Time-varying Binary Phasmatodea Population Evolution Algorithm

Jiayin Lou[1], Shu-Chuan Chu[2], Jeng-Shyang Pan[1,2,3], Zhongjie Zhuang[3]*

[1] College of Computer Science, Northeast Electric Power University, China
[2] Department of Information Management, Chaoyang University of Technology, Taiwan
[3] College of Computer Science and Engineering, Shandong University of Science and Technology, China
loujiayin76@126.com, scchu0803@gmail.com, jspan@cc.kuas.edu.tw, zhongjiezhuang@126.com

Abstract

Phasmatodea Population Evolution Algorithm (PPE) is an optimization algorithm based on insect behavior. It excels in solving tasks in continuous space. Standard PPE is not suitable for addressing binary problems such as path selection problems, neural network training, and feature selection problems. However, real-world binary problems cannot be solved by the original PPE algorithm. Because binary solutions can only have values of 0 or 1, while the solution space of standard PPE is continuous. To address these issues, we propose the Binary Phasmatodea Population Evolution (BPPE) algorithm and Time-Varying Binary Phasmatodea Population Evolution (TV-BPPE) for dealing with issues with binary properties and study the effect of different transfer functions on the algorithm's performance.

Keywords: Phasmatodea population evolution algorithm, Swarm intelligence, Transfer function, 0-1 Knapsack problem

1 Introduction

Meta-heuristic optimization algorithms are commonly utilized to tackle optimization problems [1]. They can be applied to engineering, business, economics, and finance. Particle swarm optimization (PSO) is the earliest-used meta-heuristic algorithm. Some scholars have applied differential Evolution [2], salient object detection [3], Communication Strategies [4], Pattern Recognition [5-6], and Dynamic regional splitting planning [7].

The Gray Wolf Optimizer (GWO), locust algorithm, and other algorithms are bionic algorithms that simulate the behavior of animals in nature. They are all members of the population-based meta-heuristic method, and their ease of use has drawn a lot of interest from scholars.

The 0-1 knapsack is a typical NP-hard problem. There are N items, and item i (i starts with 1) has quantity $n[i]$, weight $w[i]$, and value $v[i]$. As long as the total weight does not exceed the upper limit of the backpack W. Find the maximum value that can fit in the backpack. Due to the complexity of the actual situation, many researchers have to introduce various approximation algorithms to obtain near-optimal solutions.

Swarm intelligence is a part of the meta-heuristic which can solve 0-1 knapsack problems. PSO is a typical algorithm to simulate biological behavior. It simulates bird foraging and uses its behavior as a guide to finding the optimal solution. The Crow Search Algorithm (CSA) mimics crow foraging behavior [8-10]. GWO algorithm simulates the hunting behavior of wolves [11]. The Ant Colony Optimization (ACO) simulates the process of ants foraging [12-16]. The Cat Swarm Optimization (CSO) algorithm simulates cats searching for prey [17-20]. The Artificial Fish Swarm Algorithm (AFSA) simulates fish swimming [21-24]. Phasmatodea Population Evolution (PPE) [25-27] is an optimization algorithm based on Phasmatodea population evolution and predation behavior.

PPE is a new population intelligence algorithm proposed recently. In real society, PPE is mainly used to deal with continuity problems, and there are many areas of optimization, such as image segmentation, elevator group control, application offloading decision, and transportation planning. However, many real-world situations are discrete, such as the path planning problem, the 0-1 knapsack problem [28-30], and the traveling salesman problem. Because meta-heuristic algorithms are typically only capable of solving continuous problems, corresponding binary versions are required to address these discrete difficulties [31-42]. The combination of the time-varying function and transfer function has been proved in several papers to improve the performance of the transfer function effectively [43-47].

In this paper, we mainly study the ability of plane binary phasmatodea population evolution (BPPE) to find the optimal solution and its application. The process of finding the optimal solution is divided into two stages: exploration and exploitation. In the exploration stage, the algorithm will explore to find a new space as possible. In the exploitation stage, the PPE proposes to first search for local optimal solutions in multiple domains. Multiple local optimal solutions are compared to determine the optimal solution. This proposal effectively solves the problem of the PPE repeating the search during the search.

The output of an object with binary properties and the input of an object with continuous properties are represented by a function called the binary transfer function. If x is the evolutionary trend of the population. The binary transfer function is the function that determines the range of x values. It maps the value of x to [0,1]. The

*Corresponding Author: Jeng-Shyang Pan; E-mail: jspan@cc.kuas.edu.tw
DOI: https://doi.org/10.70003/160792642025012601003

binary transfer function can map the solution x from the continuous space to the binary space. Because the transfer function transformation equation is related to the growing trend of parameter population. Therefore, the transfer function also has an important effect on the performance of the binary optimization algorithm in the process of binary conversion.

We add a transfer function to the standard PPE to transform it into a binary version. The impact of four different transfer functions on the improvement of the binary algorithm is examined in this work. The proposed BPPE's performance is evaluated using 23 benchmark functions. According to experiment results, the BPPE algorithm is more capable of global optimization. BPPE is further enhanced by incorporating time-varying functions, and a Time-Varying Binary Phasmatodea Population Evolution algorithm (TV-BPPE) is proposed. Then, we apply TV-BPPE to solve the 0-1 knapsack selection problem and compare the solutions for the problems in 20 dimensions, 40 dimensions, 50 dimensions, and 100 dimensions. The experimental results show that TV-BPPE performs well when dealing with high-dimensional binary situations.

To find a high-performance transfer function to improve the quality of the solution. This paper attempts to optimize the binary transfer function by using a time-varying function.

The main contributions of this paper are as follows:

(1) Proposed binary version of PPE (BPPE).

(2) The experimental effects of the four transfer functions are analyzed through the benchmark function.

(3) Based on the result of (2), the time-varying function is added to form a Time-Varying BPPE (TV-BPPE).

(4) The 0-1 knapsack application is implemented based on BPPE and TV-BPPE.

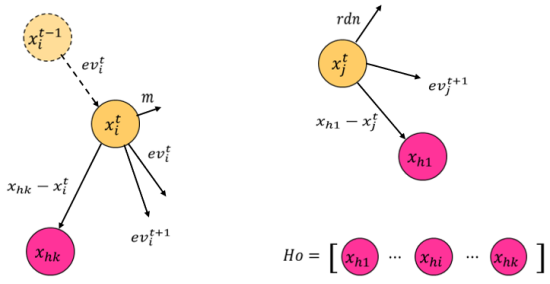


Figure 1. Population evolution trend

2 Related Works

This section introduces the standard PPE and the evolution process of PPE.

The algorithm imitates the living habits of phasmatodea. The PPE algorithm approximates each problem to be solved as a virtual population. The virtual population has two properties: population size and growth rate. Population size is affected during algorithm development and exploration. This algorithm consists of two conceptual models, namely, the similarity evolution model and the competition model. The competition model is affected by two factors, the size of the population and the migration of the population. In terms of population space size, larger populations tend to be able to explore more space and have the potential to explore harsher environments. But populations can explore harsh environments. In terms of population migration, the process of population migration has an impact on the population growth rate. If the population moves to a better environment, the population growth rate will increase and vice versa. The similarity evolution model is to show that because the population is geographically close and living in a similar environment, they have also undergone similar changes. So populations evolve as they explore new environments.

The population growth of this algorithm is closely related to population size. As the population increases, the growth rate k gradually decreases until it no longer increases. In addition, the effects of competition and mutation are considered. Under the competition condition, the population determines whether to increase according to the development condition and the size of the critical value.

During the evolution process, different influencing factors will lead to the evolution of the population in different development directions. Both population size and growth rate have guiding effects on population development. This evolution mode has better performance in obtaining optimal solutions.

In the initialization stage of the population, In the PPE algorithm, x represents the solution of the problem and represents a virtual Phasmatodea population. Initialize the population size to Np, A total of Np solutions are generated. Use Equation (1) to initialize the population size.

$$p_i = \frac{1}{Np} \tag{1}$$

There are three factors affecting the population growth trend, which are respectively natural growth factors, population competition factors, and mutation factors.

In Equation (2), a represents the growth rate. During the development of the population, certain boundary constraints will be imposed on the value of a. After the population grows to a certain level. If the growth trend of the population is too large or too small, the PPE will be recalculated. The updated population growth rate is 1.1, and the population growth trend ev is updated to 0. A represents the value obtained by multiplying the local optimal solution by the specified coefficient in the environment of natural growth. When the value of population growth trend A is less than 0.1 or greater than 4, Equation (2) will be used to update the population instead of the original solution. The value of a is between 0 and 4. When a exceeds the interval [0-4], it has no meaning. If $a > 3$, the population will become unstable and go into chaos. If $a<1$, it means that the population p gradually

decreases. Eventually, the population will decrease to 0 and the population will disappear. If $1<a<3$, the population will slowly converge to a stable value $p=(a-1)a$. So the value range of a is (0, 4). The pseudo-code of PPE is presented in Algorithm 1.

$$p^{t+1} = ap^t(1-p^t) \tag{2}$$

Algorithm 1. PPE

```
Initialize Np solutions;
Initialize ev, p, k, and a use Equations (1)-(3);
Calculate fitness f(x), set gbest and Ho;
for t = 2 to Maxgen do
    Update each x to newx using Equation (4);
    Calculate new fitness newx, update gbest and Ho;
    for i = 1 to Np do
      if f (newx) ≤ f (x) then
      Update x = newx, update f (x);
      Update p_i use Equation (5);
      Update ev_i use Equation (6) and (8);
    else
    If r_d < p_i then
      Update x = newx, update f (x);
      Update pi use Equation (5);
      Update evi use Equation (7);
    Randomly choose a solution x_j , (j ≠ i);
    if dist(x_j , x_i ) < G then
      Update p_i use Equation (9),
      Update ev_i use Equation (10);
```

The algorithm starts iterating after initialization. k is used to store the number of historical optimal solutions. If a total of Np solutions are generated, the number of k is calculated by Equation (3). Ho is used to represent the set of locally optimal solutions. Ho makes the set of optimal solutions $Ho=[x_{h1},...,x_{hi},...,x_{hk}]$, as shown in Figure 1. where x_{hi} is the i-th optimal solution that has appeared, and x_{h1} is the optimal solution of all existing solutions. The optimal solution is found according to the local optimal method. The new solution can exist in a variety of circumstances. So, there are three kinds of population update situations for the PPE.

