196	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.9300	0.1264
f14	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000	12.6705	0.0000
f15	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000	0.1484	0.0000
f16	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
f17	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000	27.7029	0.0000
f18	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000	600.0000	0.0000
f19	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000	−0.3348	0.0000
f20	−0.1556	0.0280	−0.1440	0.0433	−0.1657	0.0000	−0.1657	0.0000	−0.1657	0.0000	−0.1657	0.0000	−0.1657	0.0000
f21	−5.0552	0.0000	−5.0552	0.0000	−4.9161	0.7619	−5.0552	0.0000	−5.0552	0.0000	−5.0552	0.0000	−1.9948	1.8770
f22	−5.0877	0.0000	−5.0877	0.0000	−5.0877	0.0000	−5.0877	0.0000	−5.0877	0.0000	−5.0877	0.0000	−3.1387	2.1192
f23	−5.1285	0.0000	−5.1285	0.0000	−4.7105	1.2754	−5.1285	0.0000	−5.1285	0.0000	−5.1285	0.0000	−2.6071	2.0951
count	6 + 6		3 + 6		12 + 6		16 + 6		15 + 6		7 + 6		1 + 6

Figure 9. The optimal values of BDE, ABGWO-V3a, BQUATE1-U2, BQUATE1-Z2, BQUATE2-U2i, BQUATE2-Z2i, and BPSO on f1-f7.

Figure 10. The optimal values of BDE, ABGWO-V3a, BQUATE1-U2, BQUATE1-Z2, BQUATE2-U2i, BQUATE2-Z2i, and BPSO on f8-f13.

Figure 11. The optimal values of BDE, ABGWO-V3a, BQUATE1-U2, BQUATE1-Z2, BQUATE2-U2i, BQUATE2-Z2i, and BPSO on f14-f23.

Table 15. The *t*-test results of the compared algorithms (QUATRE1-S2, QUATRE1-V2, QUATRE1-U2, QUATRE1-Z2, QUATRE2-S2i, QUATRE2-V2i, QUATRE2-U2i, and QUATRE2-Z2i) on ABGWO-V3a.

Function	QUATRE1-S2	QUATRE1-V2	QUATRE1-U2	QUATRE1-Z2	QUATRE2-S2i	QUATRE2-V2i	QUATRE2-U2i	QUATRE2-Z2i
f1	=	+	+	+	+	+	+	+
f2	−	+	+	+	+	+	+	+
f3	−	+	+	+	+	+	+	+
f4	=	=	=	=	=	=	=	=
f5	+	+	+	+	+	+	+	+
f6	=	+	+	+	+	+	+	+
f7	−	+	+	+	+	+	+	+
<hr/>								
f8	−	−	−	+	+	+	+	+
f9	−	+	+	+	+	+	+	+
f10	=	+	+	+	+	+	+	+
f11	=	+	+	+	+	+	+	+
f12	−	+	+	+	+	+	+	+
f13	−	−	−	+	+	+	+	+
<hr/>								
f14	=	=	=	=	=	=	=	=
f15	=	=	=	=	=	=	=	=
f16	=	=	=	=	=	=	=	=
f17	=	=	=	=	=	=	=	=
f18	=	=	=	=	=	=	=	=
f19	=	=	=	=	=	=	=	=
f20	+	+	+	+	+	+	+	+
f21	=	=	=	=	=	=	=	=
f22	=	=	=	=	=	=	=	=
f23	=	=	=	=	=	=	=	=

We can see from Table 15 that the four BQUATRE2 algorithms and BQUATRE1 with Z2 transfer function are not inferior to ABGWO-3Va on all 23 benchmark functions. BQUATRE1 with V2 and U2 transfer function are superior to ABGWO-3Va on 11 functions, while inferior on 2 functions.

Table 16 shows the results between BDE and the eight BQUATRE. From Table 16, QUATRE1-Z2, QUATRE2-S2i, QUATRE2-U2i and QUATRE2-Z2i are not inferior to BDE on all 23 benchmark functions. QUATRE1-V2, QUATRE1-U2 and QUATRE2-V2i are superior to BDE on 9 functions, while inferior on 1 or 2 functions.

Table 16. The *t*-test results of the compared algorithms (QUATRE1-S2, QUATRE1-V2, QUATRE1-U2, QUATRE1-Z2, QUATRE2-S2i, QUATRE2-V2i, QUATRE2-U2i, and QUATRE2-Z2i) on BDE.

