Swarm and Evolutionary Computation 9 (2013) 1–14

Contents lists available at SciVerse ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Regular paper

S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization

Seyedali Mirjalili *, Andrew Lewis

School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD 4111, Australia

ARTICLE INFO

Article history:
Received 11 September 2011
Received in revised form
14 September 2012
Accepted 14 September 2012
Available online 2 October 2012

Keywords:
PSO
Particle swarm
Transfer function
Evolutionary algorithm
Heuristic algorithm
BPSO

ABSTRACT

Particle Swarm Optimization (PSO) is one of the most widely used heuristic algorithms. The simplicity and inexpensive computational cost makes this algorithm very popular and powerful in solving a wide range of problems. The binary version of this algorithm has been introduced for solving binary problems. The main part of the binary version is a transfer function which is responsible to map a continuous search space to a discrete search space. Currently there appears to be insufficient focus on the transfer function in the literature despite its apparent importance. In this study six new transfer functions divided into two families, s-shaped and v-shaped, are introduced and evaluated. Twenty-five benchmark optimization functions provided by CEC 2005 special session are employed to evaluate these transfer functions and select the best one in terms of avoiding local minima and convergence speed. In order to validate the performance of the best transfer function, a comparative study with six recent modifications of BPSO is provided as well. The results prove that the new introduced v-shaped family of transfer functions significantly improves the performance of the original binary PSO.

1. Introduction

Particle Swarm Optimization (PSO) is one of the most widely used evolutionary algorithms inspired from social behavior of animals [1,2]. The simplicity and inexpensive computational cost makes this algorithm very popular. Due to these advantages, the PSO algorithm has been applied to many domains such as medical diagnosis [3], grid scheduling [4], robot path planning [5], wavelength detection [6], and video abstraction [7]. This algorithm is capable of solving problems with continuous search spaces, while some problems have discrete search spaces.

There are many optimization problems with discrete binary search spaces. They need binary algorithms for their solution. Most of the well-known evolutionary algorithms such as Harmony Search (HS) [8], Differential Evolution Algorithm (DE) [9], Magnetic Optimization Algorithm (MOA) [10], and Gravitational Search Algorithm (GSA) [11] have binary versions which make them capable of performing in binary spaces. There are different methods to develop the binary version of a continuous heuristic algorithm while preserving the concepts of the search process. For instance, a set of harmony search considerations and pitch adjustment rules have been employed with the purpose of converting HS to Binary HS (BHS) [12], while Wang et al. [13] proposed a probability estimation operator in order to solve discrete problems by DE. Furthermore, the Binary MOA [14] and Binary GSA [15] algorithms have been introduced utilizing transfer functions and position updating rules. Whereas BHS and BDE are different from their originating algorithms, the MOA and GSA algorithms retain the same concepts of position and velocity updating procedures. Consequently, the concepts used in converting MOA and GSA to binary algorithms are readily applicable to PSO.

The binary version of PSO (BPSO) was proposed by Kennedy and Eberhart in 1997 [16]. The continuous and binary versions of PSO are distinguished by two different components: a new transfer function and a different position updating procedure. The transfer function is used to map a continuous search space to a binary one, and the updating process is designed to switch positions of particles between 0 and 1 in binary search spaces. According to Luh and Lin [17], the original version of BPSO suffers trapping in local minima, so some modifications of BPSO have been introduced in order to overcome this problem.

In 2008 Yin proposed a new, modified BPSO using a new updating position formula for optimal polygonal approximation of digital curves [18]. In the same year a modified BPSO for variable selection in MLR and PLS modeling was proposed by Shen and Jiang [19]. In 2008, Wang et al. introduced a novel probability BPSO algorithm and investigated its application in solving the knapsack problem [20]. Moreover, an analogous

* Corresponding author.
E-mail addresses: seyedali.mirjalili@griffithuni.edu.au (S. Mirjalili), a.lewis@griffith.edu.au (A. Lewis).

2210-6502/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.swevo.2012.09.002

modified BPSO algorithm adopting the genotype–phenotype representation and mutation operator was introduced in this year by Lee et al. [21]. Later, Chuand et al. proposed an improved BPSO using the "catfish effect" and employed it in a feature selection method [22]. Some of these researchers manipulated the interaction topology of BPSO, such as Catfish BPSO, whereas others tried to use new updating procedures for position and velocity vectors in order to improve the performance of conventional BPSO.

Considering the above-mentioned works, it could be stated that the transfer function is the most important part of the BPSO algorithm [23]. Currently there appears to be insufficient focus on the transfer function in the literature despite its apparent importance. In this study, two different families of transfer functions, v-shaped and s-shaped, consisting of six new transfer functions are introduced and evaluated. The effectiveness of employing these new transfer functions is investigated in terms of avoiding local minima and convergence speed. Additionally, a comparative study is provided to compare the proposed approach with some recent modifications of BPSO in order to validate its performance.

The rest of the paper is organized as follows: Section 2 presents a brief introduction to PSO. Section 3 discusses the basic principles of a binary version of PSO. The introduced families of transfer functions are described in Section 4. The experimental results are demonstrated in Section 5. Finally, Section 6 concludes the work and suggests some directions for future research.

2. The Particle Swarm Optimization

PSO is an evolutionary computation technique that was proposed by Kennedy and Eberhart [1,2]. It was inspired from the social behavior of bird flocking. It uses a number of particles (candidate solutions) which fly around in the search space to find the best solution. Meanwhile, they all trace the best location (best solution) in their paths. In other words, particles consider their own best solutions as well as the best solution the swarm has obtained so far.

Each particle in PSO should consider the current position, the current velocity, the distance to their personal best solution, *pbest*, and the distance to the global best solution, *gbest*, to modify its position. PSO was mathematically modeled as follow:

$$v_i^{t+1} = wv_i^t + c_1 \times rand \times (pbest_i - x_i^t) + c_2 \times rand \times (gbest - x_i^t) \quad (1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2)$$

where v_i^t is the velocity of particle i at iteration t, w is a weighting function, c_j is an acceleration coefficient, *rand* is a random number between 0 and 1, x_i^t is the current position of particle i at iteration t, $pbest_i$ is the best solution that the i-th particle has obtained so far, and *gbest* indicates the best solution the swarm has obtained so far.

The first part of Eq. (1), wv_i^t, provides exploration ability for PSO. The second and third parts, $c_1 \times rand \times (pbest_i - x_i^t)$ and $c_2 \times rand \times (gbest - x_i^t)$, represent private thinking and collaboration of particles, respectively. The PSO starts with randomly placing the particles in a problem space. At each iteration, the velocities of particles are calculated using Eq. (1). After defining the velocities, the position of particles can be calculated using Eq. (2). The process of changing particles' positions will continue until satisfying an end criterion.

3. Binary version of Particle Swarm Optimization

Generally, there are many problems that have intrinsic discrete binary search spaces, like feature selection and dimensionality reduction [24,25]. In addition, problems with continuous real search space can be converted into binary problems by converting their variables to binary variables. Regardless of binary problems' types, a binary search space has its own structure with some limitations.

A binary search space can be considered as a hypercube. The agents (particles) of a binary optimization algorithm can only shift to nearer and farther corners of the hypercube by flipping various numbers of bits [16]. Hence, in designing the binary version of PSO, some basic concepts of the velocity and position updating process must be modified.

In the continuous version of PSO, particles can move around the search space utilizing position vectors within the continuous real domain. Consequently, the concept of position updating can be easily implemented for particles by adding velocities to positions using Eq. (2). However, the meaning of position updating is different in a discrete binary space [26]. In binary space, due to dealing with only two numbers ("0" and "1"), the position updating process cannot be performed using Eq. (2). Therefore, a way should be found to use velocities for changing agents' positions from "0" to "1" or vice versa. In other words, a link has to be devised between velocity and position as well as revising the position updating Eq. (2).