The first case converges the population to the nearest optimum. The location update formula uses Equations (4) - (5) to update the population number and uses Equation (6) to update the species evolution trend. t is the current iteration number. m represents the growth trend under the competition state.

$$k = \log(Np) + 1 \tag{3}$$

$$x_t + 1 = x_t + ev \tag{4}$$

$$p^{t+1} = a^{t+1}p^t(1-p^t) \tag{5}$$

$$ev^{t+1} = (1-p^{t+1})A + p^{t+1}(ev^t + m) \tag{6}$$

The second case is that after the population has moved, the fitness of the moving population is worse than before. The evolutionary trend is calculated using Equation (7). A_1 represents the degree of closeness to the closest optimal value. B represents an n-dimensional random vector generated using the standard normal distribution. st is initially set to $(Ub-Lb)\times 0.1$. A_1 represents the degree of approach to the nearest optimal as shown in Equation (8). $s(Ho, x_t)$ is used to find the historical optimal solution that is closest to x_t in Ho. c is the coefficient of the influence degree of the nearest optimal solution on the current population, usually less than 1.

$$ev^{t+1} = rand \cdot A + st \cdot B \tag{7}$$

$$A_1 = (s(Ho, x^t) - x^t) \cdot c \tag{8}$$

The third case is the renewal mode of Phasmatodea population under a simulated competitive state. The population number is calculated using Equation (9). p_i and p_j represent the populations of two competing populations. a_i represents the population growth rate of the current population. Equation (10) is used to calculate the population evolution trend under the competitive state.

$$p_i = p_i + a_i p_i(1 - p_i - \frac{f(x_j)}{f(x_i)} p_j) \tag{9}$$

$$ev^{t+1} = ev^{t+1} + \frac{f(x_j) - f(x_i)}{f(x_j)}(x_j - x_i) \tag{10}$$

In Algorithm 1, $gbest$ is the global optimal solution. $Maxgen$ is the maximum number of iterations. $newx$ is the updated position information of x. $dist(xj, xi)$ represents the distance between two competing populations.

3 The Proposed Algorithm

The standard PPE algorithm only can solve continuous problems well. However, BPPE can solve discrete optimization problems with binary characteristics. For example, whether to take 1 or not can be used to indicate the connection of the neural network. 1 indicates that it is connected to the last two nodes and 0 indicates that it is disconnected from the last two nodes. Therefore, the BPPE algorithm can be used to optimize the neural network structure. In standard PPE, it is located at any point in continuous space. Therefore, the modified equation is easy to implement.

In particle swarm optimization, the velocity of a particle is usually used to determine the probability of a particle being in position 0 or 1. Think of the search space as a hypercube where the position of particles is limited to 0 or 1 on each dimension of the search space. So the standard PPE equation does not satisfy this condition. We propose a binary phasmatodea population evolution

algorithm (BPPE) to solve the binary problem and describe the algorithmic structure of BPPE in detail.

3.1 PPE Algorithm for Binary (BPPE)

Similar to PPE, the initial population growth rate in BPPE is set as k. Using Equation (1), Equation (2) generates the initial population. The different development cases are resolved and divided into three cases. In the first case, the population converges and gets a good value after the position update, and evolves towards the nearest optimal. The value of x is updated using Equation (4). If the new optimal solution is worse than the last optimal solution, Equation (5) was used to update the method of obtaining pi, and Equation (7) was used to update the method of solving evi. The calculation method of A can be obtained from Equation (8). With the development of search space competition and optimization behavior, the position of the optimal solution will change. If competition occurs and dist value is less than G, pi and $e$vi are updated with Equations (9) and (10). The pseudo-code of BPPE is presented in Algorithm 2.

$$x = \begin{cases} 0 & if \quad rand() \le 0.5 \\ 1 & else \quad rand() > 0.5 \end{cases} \tag{11}$$

$$x = \begin{cases} 1 & if \quad ev > 0 \\ 0 & if \quad ev \le 0 \end{cases} \tag{12}$$

Algorithm 2. BPPE

Initialize *Np* solutions using Equation (11)
Initialize *ev*, *p*, *k*, and *a* use Equation (1)-(3);
Calculate fitness f(x), set *gbest* and *Ho*;
for t = 2 to *Maxgen* do
  Update each *x* to *newx* using Equation (12);
  Calculate new fitness *newx*, update *gbest* and *Ho*;
  for i = 1 to Np do
    if f (*newx*) ≤ f (x) then
    Update *x* = *newx*, update f (x);
    Update p_i using Equation (5);
    Update *evi* use Equation (6) and (8);
  else
  if $r_d < p_i$ then
    Update *x* = *newx*, update f (x);
    Update p_i using Equation (5);
    Update ev_i use Equation (7);
  Randomly choose a solution x_j , ($j \neq$ i);
  if dist(*xj* , *xi*) < G then
    Update *pi* use Equation (9),
    Update *evi* use Equation (10);

The initial position x needs to be represented by a binary number. Use Equation (11) to initialize the value of x to a binary number. x is initialized by the random function to generate a random number between (0,1). Compare the generated random number to 0.5. If the random number is less than 0.5, the x value is 0. If the random number is greater than 0.5, then x is 1.

Equation (12) is used to calculate the value of x after iteration. The value of x is related to ev, which is 0 or 1. If $ev > 0$, then x is equal to 1, otherwise x is equal to 0.

3.2 Analysis of the Behavior of existing Transfer Functions

A common way to convert binaries is to use a transfer function. It works by mapping continuous values to [0,1] and then discrete them into 0 and 1 based on probability.

This section used four types of transfer functions to improve the standard PPE. Each type of transfer function is described in detail. Four common transfer functions of each type are selected to improve the algorithm. A total of 16 transfer functions were tested. The performance of these transfer functions is compared across 23 benchmark functions. And analyzes their influencing factors and performance. In the improvement of the binary algorithm, some scholars add time tracking to the binary algorithm. Through reading literature, the better effect of time-varying factors on binary was confirmed.

3.2.1 Sigmoid Transfer Function

This section shows four sigmoid functions, the expression details of the four functions are shown in Table 1, and Figure 2 and Figure 3 show the curves formed by the functions

Table 1. The expressions of the S-shaped families of transfer functions

Name	Transfer function
S_1	$T(x) = \frac{1}{(1+e^{-2x})}$
S_2	$T(x) = \frac{1}{(1+e^{-x})}$
S_3	$T(x) = \frac{1}{(1+e^{-x/2})}$
S_4	$T(x) = \frac{1}{(1+e^{-x/3})}$

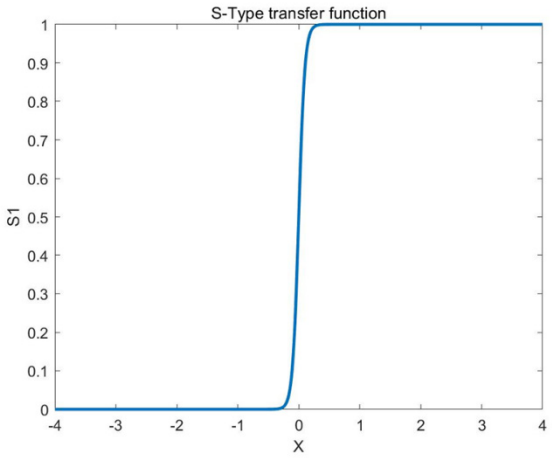


Figure 2. S-types transfer function

If BPPE uses S_1 as the transfer function, the value of $S_1(x)$ increases as the value of x increases. As shown in Figure 2, the sigmoid function maps x to a number at [0,1]. It can be seen that the sigmoid function is applicable when the absolute value of x is small in the early stage, which is conducive to the better exploration of search space in the early stage.

It can be seen from Figure 3 that the convergence conditions formed by the four different sigmoid functions are different, but the trend is roughly the same. Where $x = 0.5$ is the critical value, if $x = 0.5$, then $S_1(x) = 0.5$; when x is equal to 0.5, BPPE has a 50% probability of becoming 0; if $x < 0.5$, then its probability of becoming 0 is higher than large; if $x > 0.5$, the probability of it becoming 0 is small.