Function	QUATRE1-S2	QUATRE1-V2	QUATRE1-U2	QUATRE1-Z2	QUATRE2-S2i	QUATRE2-V2i	QUATRE2-U2i	QUATRE2-Z2i
f1	=	+	+	+	+	+	+	+
f2	=	+	+	+	+	+	+	+
f3	=	+	+	+	+	+	+	+
f4	=	=	=	=	=	=	=	=
f5	−	−	−	=	=	−	=	=
f6	=	+	+	+	+	+	+	+
f7	=	+	+	+	+	+	+	+
<hr/>								
f8	−	−	−	=	=	=	=	=
f9	=	+	+	+	+	+	+	+
f10	=	+	+	+	+	+	+	+
f11	+	+	+	+	+	+	+	+
f12	=	+	+	+	+	+	+	+
f13	−	−	−	=	=	=	=	=
<hr/>								
f14	=	=	=	=	=	=	=	=
f15	=	=	=	=	=	=	=	=
f16	=	=	=	=	=	=	=	=
f17	=	=	=	=	=	=	=	=
f18	=	=	=	=	=	=	=	=
f19	=	=	=	=	=	=	=	=
f20	=	=	=	=	=	=	=	=
f21	=	=	=	=	=	=	=	=
f22	=	=	=	=	=	=	=	=
f23	=	=	=	=	=	=	=	=

Finally, we compare the runtime of the 10 algorithms in this experiment, as shown in Table 17. It can be seen that BDE algorithm has the shortest running time and ABGWO-V3A has the longest running time. As a whole, the four BQUATE2 takes longer to run than BQUATE1, as the transfer functions they used are more complex.

Table 17. The time consumption of the algorithms(BDE, ABGWO-V3a, BQUATE1-S2, BQUATE1-V2, BQUATE1-U2, BQUATE1-Z2, BQUATE2-S4i, BQUATE2-V4i, BQUATE2-U4i, and BQUATE2-Z4i).

Function	BDE	ABGWO-V3a	BQUATE1-S2	BQUATE1-V2	BQUATE1-U2	BQUATE1-Z2	BQUATE2-S4i	BQUATE2-V4i	BQUATE2-U4i	BQUATE2-Z4i
f1	0.62744	16.2509	3.4680	3.4155	5.3539	3.6765	6.7833	6.3142	5.9397	6.6418
f2	0.70993	16.7380	2.5546	3.2635	5.2737	3.7985	6.6544	6.3128	6.0067	6.5947
f3	2.1763	17.2174	3.1963	3.8897	6.1809	4.6047	7.6322	7.0619	6.1531	7.3415
f4	0.6807	16.7796	2.6008	3.3246	5.3231	3.9412	6.8011	6.2822	5.6701	6.5871
f5	0.8110	17.6103	2.7293	3.3003	5.4441	4.2954	6.9998	6.3329	5.7365	6.5454
f6	0.7057	15.4679	2.5756	3.2538	5.3656	4.4473	7.2255	6.2649	5.6854	6.6404
f7	0.7183	15.7941	2.5958	3.2864	5.4784	4.2904	6.7943	6.2623	5.6009	6.6486
f8	0.7162	14.9921	2.5957	3.4029	5.4349	4.3990	7.5326	6.2789	5.7329	6.4835
f9	0.7161	17.7780	2.5965	3.2933	5.3374	4.0391	8.2188	6.3012	6.6056	6.6817
f10	0.7351	15.4774	2.6108	3.2774	5.4029	3.7106	7.8801	6.3021	6.4067	6.6676
f11	0.7716	14.5106	2.6574	3.2962	5.3558	3.7291	8.3399	6.5746	6.3531	6.6952
f12	1.2479	14.7338	3.0666	3.5307	5.6124	4.0569	7.9435	6.4522	6.6786	7.2990
f13	1.2632	14.7007	3.1742	3.6226	5.9234	4.0865	8.1302	6.4171	6.5651	6.8091
f14	0.6115	1.8261	1.4992	1.0958	1.4449	1.2067	1.8882	1.5145	1.5213	1.5724
f15	0.6694	2.0374	0.5284	0.6029	1.1020	0.7056	1.3289	1.0319	1.0575	1.0909
f16	0.5553	1.0338	0.4202	0.3506	0.6607	0.4244	0.9298	0.5916	0.6031	0.6352
f17	0.5428	1.0266	0.3377	0.3545	0.5409	0.4188	0.5847	0.5956	0.5930	0.6113
f18	0.5592	1.0166	1.5641	0.3535	0.5985	0.4313	0.5706	0.5909	0.5966	0.6220
f19	0.7770	1.6017	1.1716	0.5437	1.0079	0.6612	0.8813	0.9154	0.9052	0.9389
f20	0.7973	3.0584	2.4736	0.8634	1.4021	1.0917	1.4618	1.5124	1.5285	1.5924
f21	0.8798	2.1233	1.6826	0.6974	1.0730	0.8095	1.1294	1.1596	1.1616	1.2014
f22	0.9692	2.1611	1.3334	0.7354	1.2336	0.8218	1.1837	1.2024	1.2066	1.2449
f23	1.0245	2.2511	1.1589	0.8068	1.1845	0.8869	1.4182	1.2899	1.2973	1.3347