Basically, in discrete binary spaces, the position updating means switching between "0" and "1" values. This switching should be done based on the velocities of agents. The question here is how the concept of velocity in a real space should be employed in order to update the positions in a binary space. According to [14–16], the idea is to change the position of an agent with the probability of its velocity. In order to do this, a transfer function is necessary to map velocity values to probability values for updating the positions.

The original BPSO was proposed by Kennedy and Eberhart [16] to allow PSO to operate in binary problem spaces. In this version, particles could only fly in a binary search space by taking values of 0 or 1 for their position vectors. The roles of velocities are to present the probability of a bit taking the value 0 or 1. A sigmoid function (transfer function part) as in Eq. (3) was employed to transform all real values of velocities to probability values in the interval [0,1].

$$T\left(v_i^k(t)\right) = \frac{1}{1+e^{-v_i^k(t)}} \quad (3)$$

where $v_i^k(t)$ indicates the velocity of particle i at iteration t in k-*th* dimension.

After converting velocities to probability values, position vectors could be updated with the probability of their velocities as follow:

$$x_i^k(t+1) = \begin{cases} 0 & \text{If } rand < T(v_i^k(t+1)) \\ 1 & \text{If } rand \geq T(v_i^k(t+1)) \end{cases} \quad (4)$$

The general steps of the BPSO algorithm are as follows.

a) All particles are initialized with random values.
b) Repeat steps c–e until meeting the end condition.
c) For all particles, velocities are defined using Eq. (1).
d) Calculate probabilities for changing the elements of position vectors using Eq. (3).
e) Update the elements of position vectors with the rules in Eq. (4).

As mentioned above, the original version of BPSO suffers trapping in local minima, so some modifications of the BPSO algorithm have been introduced in order to overcome this problem as follows:

Shen et al, [19] proposed a modification of BPSO for variable selection in MLR and PLS modeling. In this modification, the authors tried to use some rules to search binary spaces using PSO as in (5).

$$x_i^k(t+1)=\begin{cases} x_i^k(t) & \text{if } (0<v_i\le a)\\ pBest_i & \text{if } (0<v_i\le 0.5(1+a))\\ gBest & \text{if } (0.5(1+a)<v_i\le 1)\end{cases} \quad (5)$$

where v_i is a random number in the interval of [0,1] for all particles, and a starts from 0.5 and decreases to 0.33 through iterations.

This algorithm still suffered the problem of trapping in local minima, so the authors allowed 10% of particles to randomly move around search spaces without following the social and individual behaviors.

In 2008, a novel BPSO was proposed by Wang et al. [20] that utilized a novel updating strategy as follows:

$$L\left(x_i^k(t)\right)=\frac{\left(x_i^k(t)-R_{min}\right)}{\left(R_{max}-R_{min}\right)} \quad (6)$$

$$x_i^k(t+1)=\begin{cases} 1 & \text{if } rand()\le L\left(x_i^k(t)\right)\\ 0 & \text{if } rand()>L\left(x_i^k(t)\right)\end{cases} \quad (7)$$

where $L(x)$ is a linear function with output values in the interval of (0,1), $rand()$ is a random number in [0,1], and $[R_{max}, R_{min}]$ is a predefined range for gaining the probability value with $L(x)$ function.

By introducing this method, it was claimed that the computational complexity of BPSO could be reduced, while the optimization ability could be improved. However, this modification was not able to solve the problem of trapping in local minima for a wide range of problems.

In the same year, Lee et al. [21] introduced another modification of BPSO whereby they allow continuous PSO update both velocities and positions. After that they used position values instead of velocity values to calculate the probability of changing elements of particles' position vectors. In other words, they have two procedures for updating positions. The first one is the same as Eq. (2), and the second one is given as (8):

$$x_i^k(t+1)=\begin{cases} 0 & \text{If } rand<S(x_i^k(t+1))\\ 1 & \text{If } rand\ge S(x_i^k(t+1))\end{cases} \quad (8)$$

Similarly, they used Eq. (9) to calculate the probability values.

$$S\left(x_i^k(t)\right)=\frac{1}{1+e^{-x_i^k(t)}} \quad (9)$$

where $x_i^k(t)$ is the position of particle i at iteration t in k-th dimension.

It should be noted that this formula uses position values $(x_i^k(t))$ rather than velocity values $(v_i^k(t))$ to calculate the probability values.

This algorithm also faced the problem of trapping in local minima so that it was equipped with a mutation operator to deal with this problem.

All the above-mentioned algorithms could not resolve the problem of trapping in local minima completely. In addition, they have fluctuating convergence speeds in dealing with diverse optimization problems.

In the following section the efficiency of the transfer function in improving these two problems is explored by introducing a new family of transfer functions.

4. Proposed families of transfer functions

A transfer function defines the probability of changing a position vector's elements from 0 to 1 and vice versa. Transfer functions force particles to move in a binary space. According to Rashedi et al. [15], some concepts should be taken into account for selecting a transfer function in order to map velocity values to probability values as follows:

- The range of a transfer function should be bounded in the interval [0,1], as they represent the probability that a particle should change its position.
- A transfer function should provide a high probability of changing the position for a large absolute value of the velocity. Particles having large absolute values for their velocities are probably far from the best solution, so they should switch their positions in the next iteration.
- A transfer function should also present a small probability of changing the position for a small absolute value of the velocity.

Table 1
S-shaped and v-shaped families of transfer functions.

S-shaped family		V-shaped family	
Name	Transfer function	Name	Transfer function
S1	$T(x)=\frac{1}{1+e^{-2x}}$	V1	$T(x)=\left\lvert \mathrm{erf}\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert=\left\lvert \frac{\sqrt{2}}{\pi}\int_0^{(\sqrt{\pi}/2)x} e^{-t^2}dt\right\rvert$
S2 [8]	$T(x)=\frac{1}{1+e^{-x}}$	V2 [13]	$T(x)=\lvert\tanh(x)\rvert$
S3	$T(x)=\frac{1}{1+e^{(-x/2)}}$	V3	$T(x)=\lvert(x)/\sqrt{1+x^2}\rvert$
S4	$T(x)=\frac{1}{1+e^{(-x/3)}}$	V4	$T(x)=\lvert\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\rvert$

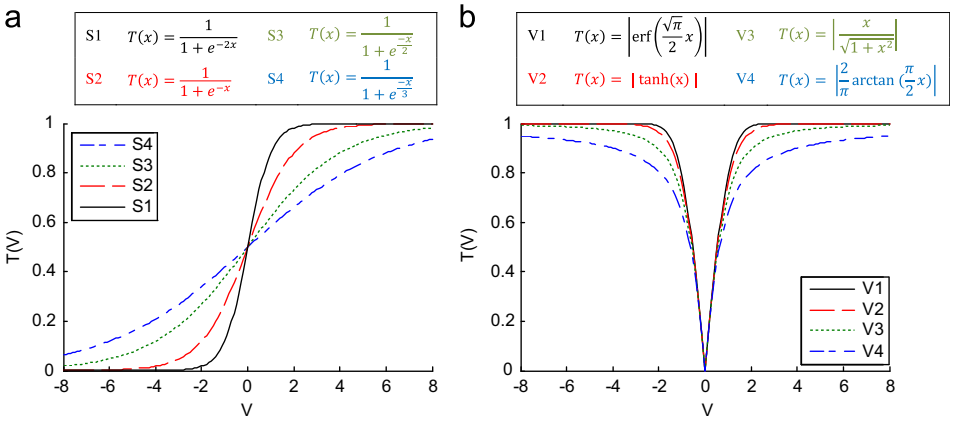

Fig. 1. (a) s-shaped and (b) v-shaped family of transfer functions.