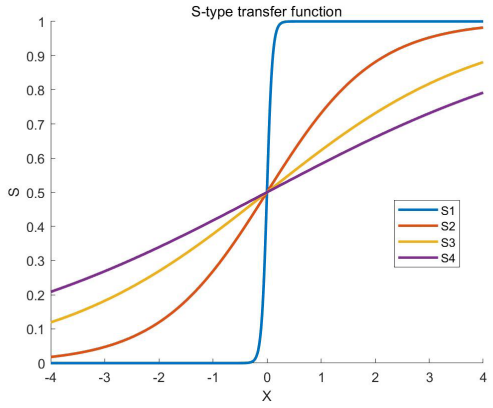


Figure 3. The curve of S_1, S_2, S_3, S_4

3.2.2 V-shaped Transfer Functions

This section shows four V-type functions, the expression details of the four functions are shown in Table 2, and Figure 4 shows the curves formed by the functions.

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

Name	Transfer function
$V_1(x)$	$\left\|T(x) = erf\left(\frac{\sqrt{\pi}}{2}x\right)\right\|$
$V_2(x)$	$\left\|T(x) = \tanh(x)\right\|$
$V_3(x)$	$\left\|T(x) = x/\sqrt{1+x^2}\right\|$
$V_4(x)$	$\left\|T(x) = \frac{2}{\pi}\arctan\left(\frac{2}{\pi}x\right)\right\|$

As shown in Figure 4, in the function curve of the four V-shaped functions, when x is less than -1 and greater than 1, the probability of the position taking 1 exceeds 0.5. The values of different types of V-shaped functions are different. In the interval from -1 to 1, the closer the value of x is to 0, the greater the probability of the position being 0. The function formulas of the four V-type functions differ greatly. The graph of the function is symmetric about $x =$ 0.

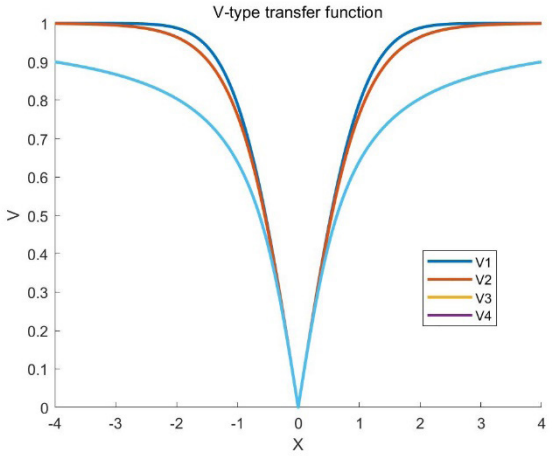


Figure 4. The curve of V_1, V_2, V_3, V_4

3.2.3 U-shaped Transfer Functions

This section introduces 4 types of the U-type transfer function. Plot the U-shaped transfer function using Equations (13) and (14). Figure 5 shows the Convergence curve.

$$U(x) = \alpha\left|x^{\beta}\right| \tag{13}$$

$$x_i^k(t+1) = \begin{cases} -x_i^k(t) & r < U(u_i^k(t+1)) \\ x_i^k(t) & r \geq U(u_i^k(t+1)) \end{cases} \tag{14}$$

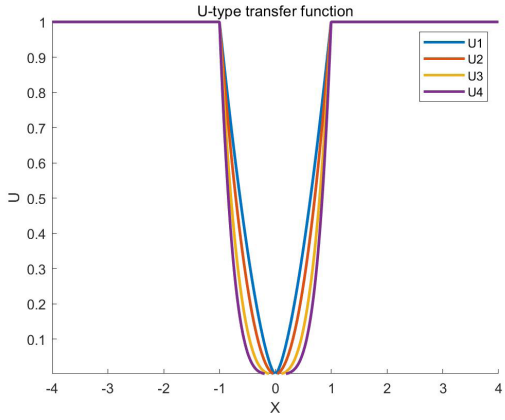


Figure 5. The curve of U_1, U_2, U_3, U_4

The inflection point of the curve represents the point of full saturation, and the return value is 1 which indicates the highest exploration volume. When the value of x is greater than 1 or less than -1, the value of the position will be 1. The value of y only exceeds 0.5 when the value of x is very close to 1 or -1. It follows that the greater the absolute value of x, the greater the probability of taking 1.

The shape of this transfer function allows for higher exploration and is ideal for solution mining by the PPE algorithm.

3.2.4 Z-shaped Transfer Functions

This section presents four Z-type transfer functions.

The functions' details are in Table 3, and the curve formed by the function is shown in Figure 6. This type of function takes 1 when the value of x is greater than 0, and the probability of the position value taking 1 when the value of x is smaller is greater. The population evolutionary trend is represented by x. The range of x is x less than 0. If the population trend value is greater than 0, it is taken as 0.

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

Name	Transfer function
$Z_1(x)$	$T(x) = \sqrt{1-2^x}$
$Z_2(x)$	$T(x) = \sqrt{1-5^x}$
$Z_3(x)$	$T(x) = \sqrt{1-8^x}$
$Z_4(x)$	$T(x) = \sqrt{1-20^x}$

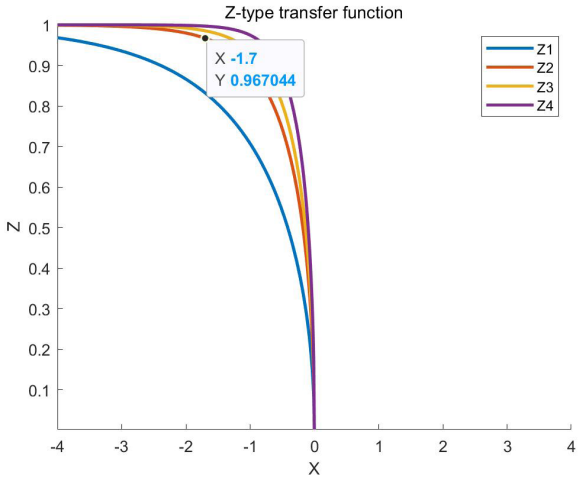


Figure 6. The curve of Z_1, Z_2, Z_3, Z_4

4 Time-varying Transfer Function-based BPPE (TV-BPPE)

This section describes how BPPE combines with the time-varying function and how the time-varying function works. The time-varying function will be added to four types of transfer functions for use and to test algorithm performance. The solution process of Time-Varying BPPE (TV-BPPE) will be introduced in detail.

4.1 Design Considerations for the Time-varying Transfer Function

As shown in Figure 7, the curve is a sigmoid time-varying transfer function that is connected by several discrete points where to evaluate x every 0.1 units. The Y-axis represents the result of the binary conversion. Correspondingly, the inversion of the position also increases with the slope of the curve.

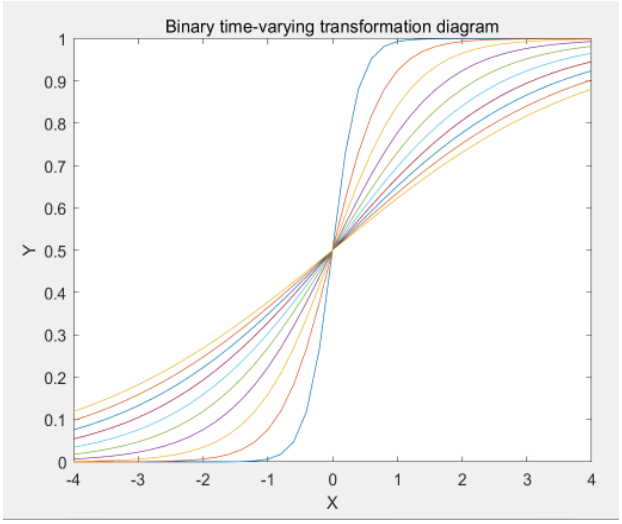


Figure 7. An illustration of different shapes of the time-varying transfer function (Equation (15)) with different values of the control parameter ϕ

The equation of the S-type function is shown as Equation (15). The variable x in the S-type equation will be replaced. The value of x is controlled by ψ to make it a non-fixed value, and the value of ψ is based on the formula (16). x is going to range from -4 to 4. Itr_{k+1} represents the k iteration and Itr_{max} represents the total number of iterations where k = 0, 1, 2, . . ., Itr_{k+1} − 1. In this paper, the total number of iterations of Itr_{max} is 500. The value of φ_{max} is -5, and the value of φ_{min} is 5. The pseudo-code of TV-BPPE is presented in Algorithm 3.

$$TV_T(V_{id}^{k+1}, \varphi) = \frac{1}{1+e^{-\frac{v_{id}^{k+1}}{\varphi}}} \tag{15}$$

$$\varphi = \varphi_{\max} - Itr_{k+1} * \left(\frac{\varphi_{\max} - \varphi_{\min}}{Itr_{\max}} \right) \tag{16}$$

Algorithm 3. TV-BPPE

Initialize *Np* solutions using Equation (11)
Initialize *ev*, *p*, *k*, and a use Equations (1)-(3);
Calculate fitness f(x), set *gbest* and *Ho*;
for t = 2 to *Maxgen* do
    calculate ϕ using Equation (18)
    update x_i using Equation (19), (20)
  Update each *x* to *newx* use Equation (14) and (15);
  Calculate new fitness *newx*, update *gbest* and *Ho*;
  for i = 1 t o *Np* do
    if f (newx) ≤ f (x) then
    Update *x* = newx, update f (x);
    Update p_i use Equation (5);
    Update ev_i use Equations (6) and (8);
  else
  if *rd* < *pi* then
    Update *x* = *newx*, update f (x);
    Update p_i using Equation (5);
    Update ev_i use Equation (7);

Randomly choose a solution x_j, ($j \neq i$);
if dist(x_j , x_i) < G then
Update p_i using Equation (9).