4.2. Hyperspectral Imagery

In this section, the proposed BQUATRE1 and BQUATRE2 are used for dimensionality reduction in hyperspectral dataset. The dataset used in the experiment is the Indian Pines image, which is obtained by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor in 1992. The dataset captures the Indian Pines unlabeled agricultural site in northwestern Indiana and contains $220 \times 145 \times 145$ bands. The spatial resolution of this image is 20 m per pixel. The water absorption bands of HSI are seriously polluted by noise and are not suitable for classification. Therefore, twenty water absorption bands have been deleted from the Indian Pines. In the image, there are 16 features, including corn, soybean, alfalfa, wheat, oat, grass, tree, etc. Some features are subdivided; for example, corn is divided into no-tillage, low-tillage and traditional tillage. Figure 12 is the false-color composite image and ground truth data of Indian Pines. As is known, the HSI cannot be displayed on the computer because it has too many bands. False color image is the color image obtained by the synthesis of different wavebands and is often used in the classification of remote sensing images. False color synthesis technology is one of the methods of image enhancement. It converts the images composed of many bands (more than four) into three-bands or four-bands synthesis images. Figure 12a is synthesized by three-bands synthesis technology.

Figure 12. Thematic maps for the Indian Pines data set with 16 classes. (a) false-color composite; (b) ground truth data with 16 classes in different colors.

Classification performance is the main index of dimensionality reduction in HSI. In this manuscript, three common metrics are used to evaluate classifier performance, namely: overall accuracy(OA), average accuracy (AA) and Kappa coefficient. Therefore, the fitness function is Equation (17), where $OA_{kfoldLoss}$, $AA_{kfoldLoss}$ and $Kappa_{kfoldLoss}$ are the OA, AA and Kappa coefficient errors of k-fold cross validation; $W_i, i = 1, 2, 3$ are weight coefficients. In our experiment, we set $k = 5$ and set $W_1 = 0.4$, $W_2 = 0.3$, $W_3 = 0.3$, for the OA is the most important index of the three classification performances.

$$fitness = W_1 * OA_{kfoldLoss} + W_2 * AA_{kfoldLoss} + W_3 * Kappa_{kfoldLoss} \quad (17)$$

In this experiment, we use the first function of the families of the transfer function. In order to compare S-shaped, V-shaped, U-shaped and Z-shaped transfer functions, we adopt the approach BQUATRE1 with U-shaped and Z-shaped transfer functions, the BQUATRE2 with with S-shaped and V-shaped ones. The compared dimensionality reduction algorithms are classic PCA and LDA. The number of dimensions selected by PCA and LDA is 10, which is computed by the "intrinsic dimensionality estimation" function of "Matlab toolbox for dimensionality reduction" [66]. The classifier used in experiments is Support Vector Machine (SVM), which is implemented from a library for LIBSVM and used a Gaussian kernel [67]. Since it is difficult to obtain samples of hyperspectral images, the classification with fewer samples is of more practical significance. Thereby, in this experiment, the training samples are randomly chosen and account for 5% of the ground truth. Thereby,

seven algorithms are compared in this experiment; that is, the traditional SVM without dimensionality reduction (record as SVM), PCA performed to dimensionality reduction before SVM (record as PCA-SVM), LDA performed to dimensionality reduction before SVM (record as LDA-SVM), BQUATRE1 performed with U-shaped function to dimensionality reduction before SVM (record as BQUATRE1-U1-SVM), BQUATRE1 performed with Z-shaped function to dimensionality reduction before SVM (record as BQUATRE1-Z1-SVM), BQUATRE2 performed with improved S-shaped function to dimensionality reduction before SVM (record as BQUATRE2-S1i-SVM), and BQUATRE2 performed with improved V-shaped function to dimensionality reduction before SVM (record as BQUATRE2-V1i-SVM). Furthermore, the proposed algorithms run 500 iterators with 20 populations. We performed random 5% sampling 500 times before doing the experiment in order to ensure the robustness.