- The return value of a transfer function should increase as the velocity rises. Particles that are moving away from the best solution should have a higher probability of changing their position vectors in order to return to their previous positions.
- The return value of a transfer function should decrease as the velocity reduces.

These concepts guarantee that a transfer function is able to map the process of search in a continuous search space to a binary search space while preserving similar concepts of the search (*Gbest* and *Pbest* in PSO) for a particular evolutionary algorithm.

The transfer function that was used for the conventional BPSO in [16] was presented as Eq. (3). This function is also shown in Fig. 1(a), named S2. As shown in this figure and Table 1, three new transfer functions manipulating the coefficient of x have been introduced. It may be observed that S1 sharply increases and reaches its saturation as the velocity

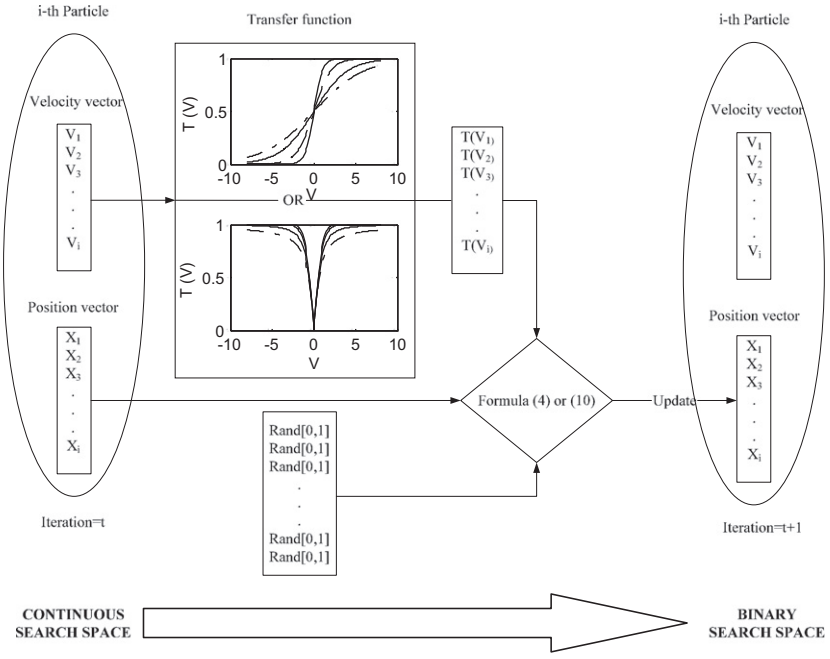

Fig. 2. Process of mapping a continuous search space to a discrete search space.

Table 2
Benchmark functions.

Benchmark functions	*Range*	*Dim*	f_{min}
Unimodal functions			
F_1: Shifted sphere function	[−100,100]	5	−450
F_2: Shifted schwefel's	[−100,100]	5	−450
F_3: Shifted rotated high conditioned elliptic function	[−100,100]	5	−450
F_4: Shifted schwefel's with noise in fitness	[−100,100]	5	−450
F_5: Schwefel's with global optimum on bounds	[−100,100]	5	−310
Multimodal functions			
F_6: Shifted rosenbrock's function	[−100,100]	5	390
F_7: Shifted rotated griewank's function without bounds	[0,600]	5	−180
F_8: Shifted rotated ackley's function with global optimum on bounds	[−32,32]	5	−140
F_9: Shifted rastrigin's function	[−5,5]	5	−330
F_{10}: Shifted rotated rastrigin's function	[−5,5]	5	−330
F_{11}: Shifted rotated weierstrass function	[−0.5,0.5]	5	90
F_{12}: Schwefel's	[−100,100]	5	−460
F_{13}: Expanded extended griewank's plus rosenbrock's function (F_8F_2)	[−3,1]	5	−130
F_{14}: Shifted rotated expanded scaffer's F_6	[−100,100]	5	−300
Composite functions			
F_{15}: Hybrid composition function	[−5,5]	10	120
F_{16}: Rotated hybrid composition function	[−5,5]	10	120
F_{17}: Rotated hybrid composition function with noise in fitness	[−5,5]	10	120
F_{18}: Rotated hybrid composition function	[−5,5]	10	10
F_{19}: Rotated hybrid composition function with a narrow basin for the global optimum	[−5,5]	10	10
F_{20}: Rotated hybrid composition function with the global optimum on the bounds	[−5,5]	10	10
F_{21}: Rotated hybrid composition function	[−5,5]	10	360
F_{22}: Rotated hybrid composition function with high condition number matrix	[−5,5]	10	360
F_{23}: Non-continuous rotated hybrid composition function	[−5,5]	10	360
F_{24}: Rotated hybrid composition function	[−5,5]	10	260
F_{25}: Rotated hybrid composition function without bounds	[−2,5]	10	260

value rises much higher than S2, while the saturations of S3 and S4 begin later than S2. It should be noted that the probability of changing the values of position vectors rises as the slope of these transfer functions increases, so S1 returns the highest probability among them for the same value of velocity, whereas S4 provides the lowest value. These transfer functions have been chosen with different slopes and saturations compared to S2 in order to investigate the efficiency of these characteristics in improving the performance of BPSO. Due to the shapes of these curves, they are named s-shaped transfer functions. Consequently, their group is named the ''s-shaped'' family of transfer functions as well. As could be inferred from Eq. (4), the position updating rules of the s-shaped family force particles to take values of 0 or 1.

On the other hand, another type of transfer functions could be used which have different position updating rules. One of these transfer functions by Rashedi et al. [15] is presented in Fig. 1(b) and Table 1as V2. Since these functions are quite different from the s-shaped family, they need new position updating rules. For these types of transfer functions formula (10) should be employed to update position vectors based on velocities.

$$x_i^k(t+1)=\begin{cases}(x_i^k(t))^{-1} & \text{If } rand < T(v_i^k(t+1))\\ x_i^k(t) & rand \geq T(v_i^k(t+1))\end{cases} \tag{10}$$

where $x_i^k(t)$ and $v_i^k(t)$ indicates the position and velocity of particle i at iteration t in k-th dimension, and $(x_i^k(t))^{-1}$ is the complement of $x_i^k(t)$.

The advantage of this method is that these transfer functions do not force particles to take values of 0 or 1. In other words, they encourage particles to stay in their current positions when their velocity values are low or switch to their complements when the velocity values are high. These are named v-shaped transfer

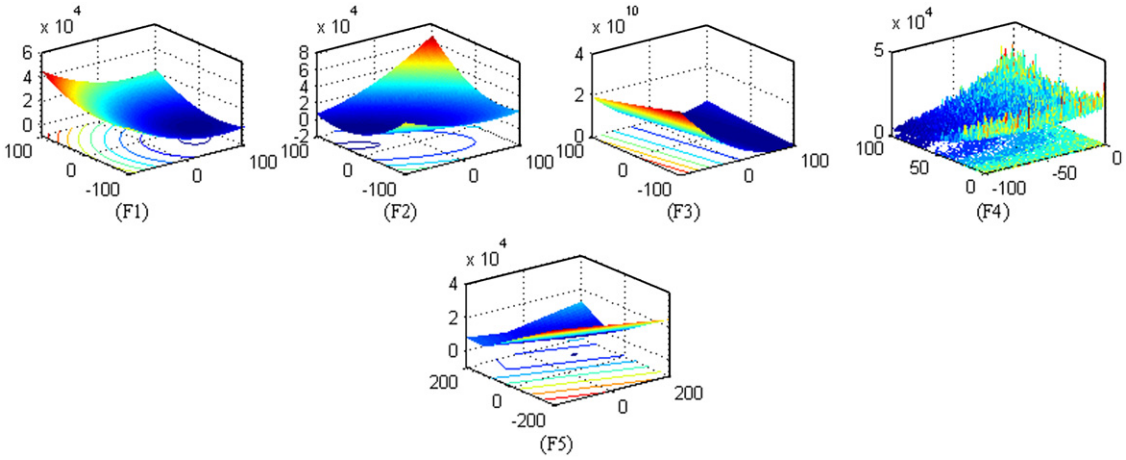

Fig. 3. 2-D versions of unimodal benchmark functions.