Compared with the traditional fix-shaped function, the proposed scheme has the following benefits.

1. For each given x value, the time-varying function curve provides a higher flip probability. This function can provide the minimum flip probability of the position of the particle x.

2. The binary transfer function of time-varying function (TV-b) converges quickly, which can quickly find the optimal solution in the process of exploration.

3. TV-b can find higher quality solutions in the later development process.

5 Experimental Results and Analysis

5.1 BPPE Experimental Results

This section tests the BPPE with 23 benchmark functions. The tests were divided into four groups. Four types of transfer functions are used to improve the PPE. Each type is tested as a group. Each type has four transfer functions.

The 23 test functions including Table 4 describe the unimodal functions. Table 5 describes the multimodal functions and Table 6 describes the Fixed-dimension benchmark functions.

Experimental data are rounded to four decimal places for ease of reading. The main reference indexes of the experimental results are the mean value followed by the optimal value and the median. We compared the results of running 500 iterations on each of BPPE's four transfer functions.

The unimodal function can verify the convergence speed of the function which is only used for optimal solutions.

Multimodal functions can verify whether the algorithm can avoid falling into local traps. Due to the complexity of its structure, composite functions can test the performance of functions in many aspects.

Space represents the boundary of the search space. *Dim* represents the dimension of the function, which the value in this paper is the same as the number of populations, and *fmin* represents the optimal value.

5.2 BPPE Experimental Analysis

This section analyzes the binary conversion method in detail. The performance of different binary algorithms is analyzed. The quality of the binary algorithm is compared by testing. The BPPE functions are compared with the other four binary algorithms on the test functions. The five functions tested are Binary Phasmatodea Population Evolution (BPPE), Binary Differential Evolution (BDE), Binary Fish Migration Optimization (BFMO), Binary Particle Swarm Optimization (BPSO), and Binary Grey Wolf Optimizer (BGWO). They were tested using 23 benchmark functions.

The 23 benchmark functions contain many function forms, which can better test the performance of the algorithm. It contains three kinds of functions which are unimodal, multimodal, and complex functions. Table 4 is the unimodal function. It has a global optimal solution and no local trap, so it can verify the convergence speed of the algorithm. Table 5 is multimodal function and Table 6 is complex functions respectively. The multimodal function can judge whether the algorithm avoids falling into a local trap. The composite multimodal function has a complex structure with many random global optimal solutions.

Table 7 shows the binary PPEs under the S-type transfer function. Table 8 shows the binary PPEs under the V-type transfer function. Table 9 shows the binary PPEs under the U-type transfer function. And Table 10 shows the binary PPEs under Z-type transfer function.

Table 4. Unimodal benchmark functions

Name	Function	Space	D_{im}	$f_{\min}$
Sphere	$f_1(x)=\sum_{i=1}^{n}x_i^2$	[-100,100]	30	−12 569
Schwefel's Function2.21	$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[-10,10]	30	0
Schwefel's Function1.2	$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j-1}^{n}x_j\right)^2$	[-100,100]	30	0
Schwefel's Function2.22	$f_4(x)=\max_{i\{\lvert x_i\rvert,1\le i\le n\}}$	[-100,100]	30	0
Rosenbrock	$f_5(x)=\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	[-30,30]	30	0
Step	$f_6(x)=\sum_{i=1}^{n}\left(\left[x_i+0.5\right]\right)^2$	[-100,100]	30	0
Dejong's noisy	$f_7(x)=\sum_{i=1}^{n}ix_i^4+\mathrm{Random}\left[0,1\right)$	[-1.28,1.28]	30	0

Table 5. Multimodal benchmark functions

Name	Function	Space	D_{im}	$f_{\min}$
Schwefel	$f_8(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	[−500,500]	30	−12 569
Rastringin	$f_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	[−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	$f_{12}(x)=\frac{\pi}{x}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\right\}$	[−50,50]	30	0
Generalized penalized 2	$f_{13}(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	[−50,50]	30	0

Table 6. Fixed-dimension benchmark functions

Name	Function	Space	D_{im}	$f_{\min}$
Fifth of Dejong	$f_{14}(x)=\left(\frac{1}{500}\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right)^{-1}$	[−65, 65]	2	1
Kowalik	$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_i\left(b_i^2+b_ix_2\right)}{b_i^2+b_ix_3+x_4}\right]^2$	[−5, 5]	4	0.00030
Six-hump camel back	$f_{16}(x)=4x_i^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	[−5, 5]	2	−1.0316
Brains	$f_{17}(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$	[−5, 5]	2	0.398
Goldstein –Price	$f_{18}(x)=[1+(x_1+x_2+1)^2(19-14x_2+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	[−2, 2]	2	3
Hartman 1	$f_{19}(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{3}aij(x_j-p_{ij})^2)$	[1, 3]	3	−3.86
Hartman 2	$f_{20}(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}aij(x_j-p_{ij})^2)$	[0, 1]	6	−3.32
Shekel 1	$f_{21}(x)=-\sum_{i=1}^{5}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	[0, 10]	4	−10.1532
Shekel 2	$f_{22}(x)=-\sum_{i=1}^{7}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	[0, 10]	4	−10.4028
Shekel 3	$f_{23}(x)=-\sum_{i=1}^{10}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	[0, 10]	4	−10.5363

Table 7. Statistical results of the transfer function in the original equation of the S-shaped function

	S_1		S_2		S_3		S_4	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	**6.2800**	0.9161	6.2940	0.9324	6.3720	0.9313	6.3920	0.9099
F2	**6.1200**	0.8940	6.2860	0.8864	6.3140	0.9408	6.3840	0.8798
F3	**361.2500**	111.5523	371.9880	103.0235	375.3380	106.4106	384.3260	103.9238
F4	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000
F5	**610.3440**	88.1663	624.5060	86.1236	630.4660	84.9715	620.7600	86.1863
F6	**19.8920**	1.7879	20.0880	1.8212	20.0280	1.8374	20.2800	1.8413
F7	**76.8208**	13.5761	78.3223	14.9077	80.4258	14.2327	82.6748	14.4170
F8	**12549.2900**	0.7467	12549.4562	0.7570	12549.5706	0.7088	12549.5016	0.7898
F9	**6.1020**	0.8816	6.2460	0.9248	6.2600	0.9022	6.3260	0.9303
F10	**1.7239**	0.1319	**1.7289**	0.1301	1.7491	0.1289	1.7463	0.1342
F11	0.2072	0.0317	0.2080	0.0357	0.2096	0.0343	**0.2066**	0.0352
F12	**2.6562**	2.6704	2.6805	0.1671	2.6971	0.1817	2.7158	0.1676
F13	**0.6142**	0.0946	0.6286	0.0913	0.6380	0.0991	0.6360	0.0888
F14	**11.6705**	1.7781	**11.6705**	1.7781	**11.6705**	0.0000	**11.6705**	0.0000
F15	**0.1481**	3.3340	**0.1481**	3.3340	**0.1481**	0.0000	**0.1481**	0.0000
F16	**1.0316**	1.0316	**1.0316**	2.4449	**1.0316**	0.0000	**1.0316**	0.0000
F17	**27.3049**	3.5563	**27.3049**	3.5563	**27.3049**	0.0000	**27.3049**	0.0000
F18	**597.0000**	0.0000	**597.0000**	0.0000	**597.0000**	0.0000	**597.0000**	0.0000
F19	**3.5252**	1.0669	**3.5252**	1.0669	**3.5252**	0.0000	**3.5252**	0.0000
F20	**3.1543**	5.3344	**3.1543**	5.3344	**3.1543**	0.0000	**3.1543**	0.0000
F21	**4.9448**	8.8907	**4.9448**	8.8907	**4.9448**	0.0000	**4.9448**	0.0000
F22	**4.9123**	8.8907	**4.9123**	8.8907	**4.9123**	0.0000	**4.9123**	0.0000
F23	**4.8715**	4.4453	**4.8715**	8.8876	**4.8715**	0.0000	**4.8715**	0.0000

5.2.1 Experimental Analysis of S-type Binary Functions

AVG represents the average of 500 results. *STD* is the variance of the function. In Table 7, F_4, $F_{14\text{-}23}$, function S_1, S_2, S_3, S_4 achieves the same optimal value.

The test results show that the S_1-type function achieves the best result among the 22 benchmark functions. The S_2-type function achieves the best result among the 12 benchmark functions. The S_3-type function achieves the best result among the 11 benchmark functions. The S_4-type function achieves the best result among the 12 benchmark functions.

In summary, there is a big difference between the results of 23 benchmark functions calculated by four functions. S_1-type works best among the 23 benchmark functions.

5.2.2 Experimental Analysis of V-type Binary Functions

Table 8 shows the operation results of the V-type function on 23 test functions. F_1, F_2, F_3, F_8, F_9, $F_{14\text{-}23}$ function V_1, V_2, V_3, V_4 achieves the same optimal value.