Table 18 describes the classification performance of these algorithms performed on the Indian Pines dataset. The red color means the proposed algorithm performs better than all three compared algorithms on this index. We can see that the proposed four algorithms all get higher OA and Kappa indexes than the three compared algorithms. However, the AA value of all proposed algorithms are slightly lower than LDA-SVM. That is due to the poor performance in classifying the "oats" and "alfalfa" by the proposed algorithms. It can be seen that the samples of these two classes are smaller than others from Figure 12. LDA-SVM correctly classified a few samples in these two classes that has a greater impact on the value of AA, but has a smaller impact on the value of OA. Figure 13 is the visualization results of the average OA, AA, Kappa values of the BQUATE2-S1i-SVM, BQUATE2-V1i-SVM, BQUATE1-U1-SVM, BQUATE1-Z1-SVM, SVM, PCA-SVM, and LDA-SVM on Indian Pines.

In order to visually see the classification results, we paint the different ground features with different colors. Figure 14 shows the classification maps. As a whole, we can see that the proposed algorithms perform better for dimensionality reduction of HSI than PCA and LDA.

Figure 13. The OA, AA, Kappa values of BQUATE2-S1i-SVM, BQUATE2-V1i-SVM, BQUATE1-U1-SVM, BQUATE1-Z1-SVM, SVM, PCA-SVM, and LDA-SVM on Indian Pines.

Table 18. The classification performance of BQUATE2-S1i-SVM, BQUATE2-V1i-SVM, BQUATE1-U1-SVM, BQUATE1-Z1-SVM, SVM, PCA-SVM, and LDA-SVM on Indian Pines

Accurate	BQUATE2-S1i-SVM		BQUATE2-V1i-SVM		BQUATE1-U1-SVM		BQUATE1-Z1-SVM		SVM		PCA-SVM		LDA-SVM	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
Alfalfa	0.5281	0.1412	0.4837	0.1407	0.5484	0.1489	0.5368	0.1400	0.4913	0.1319	0.6820	0.0403	0.7228	0.1128
Corn-no till	0.7477	0.0447	0.7496	0.0406	0.7408	0.0401	0.7457	0.0374	0.6766	0.0420	0.6065	0.0399	0.7836	0.0389
Corn-min till	0.6954	0.1016	0.7006	0.0855	0.7016	0.0886	0.7060	0.0836	0.5952	0.0443	0.3659	0.0801	0.6062	0.0400
Corn	0.4708	0.0757	0.4636	0.0552	0.4780	0.0685	0.4710	0.0594	0.4530	0.0526	0.3118	0.0575	0.4452	0.0341
Grass/pasture	0.8923	0.0303	0.8909	0.0265	0.8883	0.0295	0.8863	0.0276	0.8561	0.0265	0.8380	0.0302	0.8804	0.0169
Grass/trees	0.9217	0.0184	0.9192	0.0139	0.9172	0.0156	0.9231	0.0180	0.9087	0.0156	0.9272	0.0324	0.9605	0.0170
Grass/pasture-mowed	0.7663	0.1371	0.8224	0.1325	0.8261	0.1291	0.7637	0.1574	0.6573	0.1286	0.3899	0.0549	0.5845	0.1107
Hay-windrowed	0.9915	0.0049	0.9912	0.0029	0.9933	0.0035	0.9908	0.0057	0.9925	0.0039	0.9938	0.0049	0.9949	0.0045
Oats	0.4923	0.0940	0.4808	0.1046	0.4912	0.0916	0.4931	0.1025	0.4082	0.0677	0.3652	0.1060	0.6181	0.1453
Soybean-no till	0.6434	0.0359	0.6432	0.0365	0.6477	0.0363	0.6463	0.0370	0.6373	0.0443	0.5449	0.0569	0.6072	0.0346
Soybean-min till	0.8350	0.0250	0.8328	0.0245	0.8286	0.0258	0.8337	0.0262	0.8161	0.0280	0.7026	0.0480	0.8039	0.0178
Soybean-clean till	0.6566	0.0528	0.6578	0.0550	0.6540	0.0530	0.6635	0.0538	0.6012	0.0411	0.5602	0.0567	0.6359	0.0758
Wheat	0.9110	0.0433	0.9138	0.0397	0.9093	0.0424	0.9067	0.0450	0.9018	0.0493	0.8012	0.0438	0.9862	0.0061
Woods	0.9436	0.0165	0.9409	0.0183	0.9419	0.0170	0.9418	0.0180	0.9405	0.0165	0.9325	0.0103	0.9753	0.0086
Bldg-Grass-Tree-Drives	0.6699	0.0845	0.6571	0.0856	0.6547	0.0818	0.6637	0.0745	0.6052	0.0794	0.6559	0.0879	0.6724	0.0379
Stones-steel towers	0.8387	0.0424	0.8486	0.0540	0.8391	0.0589	0.8414	0.0529	0.8176	0.0685	0.7817	0.1337	0.9727	0.0205
OA	0.7815	0.0146	0.7805	0.0151	0.7799	0.0147	0.7827	0.0141	0.7473	0.0112	0.6717	0.0205	0.7739	0.0085
AA	0.7503	0.0160	0.7498	0.0158	0.7538	0.0169	0.7508	0.0158	0.7099	0.0121	0.6537	0.0192	0.7656	0.0134
Kappa	0.7522	0.0158	0.7510	0.0163	0.7504	0.0160	0.7535	0.0153	0.7139	0.0121	0.6289	0.0219	0.7442	0.0090