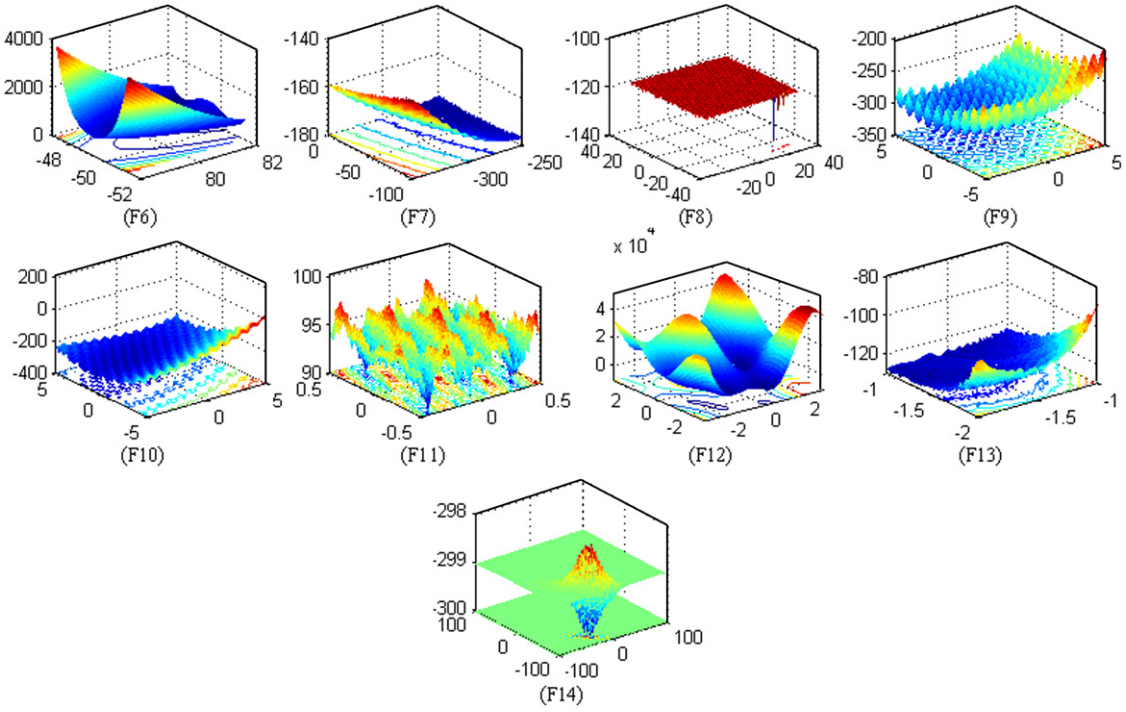

Fig. 4. 2-D versions of multimodal benchmark functions.

functions, and their group is also named the "v-shaped" family of transfer functions.

Three new v-shaped transfer functions named V1, V3, and V4 are proposed using diverse mathematical equations. These transfer functions are presented in Table 1 and Fig. 1(b). It may be seen that the saturation of V1 begins with lower absolute values of velocities compared to V2. This makes V1 able to provide higher probability of switching particles' positions than V2 for the same velocity. In contrast, the V3 and V4 transfer functions' saturations begin after V1 and V2 providing less probability of change for the same velocities.

The steps of utilizing both families of transfer functions to force particle motion in a binary search space are illustrated in Fig. 2. In the following section, the efficiency of these families is investigated.

5. Experimental results and discussion

In this section, the proposed transfer functions are compared with each other in order to evaluate them and select the best. Then a comparative study with six recent modifications of BPSO is

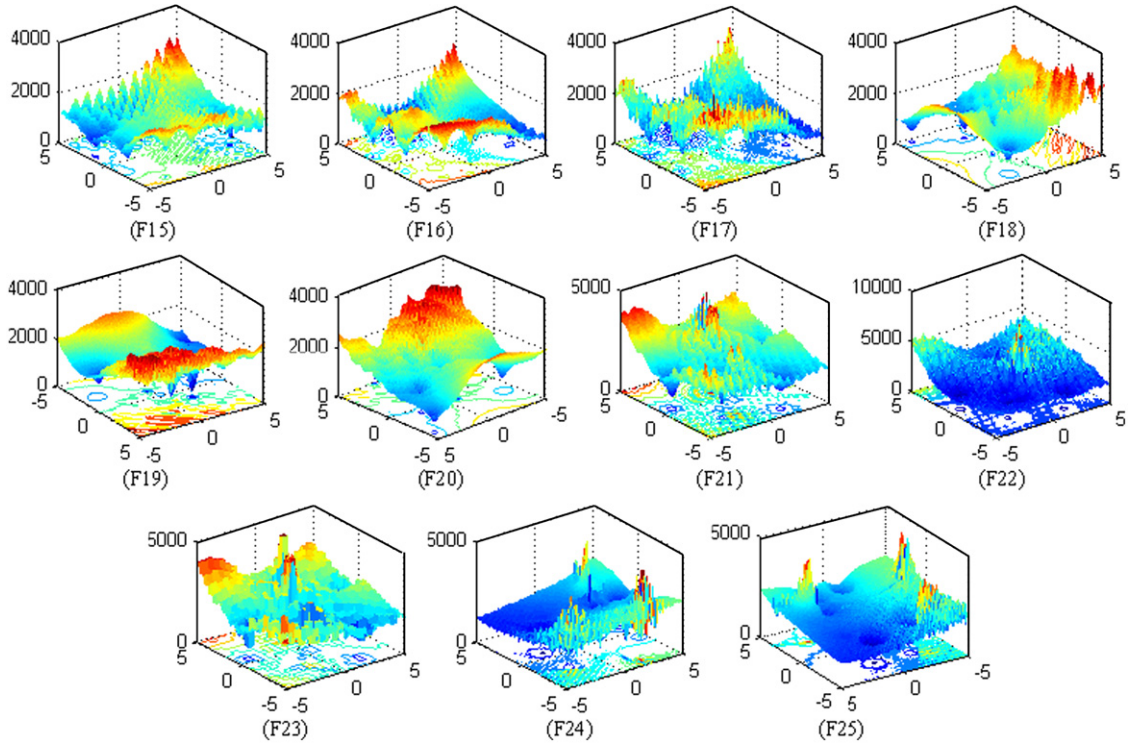

Fig. 5. 2-D versions of composite benchmark functions.