The test results show that the V_1-type function achieves the best result among the 18 benchmark functions. The V_2-type function achieves the best result among the 19 benchmark functions. The V_3-type function achieves the best result among the 16 benchmark functions. The V_4-type function achieves the best result among the 19 benchmark functions.

In summary, the four transfer functions of V-type perform roughly the same across the 23 benchmark functions, with V_1 performing the best.

5.2.3 Experimental Analysis of U-type Binary Functions

Table 9 shows the operation results of the U-type function on 23 test functions. $F_{1\text{-}4}$, F_6, $F_{8\text{-}19}$, F_{21}, F_{23} function U_1, U_2, U_3, U_4 achieves the same optimal value.

The test results show that the U_1-type function achieves the best result among the 22 benchmark functions. The U_2-type function achieves the best result among the 20 benchmark functions. The U_3-type function achieves the best result among the 21 benchmark functions. The U_4-type function achieves the best result among the 19 benchmark functions.

Table 8. Statistical results of the transfer function in the original equation of the V-shaped function

	V_1		V_2		V_3		V_4	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F2	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F3	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F4	0.0220	0.1468	0.0060	0.0773	**0.0000**	0.0000	0.0080	0.0892
F5	**3.3640**	9.2958	6.1100	15.3237	17.1000	15.5906	27.6900	7.8576
F6	**7.5040**	0.0894	7.5160	0.2187	7.5160	0.2187	7.5160	0.1783
F7	0.0027	0.0028	**0.0023**	0.0021	0.0024	0.0023	**0.0023**	0.0025
F8	**12544.2559**	0.0000	**12544.2559**	0.0000	**12544.2559**	3.6416	**12544.2559**	0.0000
F9	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F10	0.0029	0.0453	**0.0014**	0.0321	8.8818	0.0000	8.8818	0.0000
F11	0.0007	0.0095	0.0008	0.0107	0.0006	0.0097	**0.0000**	0.0000
F12	1.6809	0.0393	1.6772	0.0332	1.6731	0.0229	**1.6724**	0.0209
F13	**0.0000**	0.0000	**0.0000**	0.0000	0.0038	0.0191	0.0614	0.0581
F14	**11.6705**	0.0000	**11.6705**	0.0000	**11.6705**	1.7781	**11.6705**	0.0000
F15	**0.1481**	0.0000	**0.1481**	0.0000	**0.1481**	3.3340	**0.1481**	0.0000
F16	**1.0316**	0.0000	**1.0316**	0.0000	**1.0316**	2.4449	**1.0316**	2.4449
F17	**27.3049**	0.0000	**27.3049**	0.0000	**27.3049**	3.5563	**27.3049**	0.0000
F18	**597.0000**	0.0000	**597.0000**	0.0000	**597.0000**	0.0000	**597.0000**	0.0000
F19	**3.5252**	0.0000	**3.5252**	0.0000	**3.5252**	1.0669	**3.5252**	1.0669
F20	**3.1549**	0.0049	**3.1553**	0.0062	**3.1545**	0.0030	**3.1549**	0.0050
F21	**4.9448**	0.0000	**4.9448**	0.0000	**4.9448**	0.0000	**4.9448**	0.0000
F22	**4.9123**	0.0000	**4.9207**	0.1868	**4.9123**	0.0000	**4.9123**	0.0000
F23	**4.8715**	0.0000	**4.8715**	0.0000	**4.8715**	0.0000	**4.8715**	0.0000

The test results of the U-shaped function are consistent with the above analysis of the function image. U-shaped functions have the advantage of binary improvements to the PPE. The four U-shaped transfer functions perform well in the test functions. In summary, U_3-type works best among the 23 benchmark functions.

5.2.4 Experimental Analysis of Z-type Binary Functions

Table 10 shows the operation results of Z-type function on 23 test functions. F_3, F_9, $F_{14\text{-}19}$, F_{23} function Z_1, Z_2, Z_3, Z_4 achieves the same optimal value.

The test results show that the Z_1-type function achieves the best result among the 19 benchmark functions. The Z_2-type function achieves the best result among the 11 benchmark functions. The Z_3-type function achieves the best result among the 11 benchmark functions. And the Z_4-type function achieves the best result among the 14 benchmark functions. In summary, Z_1-type and Z_4-type work best among the 23 benchmark functions.

5.2.5 Experimental Analysis of Four Types of Binary Functions

Table 11 shows the Comparison of Statistical Results of Transfer Functions in Original Equations of U, V, S, Z. The most effective of the four types of transfer functions are S_1-type, V_3-type, U_1-type, and Z_1-type.

The test results show that the S_1-type function achieves the best result among the 10 benchmark functions. The V_3-type function achieves the best result among the 16 benchmark functions. The U_1-type function achieves the best result among the 23 benchmark functions. The Z_1-type function achieves the best result among the 17 benchmark functions. Among the 23 benchmark functions, U_1-type has the strongest ability to obtain the optimal solution. S_1-type has the worst ability to obtain the optimal solution.

The improvement effect of the S-type on the PPE algorithm is quite different from that of the other three types. S-type is not suitable for improving the PPE

algorithm. It is worth noting that the U_1-type transfer function of the PPE algorithm has obtained the optimal values in all the benchmark functions.

5.3 Comparative Analysis of Binary PPE and Other Binary Functions

BPPE(U_1) and four other binary algorithms were tested on 23 benchmark functions. The other four binary algorithms are BFMO, DBE, BPSO, and BGWO.

The experimental results are shown in Table 12. Among the 23 test functions, BPPE(U_1) obtains the optimal value in 19 benchmark functions. BPPE(U_1) failed to obtain the optimal value in F5, F8, F16, and F20. BGWO, DBE, and BFMO are optimized in individual reference functions. BFMO performs well on some specific benchmark functions, such as F16 and F20, but the number of functions to obtain the optimal value is small, which is suitable for solving fixed problems. In summary, the optimal time and function performance of BPPE(U_1) is the best. The PPE algorithm proposed in this paper has good performance on most functions and has good universality. It is suitable for solving fixed problems. In conclusion, BPPE(U_1) has the most optimal times and the best function performance. The BPPE algorithm proposed in this paper has good performance on most functions and has good universality.

Table 9. Statistical results of the transfer function in the original equation of the U-shaped function

	U_1		U_2		U_3		U_4	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F2	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F3	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F4	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F5	**2.3200**	7.8754	4.8140	10.8011	9.3380	13.5636	12.1800	14.3275
F6	**7.5000**	0.0000	**7.5000**	0.0000	**7.5000**	0.0000	**7.5000**	0.0000
F7	0.0016	0.0016	0.0015	0.0014	**0.0014**	0.0015	**0.0014**	0.0016
F8	**12544.2559**	0.0000	**12544.2559**	0.0000	**12544.2559**	0.0000	**12544.2559**	0.0000
F9	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F10	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F11	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F12	**1.6690**	0.0000	**1.6690**	0.0000	**1.6690**	0.0000	**1.6690**	0.0000
F13	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	0.0004	0.0063
F14	**11.6705**	0.0000	**11.6705**	0.0000	**11.6705**	0.0000	**11.6705**	0.0000
F15	**0.1481**	0.0000	**0.1481**	0.0000	**0.1481**	0.0000	**0.1481**	0.0000
F16	**1.0316**	0.0000	**1.0316**	0.0000	**1.0316**	0.0000	**1.0316**	0.0000
F17	**27.3049**	0.0000	**27.3049**	0.0000	**27.3049**	0.0000	**27.3049**	0.0000
F18	**597.0000**	0.0000	**597.0000**	0.0000	**597.0000**	0.0000	**597.0000**	0.0000
F19	**3.5252**	0.0000	**3.5252**	0.0000	**3.5252**	0.0000	**3.5252**	0.0000
F20	**3.1563**	0.0091	3.1576	0.0114	3.1577	0.0114	3.1592	0.0138
F21	**4.9448**	0.0000	**4.9448**	0.0000	**4.9448**	0.0000	**4.9448**	0.0000
F22	**4.9123**	0.0000	**4.9123**	0.0000	**4.9123**	0.0000	4.9207	0.1868
F23	**4.8715**	0.0000	**4.8715**	0.0000	**4.8715**	0.0000	**4.8715**	0.0000

Table 10. Statistical results of the transfer function in the original equation of the Z-function