Figure 14. The classification maps of the different tested algorithms. (a) Ground truth (b) BQUATE2-S1i-SVM (c) BQUATE2-V1i-SVM (d) BQUATE1-U1-SVM (e) BQUATE1-Z1-SVM (f) SVM (g) PCA-SVM (h) LDA-SVM.

5. Conclusions

In this manuscript, we convert the QUATRE algorithm to binary version by two approaches in order to enable the QUATRE algorithm to solve the practical application of binary types. In the first approach, the new individuals produced by mutation and crossover operation are binarized. In the second approach, binarization is done after mutation, then perform cross operation with other individuals. Mathematical analysis is performed on the proposed algorithm and the improved transfer functions are proposed in order to improve the performance of the proposed algorithm. Furthermore, in order to balance the exploration and exploitation, a new linear increment scale factor is proposed. The proposed algorithm performs well on benchmark functions and then it is applied for practical dimensionality reduction of HSI. It can be seen from the results that the proposed algorithm is superior to state-of-art algorithms and it is helpful for solving the practical problems. The proposed algorithm cannot handle multi-model low-dimensional functions well. This will be the main work in the future.

Author Contributions: Conceptualization, J.-S.P.; data curation, Z.Z.; formal analysis, J.L. investigation, S.-C.C.; methodology, S.-C.C.; software, Z.Z.; supervision, J.-S.P.; writing—original draft, Z.Z.; writing—review and editing, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Natural Science Foundation of China (61872085).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Van Laarhoven, P.J.; Aarts, E.H. Simulated annealing. In *Simulated Annealing: Theory and Applications*; Springer: Berlin, Germany, 1987; pp. 7–15.
2. Tsai, C.W.; Hsia, C.H.; Yang, S.J.; Liu, S.J.; Fang, Z.Y. Optimizing hyperparameters of deep learning in predicting bus passengers based on simulated annealing. *Appl. Soft Comput.* **2020**, *88*, 106068. [\[CrossRef\]](#)
3. Grobelny, J.; Michalski, R. A novel version of simulated annealing based on linguistic patterns for solving facility layout problems. *Knowl.-Based Syst.* **2017**, *124*, 55–69. [\[CrossRef\]](#)
4. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [\[CrossRef\]](#)
5. Olivas, F.; Valdez, F.; Melin, P.; Sombra, A.; Castillo, O. Interval type-2 fuzzy logic for dynamic parameter adaptation in a modified gravitational search algorithm. *Inf. Sci.* **2019**, *476*, 159–175. [\[CrossRef\]](#)
6. Pelusi, D.; Mascella, R.; Tallini, L.; Nayak, J.; Naik, B.; Abraham, A. Neural network and fuzzy system for the tuning of Gravitational Search Algorithm parameters. *Expert Syst. Appl.* **2018**, *102*, 234–244. [\[CrossRef\]](#)
7. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [\[CrossRef\]](#)
8. Yang, Q.; Chu, S.C.; Pan, J.S.; Chen, C.M. Sine Cosine Algorithm with Multigroup and Multistrategy for Solving CVRP. *Math. Probl. Eng.* **2020**, *2020*, 1–10. [\[CrossRef\]](#)
9. Gupta, S.; Deep, K. A novel hybrid sine cosine algorithm for global optimization and its application to train multilayer perceptrons. *Appl. Intell.* **2020**, *50*, 993–1026. [\[CrossRef\]](#)
10. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 17 November–1 December 1995; Volume 4, pp. 1942–1948.
11. Yu, H.; Tan, Y.; Zeng, J.; Sun, C.; Jin, Y. Surrogate-assisted hierarchical particle swarm optimization. *Inf. Sci.* **2018**, *454*, 59–72. [\[CrossRef\]](#)