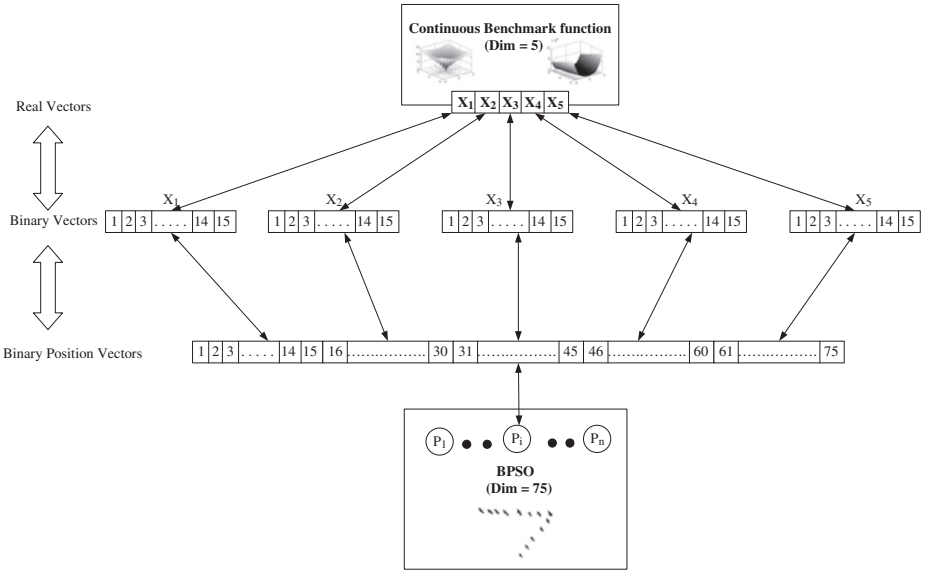

Fig. 6. Process of preparing variables of benchmark functions in binary for particles of BPSO and vice versa.

provided to validate the performance of the selected best transfer function.

In order to evaluate the performance of the proposed modified transfer function-based BPSO algorithms, 25 benchmark functions provided by CEC 2005 special session are employed [27]. These benchmark functions are the shifted, rotated, expanded, and combined variants of the classical functions which bring the highest complexity among the current benchmark functions [27,28]. The objective of the BPSO algorithms is to find the global minimum of them. These benchmark functions can be divided into three groups: unimodal, multimodal, and composite functions. Table 2 lists the aforementioned functions respectively, where *Dim* indicates dimension of the function, *Range* is the boundary of the function's search space, and f_{min} is the minimum return value of the function. Figs. 3–5 illustrate the 2-D versions of these groups of benchmark functions. A detailed description of the benchmark functions is available in the CEC 2005 technical report [27].

To represent each continuous variable in binary, 15 bits are used. It should be noticed that 1 bit is reserved for the sign of each functions' variable. Therefore, dimensions of the particles are calculated as follow:

$$Dim_{Particle} = Dim_{Function} \times 15 \tag{11}$$

where $Dim_{particle}$ indicates the dimension of each particle in BPSO and $Dim_{function}$ is the dimension of a particular benchmark function.

The process of preparing continuous variables of the benchmark functions in binary for particles of BPSO is shown in Fig. 6.

According to Eq. (11), the dimensions of the particles are 75 (5×15) for the unimodal and multimodal benchmark functions and 150 (10×15) for the composite benchmark functions.

There are some parameters in BPSO algorithms that should be defined. According to Eiben and Smit [29], there are two methods for choosing values for the parameters of evolutionary algorithms: parameter tuning and parameter control. In the former method, the

Table 3
Minimization results of the unimodal, multimodal, and composite benchmark functions over 30 independent runs.