	Z_2		Z_3		Z_3		Z_4	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	**0.0000**	0.0000	**0.0000**	0.0000	0.0020	0.0447	0.1240	0.5633
F2	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	0.0040	0.0632
F3	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F4	**0.0000**	0.0000	0.0700	0.2554	0.1740	0.3795	0.3320	0.4714
F5	**29.0000**	0.0000	63.2760	66.8132	109.2600	89.8967	171.2220	102.8733
F6	**7.5520**	0.3186	8.7320	1.1819	9.4240	1.5201	10.2640	1.7454
F7	**0.0054**	0.0055	0.0165	0.0166	0.0291	0.0297	0.0607	0.0612
F8	12550.4239	0.8428	12550.2387	0.8858	12550.1462	0.8688	**12550.0839**	0.8099
F9	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
F10	**0.0000**	0.0000	0.2080	0.3257	0.3836	0.3669	0.5848	0.3655
F11	**0.0000**	0.0000	0.0004	0.0066	0.0013	0.0119	0.0059	0.0260
F12	**1.6691**	0.0023	1.7353	0.0754	1.7766	0.0887	1.8502	0.1097
F13	0.8058	0.1172	0.7878	0.1048	0.7774	0.1066	**0.7642**	0.1006
F14	**11.6705**	0.0000	**11.6705**	0.0000	**11.6705**	0.0000	**11.6705**	0.0000
F15	**0.1481**	0.0000	**0.1481**	0.0000	**0.1481**	0.0000	**0.1481**	0.0000
F16	**1.0316**	0.0000	**1.0316**	0.0000	**1.0316**	0.0000	**1.0316**	0.0000
F17	**27.3049**	0.0000	**27.3049**	0.0000	**27.3049**	0.0000	**27.3049**	0.0000
F18	**597.0000**	0.0000	**597.0000**	0.0000	**597.0000**	0.0000	**597.0000**	0.0000
F19	**3.5252**	0.0000	**3.5252**	0.0000	**3.5252**	0.0000	**3.5252**	0.0000
F20	3.1612	0.0153	3.1596	0.0139	3.1600	0.0145	**3.1587**	0.0127
F21	4.9532	0.1866	4.9532	0.1866	**4.9448**	0.0000	**4.9448**	0.0000
F22	**4.9123**	0.0000	4.9290	0.2639	4.9207	0.1868	**4.9123**	0.0000
F23	**4.8715**	0.0000	**4.8715**	0.0000	**4.8715**	0.0000	**4.8715**	0.0000

Table 11. Comparison of statistical results of transfer functions in original equations of U, V, S, Z

	S_1	V_3	U_1	Z_1
F1	6.2800	**0.0000**	**0.0000**	**0.0000**
F2	6.1200	**0.0000**	**0.0000**	**0.0000**
F3	361.2500	**0.0000**	**0.0000**	**0.0000**
F4	1.0000	**0.0000**	**0.0000**	**0.0000**
F5	610.3440	17.1000	**2.3200**	29.0000
F6	19.8920	7.5160	**7.5000**	7.5520
F7	76.8208	0.0024	**0.0016**	0.0054
F8	12549.2900	**12544.2559**	**12544.2559**	12550.4239
F9	6.1020	**0.0000**	**0.0000**	**0.0000**
F10	1.7239	8.8818	**0.0000**	**0.0000**

F11	0.2072	0.0006	**0.0000**	**0.0000**
F12	2.6562	1.6731	**1.6690**	1.6691
F13	0.6142	0.0038	**0.0000**	0.8058
F14	**11.6705**	**11.6705**	**11.6705**	**11.6705**
F15	**0.1481**	**0.1481**	**0.1481**	**0.1481**
F16	**1.0316**	**1.0316**	**1.0316**	**1.0316**
F17	**27.3049**	**27.3049**	**27.3049**	**27.3049**
F18	**597.0000**	**597.0000**	**597.0000**	**597.0000**
F19	**3.5252**	**3.5252**	**3.5252**	**3.5252**
F20	**3.1543**	**3.1545**	**3.1563**	**3.1612**
F21	**4.9448**	**4.9448**	**4.9448**	**4.9532**
F22	**4.9123**	**4.9123**	**4.9123**	**4.9123**
F23	**4.8715**	**4.8715**	**4.8715**	**4.8715**

Table 12. Comparison of BPPE with other binary functions

	BPPE(U_1)	BFMO	DBE	PSO	BGWO
	AVG	AVG	AVG	AVG	AVG
F1	**0.0000**	0.5667	0.2353	8.2549	1.6471
F2	**0.0000**	0.5667	0.1765	7.9804	1.6471
F3	**0.0000**	5.5000	5.8824	649.6471	27.4706
F4	**0.0000**	0.9000	1.0000	1.0000	1.0000
F5	2.3200	142.9000	**0.0000**	**0.0000**	**0.0000**
F6	**7.5000**	8.5000	8.0882	23.8137	10.7549
F7	**0.0016**	7.1454	6.6079	114.9173	16.0392
F8	12544.2559	-18.4563	**-12.6751**	**-12.6751**	**-12.6751**
F9	**0.0000**	0.5667	0.2549	8.1176	1.3922
F10	**0.0000**	0.2726	0.1322	1.9695	0.7194
F11	**0.0000**	0.0148	0.0286	0.2908	0.0807
F12	**1.6690**	1.7661	1.7265	3.0965	1.9142
F13	**0.0000**	0.7933	**0.0000**	**0.0000**	**0.0000**
F14	**11.6705**	12.6705	**11.6705**	**11.6705**	**11.6705**
F15	**0.1481**	**0.1484**	**0.1481**	**0.1481**	**0.1481**
F16	1.0316	**0.0000**	1.0316	1.0316	1.0316
F17	**27.3049**	27.7029	**27.3049**	**27.3049**	**27.3049**
F18	**597.0000**	600.0000	**597.0000**	**597.0000**	**597.0000**
F19	**3.5252**	-0.3348	**3.5252**	**3.5252**	**3.5252**
F20	3.1563	**-0.1657**	3.1543	3.1597	3.1594
F21	**4.9448**	-5.0552	**4.9448**	**4.9448**	**4.9448**
F22	**4.9123**	-5.0877	**4.9123**	**4.9123**	**4.9123**
F23	**4.8715**	-5.1285	**4.8715**	**4.8715**	**4.8715**

Table 13. Comparison of TV-BPPE with PPE and other binary functions

	BPPE-U_1	TV-BPPE		BPPE-U_1	TV-BPPE
F1	**0.0000**	**0.0000**	F13	**0.0000**	**0.0000**
F2	**0.0000**	**0.0000**	F14	**11.6705**	**11.6705**
F3	**0.0000**	**0.0000**	F15	**0.1481**	**0.1481**
F4	**0.0000**	**0.0000**	F16	**1.0316**	**1.0316**
F5	2.3200	**0.0000**	F17	**27.3049**	**27.3049**
F6	**7.5000**	**7.5000**	F18	**597.0000**	**597.0000**
F7	0.0016	**0.0015**	F19	**3.5252**	**3.5252**
F8	**12544.2559**	**12544.2559**	F20	**3.1563**	**3.1570**
F9	**0.0000**	**0.0000**	F21	**4.9448**	**4.9448**
F10	**0.0000**	**0.0000**	F22	4.9123	4.9123
F11	**0.0000**	**0.0000**	F23	**4.8715**	**4.8715**
F12	**1.6690**	**1.6690**			

6 TV-BPPE and BPPE Experimental results and Analysis

The experimental results are shown in Table 13. Among the 23 test functions, BPPE(U_1) obtains the optimal value in 21 benchmark functions. BPPE(U_1) failed to obtain the optimal value in F5 and F8. TV-BPPE(U_1) obtains the optimal value in 23 benchmark functions. It is worth noting that the calculation result using the TV-BPPE(U_1) function in function F5 reaches the optimal value of the function. Adding a time function to the BPPE(U_1) optimizes the BPPE(U_1).

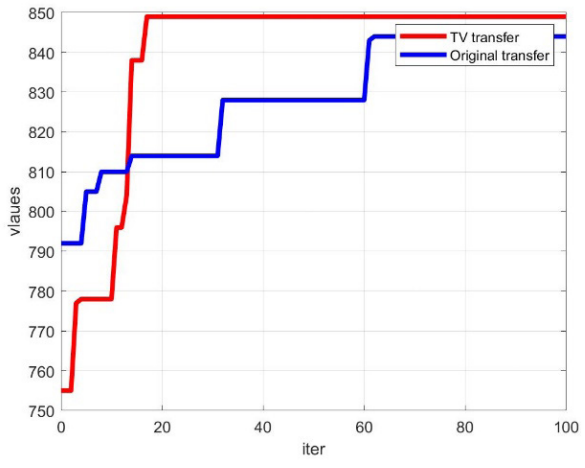


Figure 8. The convergence curves of average fitness functions for the datasets

In Figure 8, *iter* stands for the number of iterations, and *values* stand for fitness value, which in this paper is equal to the backpack value. These curves are drawn from the data set generated by 100 iterations of knapsack data in 30 dimensions. TV-BPPE provides a better solution. According to this figure, the time-varying transfer function tends to find the best accuracy with the minimum number of features faster than the original BPPE. TV-BPPE has a stronger exploration ability and faster convergence speed.

According to the experimental results and convergence curves, we can see that using time-varying indeed has the advantages of stronger convergence capability and better global exploration capability.

6.1 Application for the 0–1 Knapsack Problem

The 0-1 knapsack is a classic NP-hard problem. A mathematical model of the problem is that: Suppose we have n items, each item i has a weight w_i and a profit p_i, where $i = 1, 2, ..., n$. The goal is to select a subset of items such that the total profit is maximized such that the total weight of the selected items does not exceed the knapsack capacity C. Among them, w_i and p_i are randomly set test data, and c is the set fixed maximum capacity. To model this problem, x_i is introduced as a decision variable for each item. Use x_i as the solution set to store the knapsack combination that maximizes profit.

6.2 Datasets Description

The results of different binary swarm intelligence algorithms in 0-1 knapsack in multiple dimensions are tested experimentally.