12. Sun, C.; Jin, Y.; Zeng, J.; Yu, Y. A two-layer surrogate-assisted particle swarm optimization algorithm. *Soft Comput.* **2015**, *19*, 1461–1475. [\[CrossRef\]](#)
13. Qin, S.; Sun, C.; Zhang, G.; He, X.; Tan, Y. A modified particle swarm optimization based on decomposition with different ideal points for many-objective optimization problems. In *Complex & Intelligent Systems*; Springer: Berlin, Germany, 2020; pp. 1–12.
14. Dorigo, M.; Di Caro, G. Ant colony optimization: A new meta-heuristic. In Proceedings of the IEEE 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1470–1477.
15. Chu, S.C.; Roddick, J.F.; Pan, J.S. Ant colony system with communication strategies. *Inf. Sci.* **2004**, *167*, 63–76. [\[CrossRef\]](#)
16. Pan, H.; You, X.; Liu, S.; Zhang, D. Pearson correlation coefficient-based pheromone refactoring mechanism for multi-colony ant colony optimization. In *Applied Intelligence*; Springer: Berlin, Germany, 2020; pp. 1–23.
17. Chu, S.C.; Tsai, P.W.; Pan, J.S. Cat swarm optimization. In *Pacific Rim International Conference on Artificial Intelligence*; Springer: Berlin, Germany, 2006; pp. 854–858.
18. Ahmed, A.M.; Rashid, T.A.; Saeed, S.A.M. Cat Swarm Optimization Algorithm: A Survey and Performance Evaluation. *Comput. Intell. Neurosci.* **2020**, *2020*, 4854895. [\[CrossRef\]](#)
19. Yang, X.S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [\[CrossRef\]](#)
20. Nguyen, T.T.; Pan, J.S.; Dao, T.K. A compact bat algorithm for unequal clustering in wireless sensor networks. *Appl. Sci.* **2019**, *9*, 1973. [\[CrossRef\]](#)
21. Cai, X.; Wang, H.; Cui, Z.; Cai, J.; Xue, Y.; Wang, L. Bat algorithm with triangle-flipping strategy for numerical optimization. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 199–215. [\[CrossRef\]](#)
22. Duan, H.; Qiao, P. Pigeon-inspired optimization: A new swarm intelligence optimizer for air robot path planning. *Int. J. Intell. Comput. Cybern.* **2014**, *7*, 24–37. [\[CrossRef\]](#)
23. Tian, A.Q.; Chu, S.C.; Pan, J.S.; Cui, H.; Zheng, W.M. A compact pigeon-inspired optimization for maximum short-term generation mode in cascade hydroelectric power station. *Sustainability* **2020**, *12*, 767. [\[CrossRef\]](#)
24. Abdullahi, M.; Ngadi, M.A. Symbiotic Organism Search optimization based task scheduling in cloud computing environment. *Future Gener. Comput. Syst.* **2016**, *56*, 640–650. [\[CrossRef\]](#)
25. Chu, S.C.; Du, Z.G.; Pan, J.S. Symbiotic organism search algorithm with multi-group quantum-behavior communication scheme applied in wireless sensor networks. *Appl. Sci.* **2020**, *10*, 930. [\[CrossRef\]](#)
26. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
27. Shieh, C.S.; Wang, H.Y.; Dao, T.K.; others. Enhanced diversity herds grey wolf optimizer for optimal area coverage in wireless sensor networks. In *International Conference on Genetic and Evolutionary Computing*; Springer: Berlin, Germany, 2016; pp. 174–182.
28. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the IEEE 2009 World congress on nature & biologically inspired computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
29. Cui, Z.; Zhang, M.; Wang, H.; Cai, X.; Zhang, W. A hybrid many-objective cuckoo search algorithm. *Soft Comput.* **2019**, *23*, 10681–10697. [\[CrossRef\]](#)
30. Song, P.C.; Pan, J.S.; Chu, S.C. A parallel compact cuckoo search algorithm for three-dimensional path planning. *Appl. Soft Comput.* **2020**, *94*, 106443. [\[CrossRef\]](#)