	F1	F2	F3	F4	F5
BPSO1	−426.02 ± 33.81 (0.258)	−426.13 ± 33.86 (0.301)	−426.84 ± 33.55 (0.316)	−427.48 ± 38.94 (0.464)	−279.77 ± 35.75 (0.889)
BPSO2	−410.47 ± 34.38 (2.660)	−423.97 ± 23.74 (2.341)	−409.22 ± 40.22 (2.169)	−418.12 ± 27.66 (2.041)	−271.17 ± 39.24 (1.708)
BPSO3	−367.93 ± 48.82 (11.48)	−365.67 ± 51.70 (18.66)	−374.47 ± 38.29 (15.07)	−374.64 ± 41.95 (15.96)	−235.79 ± 51.22 (19.51)
BPSO4	−318.40 ± 69.29 (22.31)	−316.11 ± 60.22 (34.87)	−293.99 ± 69.12 (42.84)	−312.81 ± 63.12 (11.52)	−159.96 ± 70.80 (8.499)
BPSO5	−394.17 ± 74.42 (0.140)	−391.93 ± 72.50 (**0.016**)	−383.25 ± 99.99 (**0.011**)	−380.48 ± 85.21 (2.045)	−257.71 ± 67.75 (0.103)
BPSO6	−395.56 ± 69.45 (0.190)	−404.36 ± 100.8 (0.309)	−404.46 ± 46.04 (0.129)	−397.98 ± 69.89 (0.369)	−273.47 ± 47.12 (**0.017**)
BPSO7	−422.72 ± 34.12 (0.053)	−428.98 ± 27.10 (0.647)	−430.96 ± 29.62 (0.058)	−429.74 ± 34.87 (**0.109**)	−286.32 ± 29.58 (0.514)
BPSO8	**−433.26 ± 39.83 (0.020)**	**−429.61 ± 44.08** (0.120)	**−437.02 ± 23.65** (0.216)	**−435.21 ± 23.29** (0.128)	**−297.92 ± 23.01** (0.478)
	F6	F7	F8	F9	F10
BPSO1	**401.99 ± 11.67** (18.69)	−149.06 ± 30.06 (42.03)	**−119.91 ± 0.367** (24.13)	−309.903 ± 0.050 (25.3)	−309.91 ± 0.030 (24.41)
BPSO2	431.42 ± 38.10 (50.83)	−92.768 ± 50.51 (119.6)	−119.90 ± 0.464 (26.52)	−309.895 ± 0.045 (21.6)	−309.90 ± 0.042 (27.64)
BPSO3	458.64 ± 44.53 (101.9)	69.386 ± 121.6 (270.6)	−119.89 ± 0.053 (17.75)	−309.892 ± 0.049 (28.4)	−309.87 ± 0.040 (28.43)
BPSO4	534.19 ± 76.09 (147.7)	276.31 ± 286.7 (684.1)	−119.90 ± 0.037 (27.08)	−309.904 ± 0.037 (**10.3**)	−309.89 ± 0.042 (22.14)
BPSO5	453.62 ± 90.57 (70.92)	18.665 ± 554.9 (201.5)	−119.89 ± 0.041 (**10.37**)	−309.914 ± 0.044 (21.3)	−309.91 ± 0.035 (18.25)
BPSO6	420.20 ± 47.46 (38.94)	−64.102 ± 141.5 (168.6)	−119.89 ± 0.046 (25.71)	−309.913 ± 0.043 (14.5)	−309.91 ± 0.042 (14.59)
BPSO7	421.35 ± 49.32 (38.48)	−155.11 ± 41.69 (**26.86**)	−119.85 ± 0.043 (24.89)	**−309.918 ± 0.047** (25.9)	−309.91 ± 0.041 (**0.681**)
BPSO8	406.77 ± 30.53 (**12.70**)	**−158.97 ± 33.08** (27.98)	−119.90 ± 0.039 (18.01)	−309.908 ± 0.043 (15.9)	**−309.92 ± 0.041** (25.19)
	F11	F12	F13	F14	F15
BPSO1	92.949 ± 0.469 (1.311)	−457.11 ± 0.526 (1.012)	−127.10 ± 0.630 (1.352)	**−281.81 ± 18.10** (11.32)	283.94 ± 51.47 (113.8)
BPSO2	92.962 ± 0.391 (2.651)	−456.88 ± 0.559 (2.853)	−126.96 ± 0.387 (0.635)	−271.39 ± 19.60 (14.53)	265.74 ± 37.16 (84.62)
BPSO3	93.091 ± 0.409 (2.391)	−456.91 ± 0.486 (0.925)	−126.85 ± 0.442 (2.444)	−222.48 ± 44.72 (19.81)	348.96 ± 40.94 (175.1)
BPSO4	93.049 ± 0.562 (0.977)	−456.94 ± 0.536 (0.778)	−126.95 ± 0.330 (2.866)	−141.51 ± 67.44 (16.07)	408.03 ± 42.74 (237.1)
BPSO5	**92.395 ± 0.542 (0.008)**	−457.56 ± 0.549 (1.925)	−127.67 ± 0.601 (0.855)	−235.84 ± 92.98 (0.021)	**244.06 ± 51.49** (33.44)
BPSO6	92.455 ± 0.584 (0.463)	−457.45 ± 0.643 (0.263)	−127.64 ± 0.590 (1.724)	−261.37 ± 39.59 (33.42)	246.32 ± 60.01 (**2.227**)
BPSO7	92.489 ± 0.604 (2.365)	−457.55 ± 0.780 (0.152)	−127.73 ± 0.658 (**0.105**)	−277.59 ± 35.94 (18.91)	271.99 ± 32.42 (92.90)
BPSO8	92.516 ± 0.636 (0.021)	**−457.70 ± 0.667 (0.063)**	**−127.99 ± 0.532** (0.642)	−272.34 ± 44.50 (15.22)	254.45 ± 34.62 (53.32)
	F16	F17	F18	F19	F20
BPSO1	287.95 ± 95.23 (8.272)	264.35 ± 53.06 (8.191)	164.08 ± 61.57 (60.25)	179.44 ± 59.63 (103.4)	199.39 ± 45.66 (130.8)
BPSO2	270.05 ± 26.70 (120.1)	291.23 ± 34.18 (117.4)	193.82 ± 31.04 (127.9)	186.10 ± 27.47 (120.3)	179.77 ± 31.90 (138.5)
BPSO3	352.03 ± 48.56 (161.3)	366.73 ± 32.47 (172.7)	248.51 ± 40.90 (179.6)	246.82 ± 55.08 (155.6)	243.31 ± 22.73 (202.3)
BPSO4	402.79 ± 41.54 (236.3)	378.26 ± 39.94 (197.6)	289.67 ± 53.68 (217.5)	293.18 ± 24.99 (242.0)	276.24 ± 33.21 (217.3)
BPSO5	277.48 ± 23.89 (112.1)	251.35 ± 22.84 (95.50)	161.67 ± 93.95 (4.069)	165.81 ± 60.86 (85.22)	149.93 ± 45.40 (60.74)
BPSO6	382.74 ± 133.8 (88.76)	256.41 ± 65.79 (**1.885**)	**144.37 ± 55.07** (4.140)	160.77 ± 64.28 (**36.08**)	197.35 ± 67.46 (108.7)
BPSO7	**240.29 ± 57.68** (31.06)	**247.46 ± 59.94** (5.926)	169.28 ± 55.33 (36.88)	**155.07 ± 20.10** (114.7)	166.82 ± 66.37 (73.39)
BPSO8	248.66 ± 51.05 (**4.770**)	248.91 ± 47.56 (40.86)	161.04 ± 54.41 (74.12)	176.52 ± 75.19 (98.17)	**131.86 ± 43.93** (**24.92**)
	F21	F22	F23	F24	F25
BPSO1	909.46 ± 140.2 (216.7)	621.63 ± 45.45 (208.7)	923.79 ± 13.47 (557.0)	820.76 ± 9.334 (557.0)	**823.87 ± 13.51 (557.0)**
BPSO2	936.02 ± 86.43 (503.6)	694.02 ± 30.22 (283.4)	936.56 ± 40.47 (557.1)	863.79 ± 78.37 (557.0)	835.87 ± 23.50 (557.1)
BPSO3	1044.2 ± 70.16 (539.3)	774.14 ± 59.64 (334.6)	1063.3 ± 66.58 (627.9)	946.40 ± 91.48 (572.8)	897.24 ± 60.38 (557.2)
BPSO4	1117.3 ± 98.80 (571.8)	832.16 ± 46.64 (409.8)	1149.0 ± 100.5 (575.6)	1053.8 ± 113.5 (572.8)	995.89 ± 69.58 (645.7)
BPSO5	963.59 ± 180.6 (**201.4**)	712.98 ± 126.8 (205.1)	923.74 ± 11.55 (557.0)	885.86 ± 123.7 (557.1)	830.24 ± 17.93 (**557.0**)
BPSO6	926.96 ± 136.5 (390.4)	640.57 ± 104.1 (204.9)	942.28 ± 79.66 (557.0)	828.04 ± 93.22 (373.7)	942.09 ± 200.2 (557.1)
BPSO7	932.54 ± 167.1 (208.1)	609.25 ± 108.7 (**201.1**)	963.15 ± 155.2 (**376.9**)	**817.30 ± 0.725** (557.0)	846.91 ± 93.35 (**557.0**)
BPSO8	**905.57 ± 147.1** (203.7)	**591.75 ± 41.01 (201.1)**	**913.49 ± 130.1 (376.9)**	830.76 ± 115.1 (**373.7**)	829.25 ± 29.59 (557.1)

values of parameters are defined before starting the algorithms, and these parameters remain fixed until the end of iterations. In the latter method, the values of parameters are defined during execution of the algorithm. In this work the first method, parameter tuning, has been employed. The standard values recommended by Eberhart and Shi [30] are used in order to tune the parameters of BPSO. For all BPSO algorithms, the number of particles is equal to 30, C_1 and C_2 are set to 2, w is linearly decreased from 0.9 to 0.4, the maximum number of iterations is set to 500, and the stopping criterion is satisfied when the maximum number of iterations is

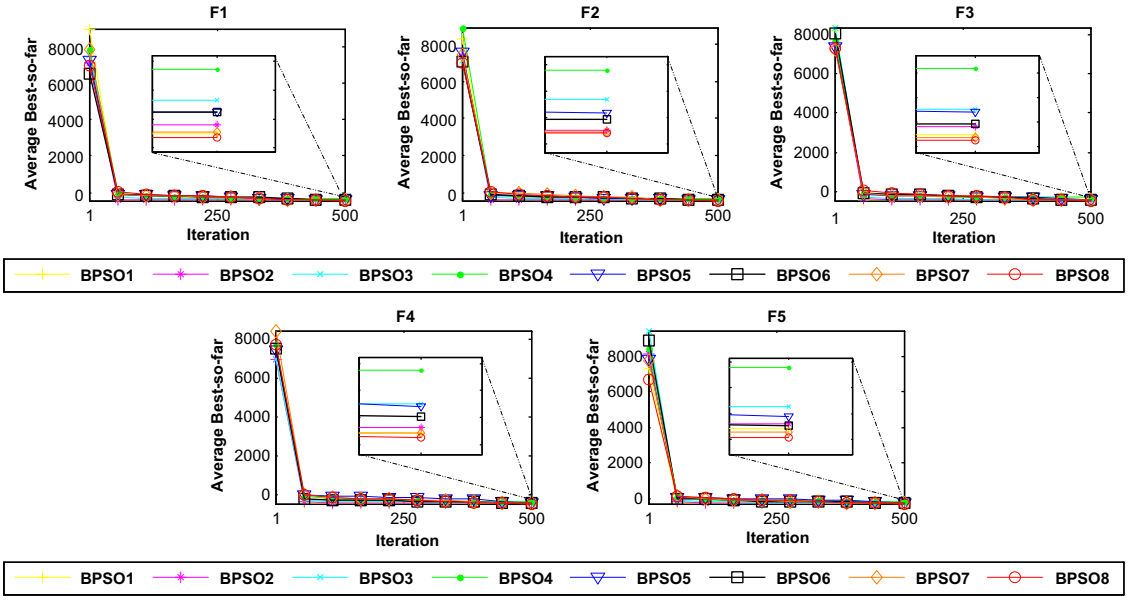

Fig. 7. Comparison between BPSO algorithms with different transfer functions on unimodal functions.