The number of iterations is 500. *avg* is average, and *the median* is the median value in the results. *best* is the best result. *The winner* is the name of the best algorithm. The binary algorithm is used to obtain the optimal solution and the algorithm is used to calculate the maximum value of the backpack. In the multi-dimensional test, 20, 40, 50, and 100 dimensions were tested respectively. Comparison binary algorithms include BPPE, TV-BPPE, DBE, BPSO, and BFMO. Table 14 to Table 17 lists the experimental results.

Table 14. 20-dimensional comparison

	Best	Median	Avg
BPPE	1024	970	971.26
DBE	939	838	839.7
BPSO	959	845	851.42
BFMO	910	854	876.6
TV-BPPE	1024	964.5	968.66

Table 15. 40-dimensional comparison

	Best	Median	Avg
BPPE	1219	1125	1119.5
DBE	1152	1089	1091.14
BPSO	1157	1094.5	1098.24
BFMO	1150	1094.5	1100.53
TV-BPPE	1201	1133	1133.34

As shown in Table 14, in the 0-1 backpack experiment with dimension 20, the maximum backpack capacity is set to 878. Taking the average as the evaluation standard, the optimal volumetric value function for 0-1 knapsack in 20 dimensions is BPPE. TV-BPPE has the second-highest volume value after BPPE.

As shown in Table 15, the maximum backpack capacity is set to 1300. From the point of view of the average *avg*, TV-BPPE gains the best which a dimension is 40. From the point of view of optimum value *best*, BPPE gains the best. It is proved that the stability of TV-BPPE is better.

As shown in Table 16, the experiment dimension is set to 50, the maximum backpack capacity is set to 1600. The algorithm that performs best in 0-1 knapsack in 50 dimensions is TV-BPPE, followed by BPPE. The worst-performing algorithm was DBE. After the dimension of 0-1 knapsack is increased, the optimization function of the TV-BPPE algorithm is better played.

As shown in Table 17, the dimension is set to 100, the maximum backpack capacity is set to 3500. Whether it's the optimal value or the average value, the optimal volumetric value function for 0-1 knapsack in 100 dimensions is TV-BPPE. The second-best algorithm is BPPE. The remaining three algorithms have similar effects. The performance of TV-BPPE begins to exceed that of BPPE when the dimension is increased.

Table 18 compares the effects of four algorithms under 20 dimensions, 40 dimensions, 50 dimensions, and 100 dimensions. According to the results of the comparison and data analysis with other binary functions, BPPE and TV-BPPE are good solutions to binary problems in multiple dimensions.

Table 16. 50-dimensional comparison

	Best	Median	Avg
BPPE	1536	1493.5	1495.58
DBE	1437	1375	1375.6
BPSO	1545	1403.5	1406.6
BFMO	1459	1094.5	1409.8
TV-BPPE	1551	1516	1517.6

Table 17. 100-dimensional comparison

	Best	Median	Avg
BPPE	3235	3094	3093.26
DBE	2937	2712.5	2715.14
BPSO	3112	2717.5	2731.26
BFMO	3034	2654.83	2698.3
TV-BPPE	3291	3149	3152.46

Table 18. 20, 40, 50, 100-dimensional comparison

	BPPE		DBE		BPSO		BFMO		TV-BPPE		winner	
Dim	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best
20	**971.26**	**1024**	893,7	939	851.42	959	876.6	910	968.66	**1024**	BPPE	TV-BPPE BPPE
40	1119.5	**1219**	1091.14	1152	1098.24	1157	1100.53	1150	**1133.34**	1201	TV-BPPE	BPPE
50	1495.58	1536	1375.6	1437	1406.6	1545	1409.8	1459	**1517.6**	1551	BPPE	TV-BPPE
100	3093.26	3235	2715.14	2937	2731.26	3112	2698.3	3034	**3152.46**	3291	BPPE	TV-BPPE

6.3 Experimental Results and Analysis

As can be seen from Table 14 to Table 17, both BPPE and TV-BPPE get the best results in the case of 20 dimensions. With the increase of dimension, the difficulty of finding the optimal solution increases. In terms of the optimal value, the TV-BPPE is worse than BPPE in the 40-dimensional case. In terms of average value TV-BPPE get the best. But, TV-BPPE exceeds that of BPPE in 50 dimensions in the double index of mean and optimal value. The result on 100-dimensional TV-BPPE is quite better than other functions.

At low dimensions, BPPE is slightly better than TV-BPPE. But TV-BPPE is more stable. As the dimension increases, the performance of TV-BPPE gradually surpasses that of BPPE. Several experiments have shown that TV-BPPE performs better at high latitudes. However, It can be seen from the median value that TV-BPPE performs better in the later stage of development.

Therefore, it can be predicted that the superiority of TV-BPPE will be more prominent after the dimension continues to rise. It can be seen that the time-varying binary PPE algorithm can achieve better results in high-latitude knapsack problems.

7 Conclusion

The binary Phasmatodea Population Evolution Algorithm solves the discretization problems of 0-1 knapsack. The transfer functions with the focus of converting the standard PPE into a binary PPE and a time-varying binary PPE. This paper analyzes the problem-solving effect of PPE in binary conditions. Then, based on improving the ability of exploration and development, the transfer function is optimized, and a transfer function updating equation with a time variable is proposed. These methods show remarkable performance in the benchmark function and have fast convergence.

Finally, a 0-1 knapsack is used for multi-dimensional testing. Simulation results show that the improved method (TV-BPPE) has better late development ability and stability than BPPE. It is more advantageous to solve high latitude problems. In this paper, 0-1 knapsack is just used to verify the ability of this algorithm to solve discrete problems. In the future, it can be combined with neural networks to solve more discrete problems.

References

[1] P. M. Vasant, *Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, and Finance*, IGI Global, 2013.

[2] A. P. Piotrowski, J. J. Napiorkowski, A. E. Piotrowska, Particle Swarm Optimization or Differential Evolution—A comparison, *Engineering Applications of Artificial Intelligence*, Vol. 121, Article No. 106008, May, 2023.

[3] N. Singh, R. Arya, R. K. Agrawal, A novel approach to combine features for salient object detection using constrained particle swarm optimization, *Pattern Recognition*, Vol. 47, No. 4, pp. 1731 -1739, April, 2014.

[4] J. F. Chang, S. C. Chu, J. F. Roddick, J. S. Pan, A Parallel Particle Swarm Optimization Algorithm with Communication Strategies, *Journal of Information Science and Engineering*, Vol. 21, No. 4, pp. 809-818, July, 2005.

[5] X. F. Song, Y. Zhang, D. W. Gong, X. Y. Sun, Feature selection using bare-bones particle swarm optimization with mutual information, *Pattern Recognition*, Vol. 112, Article No. 107804, April, 2021.

[6] L. Wang, J. F. Cao, C. Z. Han, Multidimensional particle swarm optimization-based unsupervised planar segmentation algorithm of unorganized point clouds, *Pattern Recognition*, Vol. 45, No. 11, pp. 4034-4043, April, 2012.

[7] X. D. Wu, Y. H. Yang, Y. Q. Sun, Y. Xie, X. S. Song, B. Huang, Dynamic regional splitting planning of remote sensing satellite swarm using parallel genetic PSO algorithm, *Acta Astronautica*, Vol. 204, pp. 531-551, November, 2022.

[8] R. Ranjan, J. K. Chhabra, Automatic clustering and feature selection using multi-objective crow search algorithm, *Applied Soft Computing*, Vol. 142, Article No. 110305, July, 2023.

[9] A. Chaudhuri, T. P. Sahu, Feature selection using Binary Crow Search Algorithm with time varying flight length, *Expert Systems with Applications*, Vol. 168, Article No. 114288, April, 2021.

[10] Y. K. Ke, J. Xie, S. Pouramini, Utilization of an improved crow search algorithm to solve building energy optimization problems: Cases of Australia, *Journal of Building Engineering*, Vol. 38, Article No. 102142, June, 2021.

[11] J. Wang, D. Lin, Y. Z. Zhang, S. G. Huang, An adaptively balanced grey wolf optimization algorithm for feature selection on high-dimensional classification, *Engineering Applications of Artificial Intelligence*, Vol. 114 , Article No. 105088, September, 2022.

[12] H. J. Liu, Research on cloud computing adaptive task scheduling based on ant colony algorithm, *Optik*, Vol. 258, Article No. 168677, May, 2022.

[13] Q. Y. Wang, Y. K. Sun, Q. C. Tang, L. Ma, D. Zhao, J. P. Yang, J. Xu, A Dual-Robot Cooperative Welding Path Planning Algorithm Based on Improved Ant Colony Optimization, *IFAC-PapersOnLine*, Vol. 55, No. 8, pp. 7-12, 2022.

[14] L. Xu, K. Huang, J. P. Liu, D. S. Li, Y. F. Chen, Intelligent planning of fire evacuation routes using an improved ant colony optimization algorithm, *Journal of Building Engineering,* Vol. 61, Article No. 105208, December, 2022.

[15] H. T. Zhao, C. S. Zhang, An ant colony optimization algorithm with evolutionary experience-guided pheromone updating strategies for multi-objective optimization, *Expert Systems with Applications*, Vol. 201, Article No. 117151, September, 2022.