31. Gunen, M.A.; Besdok, E.; Civicioglu, P.; Atasever, U.H. Camera calibration by using weighted differential evolution algorithm: A comparative study with ABC, PSO, COBIDE, DE, CS, GWO, TLBO, MVMO, FOA, LSHADE, ZHANG and BOUGUET. *Neural Comput. Appl.* **2020**, *32*, 17681–17701. [\[CrossRef\]](#)
32. Meng, Z.; Pan, J.S. Monkey king evolution: A new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization. *Knowl. Based Syst.* **2016**, *97*, 144–157. [\[CrossRef\]](#)
33. Balasubramanian, D.; Govindasamy, V. Binary Monkey-King Evolutionary Algorithm for single objective target based WSN. *EAI Endorsed Trans. Internet Things* **2019**, *5*, 5. [\[CrossRef\]](#)
34. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [\[CrossRef\]](#)
35. Chu, S.C.; Xue, X.; Pan, J.S.; Wu, X. Optimizing ontology alignment in vector space. *J. Internet Technol.* **2020**, *21*, 15–22.
36. Chen, Y.H.; Huang, H.C.; Cai, H.Y.; Chen, P.F. A Genetic Algorithm Approach for the Multiple Length Cutting Stock Problem. In Proceedings of the 2019 IEEE 1st Global Conference on Life Sciences and Technologies (LifeTech), Osaka, Japan, 12–14 March 2019; pp. 162–165.
37. Zheng, Y.; Huang, M.; Lu, Y.; Li, W. Fractional stochastic resonance multi-parameter adaptive optimization algorithm based on genetic algorithm. In *Neural Computing and Applications*; Springer: Berlin, Germany, 2018; pp. 1–12.
38. Price, K.V. Differential evolution. In *Handbook of Optimization*; Springer: Berlin, Germany, 2013; pp. 187–214.
39. Sui, X.; Chu, S.C.; Pan, J.S.; Luo, H. Parallel Compact Differential Evolution for Optimization Applied to Image Segmentation. *Appl. Sci.* **2020**, *10*, 2195. [\[CrossRef\]](#)
40. Mousavirad, S.J.; Rahnamayan, S. Differential Evolution Algorithm Based on a Competition Scheme. In Proceedings of the 2019 14th International Conference on Computer Science & Education (ICCSE), Toronto, ON, USA, 19–21 August 2019; pp. 929–934.
41. Jin, C.; Tsai, P.W.; Qin, A.K. A Study on Knowledge Reuse Strategies in Multitasking Differential Evolution. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 1564–1571.
42. Pan, J.S.; Meng, Z.; Xu, H.; Li, X. QUasi-Affine TRansformation Evolution (QUATRE) algorithm: A new simple and accurate structure for global optimization. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*; Springer: Berlin, Germany, 2016; pp. 657–667.
43. Meng, Z.; Pan, J.S.; Xu, H. QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm: A cooperative swarm based algorithm for global optimization. *Knowl.-Based Syst.* **2016**, *109*, 104–121. [\[CrossRef\]](#)
44. Meng, Z.; Pan, J.S. QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm: The framework analysis for global optimization and application in hand gesture segmentation. In Proceedings of the 2016 IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, China, 6–10 November 2016; pp. 1832–1837.
45. Meng, Z.; Pan, J.S. A competitive QUasi-Affine TRansformation Evolutionary (C-QUATRE) algorithm for global optimization. In Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, Hungary, 9–12 October 2016; pp. 001644–001649.
46. Meng, Z.; Pan, J.S. QUasi-Affine TRansformation Evolution with External ARchive (QUATRE-EAR): An enhanced structure for differential evolution. *Knowl.-Based Syst.* **2018**, *155*, 35–53. [\[CrossRef\]](#)
47. Meng, Z.; Pan, J.S.; Lin, F. The QUATRE structure: An efficient approach to tackling the structure bias in differential evolution. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 3074–3079.
48. Liu, N.; Pan, J.S. A bi-population QUasi-Affine TRansformation Evolution algorithm for global optimization and its application to dynamic deployment in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 175. [\[CrossRef\]](#)
49. Du, Z.G.; Pan, J.S.; Chu, S.C.; Luo, H.J.; Hu, P. Quasi-affine transformation evolutionary algorithm with communication schemes for application of RSSI in wireless sensor networks. *IEEE Access* **2020**, *8*, 8583–8594. [\[CrossRef\]](#)
50. Sun, X.X.; Pan, J.S.; Chu, S.C.; Hu, P.; Tian, A.Q. A novel pigeon-inspired optimization with QUasi-Affine TRansformation evolutionary algorithm for DV-Hop in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 1550147720932749. [\[CrossRef\]](#)
51. Khanesar, M.A.; Teshnehlab, M.; Shoorehdeli, M.A. A novel binary particle swarm optimization. In Proceedings of the IEEE 2007 Mediterranean conference on control & automation, Guangzhou, China, 30 May–1 June 2007; pp. 1–6.
52. Rodrigues, D.; Pereira, L.A.; Almeida, T.; Papa, J.P.; Souza, A.; Ramos, C.C.; Yang, X.S. BCS: A binary cuckoo search algorithm for feature selection. In Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS), Beijing, China, 19–23 May 2013; pp. 465–468.
53. Too, J.; Mirjalili, S. A Hyper Learning Binary Dragonfly Algorithm for Feature Selection: A COVID-19 Case Study. *Knowl.-Based Syst.* **2020**, *212*, 106553. [\[CrossRef\]](#)
54. Hu, P.; Pan, J.S.; Chu, S.C. Improved Binary Grey Wolf Optimizer and Its application for feature selection. *Knowl.-Based Syst.* **2020**, *195*, 105746. [\[CrossRef\]](#)
55. Gupta, D.; Arora, J.; Agrawal, U.; Khanna, A.; de Albuquerque, V.H.C. Optimized Binary Bat algorithm for classification of white blood cells. *Measurement* **2019**, *143*, 180–190. [\[CrossRef\]](#)
56. Bostani, H.; Sheikhan, M. Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems. *Soft Comput.* **2017**, *21*, 2307–2324. [\[CrossRef\]](#)
57. Shen, X.; Liu, B.; Zhou, Y.; Zhao, J.; Liu, M. Remote sensing image captioning via Variational Autoencoder and Reinforcement Learning. *Knowl.-Based Syst.* **2020**, *203*, 105920. [\[CrossRef\]](#)