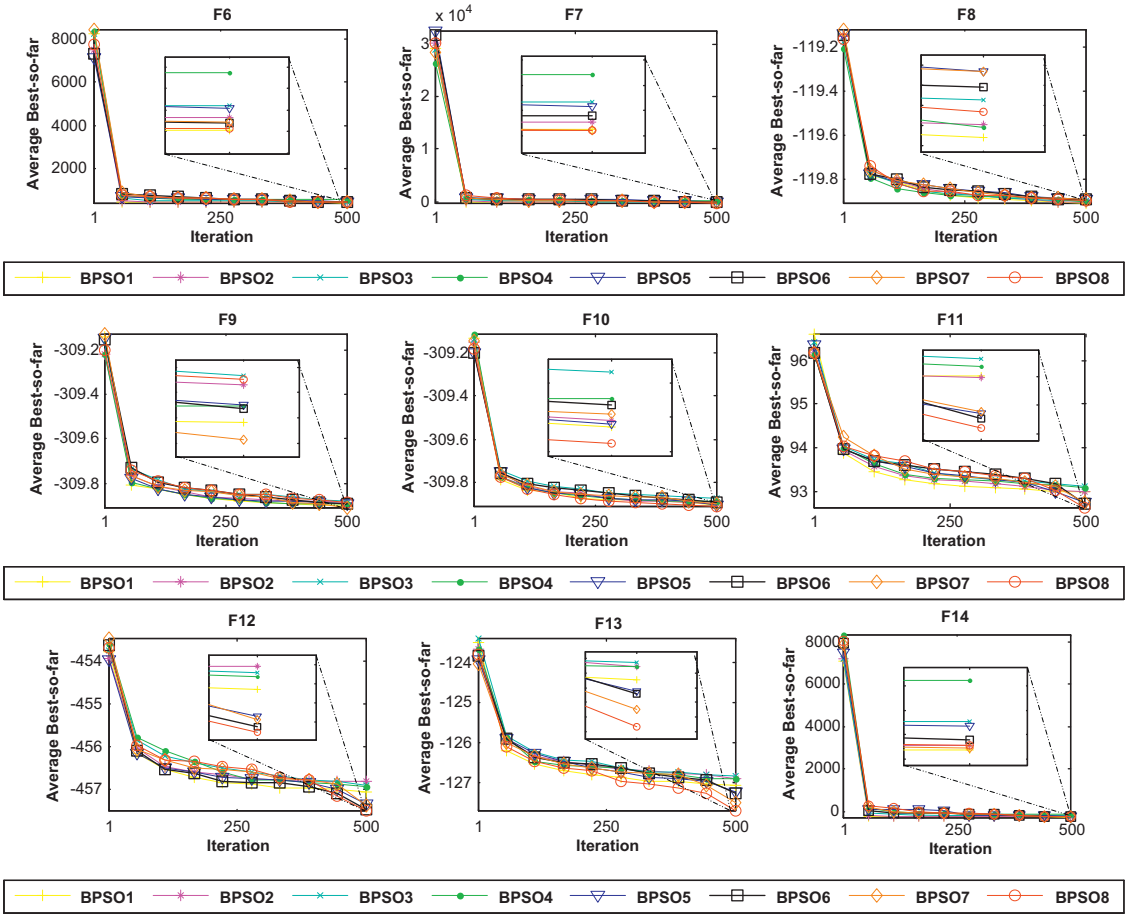

Fig. 8. Comparison between BPSO algorithms with different transfer functions on multimodal benchmark functions.

reached. Generally, the velocity should be limited to achieve a good convergence rate [30] so the maximum velocities of BPSO algorithms are set to 6. All transfer functions presented in Table 1 are applied to the BPSO and the resulting algorithms named BPSO1 to BPSO8. Thus, BPSO1 to BPSO4 use s-shaped transfer functions, and BPSO5 to BPSO8 use v-shaped transfer functions. The experimental results for all eight versions of BPSO are presented in Table 3. The results are averaged over 30 independent runs, and the best results are indicated in bold type. The average (AVE) and standard deviation (STD) of the best-so-far solution in the last iteration as well as the best obtained solution are reported in the form of AVE $\pm$ STD ($f(xbest)-f(x^*)$) where x^* is the global optimum.

According to Derrac et al. [31], to improve the evaluation of evolutionary algorithms' performance, statistical tests should be conducted. In other words, it is not enough to compare algorithms based on the mean and standard deviation values [32] and a statistical test is necessary to prove that a proposed new algorithm presents a significant improvement over other existing methods for a particular problem.

In order to judge whether the results of the s-shaped and v-shaped transfer functions differ from each other in a statistically significant way, a nonparametric statistical test, Wilcoxon's rank-sum test [33], was carried out at 5% significance level. The p-values calculated in the Wilcoxon's rank-sum comparing the algorithms over all the benchmark functions are given in Table 4. In this table, N/A indicates "not applicable" which means that the corresponding algorithm could not statistically compare with itself in the rank-sum test. According to Derrac et al. [31], those p-values that are less than 0.05 could be considered as strong evidence against the null hypothesis. In the following subsections the simulation results of each benchmark functions group are provided and discussed.

5.1. Unimodal benchmark functions

As may be seen from the results of F_1 to F_5 in Table 3, BPSO8 has the best results in all of the unimodal benchmark functions. The p-values in Table 4 also show that the results of BPSO5 to