[16] M. Dorigo, T. Stützle, *Ant Colony Optimization*, MIT Press, 2004.

[17] Y. F. Zhang, J. P. Zhao, L. M. Wang, H. G. Wu, R. H. Zhou, J. L. Yu, An improved OIF Elman neural network based on CSO algorithm and its applications, *Computer Communications*, Vol. 171, pp. 148-156, April, 2021.

[18] M. F. Sohail, C. Y. Leow, S. H. Won, A Cat Swarm Optimization based transmission power minimization for an aerial NOMA communication system, *Vehicular Communications*, Vol. 33, Article No. 100426, January, 2022.

[19] A. P. Karpenko, I. A. Leshchev, Advanced Cat Swarm Optimization Algorithm in Group Robotics Problem, *Procedia Computer Science*, Vol. 150, pp. 95-101, 2019.

[20] V. I. Skoullis, I. X. Tassopoulos, G. N. Beligiannis, Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm, *Applied Soft Computing*, Vol. 52, pp. 277-289, March, 2017.

[21] M. A. K. Azad, A. M. A. C. Rocha, E. M. G. P. Fernandes, Improved binary artificial fish swarm algorithm for the 0–1 multidimensional knapsack problems, *Swarm and Evolutionary Computation*, Vol. 14, pp. 66-75, February, 2014.

[22] D.Y. Zhuang, K. Ma, C. A. Tang, Z. Z. Liang, K. K. Wang, Z. W. Wang, Mechanical parameter inversion in tunnel engineering using support vector regression optimized by multi-strategy artificial fish swarm algorithm, *Tunnelling and Underground Space Technology*, Vol. 83, pp. 425-436, January, 2019.

[23] J. W. Huang, J. Zeng, Y. F. Bai, Z. Cheng, Z. H. Feng, L. Qi, D. K. Liang, Layout optimization of fiber Bragg grating strain sensor network based on modified artificial fish swarm algorithm, *Optical Fiber Technology*, Vol. 65, Article No. 102583, September, 2021.

[24] Y. Liu, X. S. Feng, Y. Yang, Z. J. Ruan, L. K. Zhang, K. M. Li, Solving urban electric transit network problem by integrating Pareto artificial fish swarm algorithm and genetic algorithm, *Journal of Intelligent Transportation Systems*, Vol. 26, No. 3, pp. 253-268, 2021.

[25] P. C. Song, S. C. Chu, J. S. Pan, H. M. Yang, Simplified Phasmatodea population evolution algorithm for optimization, *Complex & Intelligent Systems*, Vol. 8, No. 4, pp. 2749-2767, August, 2022.

[26] J. S. Pan, P. C. Song, C. A. Pan, A. Abraham, The phasmatodea population evolution algorithm and its application in 5G heterogeneous network downlink power allocation problem, *Journal of Internet Technology*, Vol. 22. No. 6, pp. 1199-1213, November, 2021.

[27] P. C. Song, S. C. Chu, J. S. Pan, H. M. Yang, Phasmatodea population evolution algorithm and its application in length-changeable incremental extreme learning machine, *International Conference on Industrial Artificial Intelligence*, Shenyang, China, 2020, pp. 1-5.

[28] V. Cacchiani, M. Iori, A. Locatelli, S. Martello, Knapsack problems — An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems, *Computers & Operations Research*, Vol. 143, Article No. 105693, July, 2022.

[29] V. Roostapour, A. Neumann, F. Neumann, Single- and multi-objective evolutionary algorithms for the knapsack problem with dynamically changing constraints, *Theoretical Computer Science*, Vol. 924, pp. 129-147, July, 2022.

[30] C. Wilbaut, R. Todosijevic, S. Hanafi, A. Fréville, Heuristic and exact reduction procedures to solve the discounted 0–1 knapsack problem, *European Journal of Operational Research*, Vol. 304, No. 3, pp. 901-911, February, 2023.

[31] M. A. Basset, R. Mohamed, O. M. ELkomy, Knapsack Cipher-based metaheuristic optimization algorithms for cryptanalysis in blockchain-enabled internet of things systems, *Ad Hoc Networks*, Vol. 128, Article No. 102798, April, 2022.

[32] Y. Kim, Y. E. Kim, H. Kim, A Comparison Experiment of Binary Classification for Detecting the GTP Encapsulated IoT DDoS Traffics in 5G Network, *Journal of Internet Technology,* Vol. 23, No. 5, pp. 1049-1060, September, 2022.

[33] J. L. Zhou, S. C. Chu, A. Q. Tian, Y. J. Peng, J. S. Pan, Intelligent Neural Network with Parallel Salp Swarm Algorithm for Power Load Prediction, *Journal of Internet Technology,* Vol. 23, No. 4, pp. 643-657, July, 2022.

[34] P. Hu, J. S. Pan, S. C. Chu, Improved Binary Grey Wolf Optimizer and Its Application for Feature Selection, *Knowledge-Based Systems*, Vol. 195, No. 11, Article No. 105746, May, 2020.

[35] Z. C. Dou, Z. Zhuang, L. P. Kong, J. S. Pan, S. C. Chu, Binary Fish Migration Optimization for Feature Selection Using Time-Varying Transfer Function, *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, Kaohsiung, Taiwan, 2022, pp. 311-321.

[36] 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.

[37] H. Yin, J. Qiao, P. Fu, X. Xia, Face feature selection with binary particle swarm optimization and support vector machine, *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 5, No. 4, pp. 731-739, October, 2014.

[38] B. P. Abbott, R. Abbott, T. D. Abbott, F. Acernese, Observation of Gravitational Waves from a Binary Neutron Star Inspiral, *Physical review letters*, Vol. 119, Article No. 161101, October, 2017.

[39] W. C. Yeh, C. M. Du, S. Y. Tan, M. Forghani-elahabad, Application of LSTM Based on the BAT-MCS for Binary-State Network Approximated Time-Dependent Reliability Problems, *Reliability Engineering & System Safety*, Vol. 235, Article No. 108954, July, 2023.

[40] M. Awadallah, M. A. A. Betar, M. S. Braik, A. I. Hammouri, I. A. Doush, R. A. Zitar, An enhanced binary Rat Swarm Optimizer based on local-best concepts of PSO and collaborative crossover operators for feature selection, *Computers in Biology and Medicine*, Vol. 147, Article No. 105675, August, 2022.

[41] D. Q. Zhao, C. C. Cai, L. Y. Li, A binary discrete particle swarm optimization satellite selection algorithm with a queen informant for Multi-GNSS continuous positioning, *Advances in Space Research*, Vol. 68, No. 9, pp. 3521-3530, November, 2021.

[42] G. L. Wang, S. C. Chu, A. Q. Tian, T. Liu, J. S. Pan, Improved Binary Grasshopper Optimization Algorithm for Feature Selection Problem, *Entropy*, Vol. 24, No. 6, Article No. 777, June, 2022.

[43] S. C. Chu, Z. Zhuang, C. C. Hu, J. S. Pan, Binary QUATRE Using Time-varying Transfer Functions, *Journal of Internet Technology*, Vol. 23, No. 3, pp. 425-435, May, 2022.

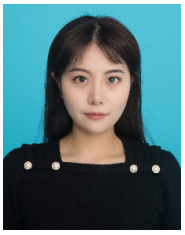
[44] Z. Beheshti, BMPA-TVSinV: A Binary Marine Predators Algorithm using time-varying sine and V-shaped transfer functions for wrapper-based feature selection, *Knowledge-Based Systems*, Vol. 252, Article No. 109446, September, 2022.

[45] H. Hizarci, O. Demirel, B. E. Turkay, Distribution network reconfiguration using time-varying acceleration coefficient assisted binary particle swarm optimization, *Engineering Science and Technology, an International Journal*, Vol. 35, Article No. 101230, November, 2022.

[46] N. M. Yusof, A. K. Muda, S. F. Pratama, R. Carbo-Dorca, A. Abraham, Improving Amphetamine-type Stimulants drug classification using chaotic-based time-varying binary whale optimization algorithm, *Chemometrics and Intelligent Laboratory Systems*, Vol. 229, Article No. 104635, October, 2022.

[47] C. M. Chen, S. Lv, J. Ning, J. M. Wu, A Genetic Algorithm for the Waitable Time-Varying Multi-Depot Green Vehicle Routing Problem, *Symmetry*, Vol. 15, No. 1, Article No. 124, January, 2023.

Biographies



Jiayin Lou received the B.S. degree from Harbin University of Science and Technology, China, in 2020. She is currently pursuing the master degree with the Northeast Electric Power University of Science and Technology, Jilin, China. Her research interests are swarm intelligence .

Shu-Chuan Chu received the Ph.D. degree in 2004 from the School of Computer Science, Engineering and Mathematics, Flinders University of South Australia. She joined Flinders University in December 2009 after 9 years at the Cheng Shiu University, Taiwan. She is the Research Fellow in the College of Science and Engineering of Flinders University, Australia from December 2009. Currently, She is the Research Fellow with PhD advisor in the College of Computer Science and Engineering of Shandong University of Science and Technology from September 2019. Her research interests are mainly in Swarm Intelligence, Intelligent Computing and Data Mining.

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 in Shandong University of Science and Technology. He is the IET Fellow, U.K., and has been the Vice Chair of the IEEE Tainan Section.

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.