-
58. Basaeed, E.; Bhaskar, H.; Al-Mualla, M. Supervised remote sensing image segmentation using boosted convolutional neural networks. *Knowl.-Based Syst.* **2016**, *99*, 19–27. [[CrossRef](#)]
 59. Cui, B.; Cui, J.; Lu, Y.; Guo, N.; Gong, M. A Sparse Representation-Based Sample Pseudo-Labeling Method for Hyperspectral Image Classification. *Remote Sens.* **2020**, *12*, 664. [[CrossRef](#)]
 60. Mohanty, R.; Happy, S.; Routray, A. Spatial-Spectral Regularized Local Scaling Cut for Dimensionality Reduction in Hyperspectral Image Classification. *IEEE Geosci. Remote Sens. Lett.* **2018**, *16*, 932–936. [[CrossRef](#)]
 61. Cui, B.; Cui, J.; Hao, S.; Guo, N.; Lu, Y. Spectral-spatial hyperspectral image classification based on superpixel and multi-classifier fusion. *Int. J. Remote Sens.* **2020**, *41*, 6157–6182. [[CrossRef](#)]
 62. Kennedy, J.; Eberhart, R.C. A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997; Volume 5, pp. 4104–4108.
 63. Mirjalili, S.; Lewis, A. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. *Swarm Evol. Comput.* **2013**, *9*, 1–14. [[CrossRef](#)]
 64. Guo, S.S.; Wang, J.S.; Guo, M.W. Z-Shaped Transfer Functions for Binary Particle Swarm Optimization Algorithm. *Comput. Intell. Neurosci.* **2020**, *2020*, 6502807. [[CrossRef](#)] [[PubMed](#)]
 65. Liu, J.; Mei, Y.; Li, X. An analysis of the inertia weight parameter for binary particle swarm optimization. *IEEE Trans. Evol. Comput.* **2015**, *20*, 666–681. [[CrossRef](#)]
 66. Van der Maaten, L.; Postma, E.O.; van den Herik, H.J. Matlab Toolbox for Dimensionality Reduction. In *MICC*; Maastricht University: Maastricht, The Netherlands, 2007.
 67. Chang, C.C.; Lin, C.J. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol. (TIST)* **2011**, *2*, 1–27. [[CrossRef](#)]