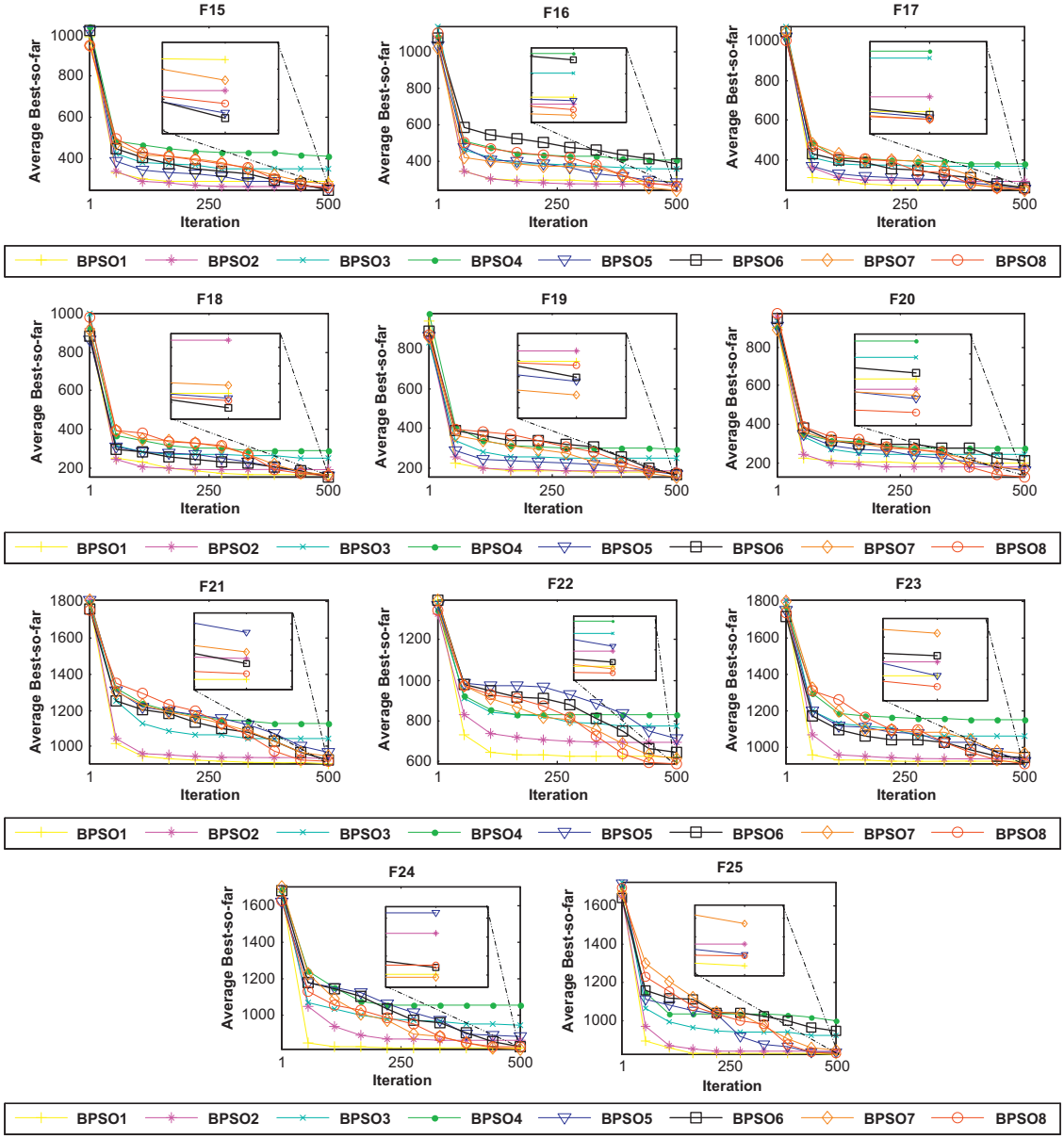

Fig. 9. Comparison between BPSO algorithms with different transfer functions on compiste benchmark functions.

BPSO8 are significantly better than BPSO1 to BPSO4. So these results show that the v-shaped family of transfer functions could improve the ability of BPSO in avoiding local minima for these benchmark functions. The best results of the BPSO algorithms with s-shaped transfer function are those of BPSO1.

Fig. 7 illustrates the convergence curves of all BPSO algorithms dealing with all unimodal benchmark functions. As can be seen from these curves, the convergence speed is slightly better for v-shaped transfer functions. Among BPSO algorithms with s-shaped transfer functions, BPSO1, which uses S1, has the fastest convergence rate.

To summarize, according to Tables 3 and 4, and Fig. 7, the BPSO algorithms using transfer functions from the v-shaped family tend to find the best solution in unimodal functions with a fast convergence rate.

5.2. Multimodal benchmark functions

The results of the multimodal benchmark functions (F_6 to F_{14}) are provided in the second and third main rows of Table 3. It should be noticed that these benchmark functions have many local minima with the number increasing exponentially with dimension. As the results show, v-shaped transfer functions perform better than the other transfer functions in most of the multimodal benchmark functions. The only benchmark functions on which v-shaped transfer functions are not able to outperform s-shaped transfer functions are F_6, F_8, and F_{14}, but the results of these two families are very close. The p-values in Table 4 also show that the significant results belong to the v-shaped family of transfer functions. These results show that the v-shaped family is better than the s-shaped family in terms of avoiding local minima for multimodal functions.

As may be observed in Fig. 8, BPSO algorithms with s-shaped and v-shaped transfer functions have distinct patterns of convergence. The s-shaped family has a steady convergence behavior, while v-shaped has an accelerated convergence. The best convergence curve belongs to BPSO8 for most of the benchmark functions, followed by BPSO7, BPSO6, and BPSO5. According to these results, the convergence speed of BPSO reduces for the v-shaped transfer functions with early saturation. In contrast, the s-shaped family has the opposite convergence behavior.

It is clear from the convergence curves of BPSO1 to BPSO4 that the convergence speed of BPSO decreases when the slope of the transfer function declines. In other words, the s-shaped transfer functions with early saturation converge much faster.

Once again it is clear that the BPSO algorithms using the v-shaped family of transfer functions (especially V4) are capable of finding the best solution with good convergence rate for multimodal benchmark functions.

5.3. Composite benchmark functions

The results of the composite benchmark functions are provided in the last two main rows of Table 3 and Fig. 9. The results of BPSO5 to BPSO8 appear better than BPSO1 to BPSO4 on all the composite benchmark functions except F_{25}. These results indicate that v-shaped transfer functions are useful for the BPSO algorithm in solving these benchmark functions as well. As shown in Fig. 5, the composition functions have extremely complex structures with many local minima, similar to real world problems. Hence, the results of Tables 3 and 4 strongly suggest that the v-shaped family of transfer functions enhances the performance of BPSO in terms of avoiding local minima. The BPSO7 and BPSO8 algorithms have very close results, and each of them has the best results for four composite functions. Among the BPSO algorithms with s-shaped transfer functions, BPSO1 performs better for all composite benchmark functions, providing p-values greater than 0.05 for four of the composite functions.

Fig. 9 depicts the convergence curves of the BPSO algorithms on composite benchmark functions. As can be observed in this figure,

Table 4
P-values calculated for Wilcoxon's rank-sum test on unimodal, multimodal, and composite benchmark functions.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
BPSO1	0.00384	0.01501	0.006972	**0.14127**	0.00081	N/A	0.01441	N/A	**0.2643**	**0.1219**
BPSO2	1.73e-06	0.00023	8.29E-06	0.00047	1.33E-05	1.24e-05	2.01e-08	**0.2339**	**0.0574**	0.0748
BPSO3	9.06e-08	1.06e-07	2.23E-09	9.26E-09	6.51E-09	1.85e-09	8.15e-11	**0.2458**	0.0292	5.61e-05
BPSO4	3.82e-09	4.18e-09	1.47e-10	9.92e-11	1.21e-10	4.97e-11	4.97e-11	**0.2009**	**0.0993**	0.0067
BPSO5	0.00085	0.001173	0.000620	0.00130	0.00055	**0.05368**	4.63e-05	0.04358	**0.6521**	**0.2772**
BPSO6	0.00168	**0.105469**	0.000398	0.01532	0.01221	**0.85338**	9.79e-05	**0.1809**	**0.6308**	**0.3403**
BPSO7	0.01383	0.093340	0.082357	**0.52978**	0.05187	**0.69521**	**0.99410**	**0.1023**	N/A	**0.2973**
BPSO8	N/A	N/A	N/A	N/A	N/A	**0.04358**	N/A	**0.3871**	**0.2643**	N/A
	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
BPSO1	0.00017	7.66e-05	2.49e-06	N/A	0.0168	0.0337	**0.1853**	**0.3549**	0.0337	1.06e-08
BPSO2	3.83e-05	2.68e-06	5.00e-09	0.01076	**0.1853**	0.0053	0.0014	3.76e-05	2.09e-05	2.09e-05
BPSO3	2.32e-06	3.83e-06	2.92e-09	3.96e-08	1.69e-10	9.35e-10	1.69e-10	9.35e-10	9.35e-10	2.87e-11
BPSO4	5.09e-06	3.81e-07	1.17e-09	3.68e-11	2.87e-11	2.87e-11	1.69e-10	7.04e-11	2.87e-11	2.87e-11
BPSO5	N/A	**0.27719**	0.01628	**0.29047**	N/A	2.09e-05	**0.8998**	**1**	**0.6949**	**0.1449**
BPSO6	**0.58945**	**0.78446**	0.02236	**0.26432**	**0.8999**	3.24e-06	**0.7956**	N/A	**0.0847**	8.91e-04
BPSO7	**0.57929**	**0.25189**	**0.14128**	**0.05187**	0.0337	N/A	N/A	0.0116	N/A	**0.632**
BPSO8	**0.61001**	N/A	N/A	**0.239849**	**0.5993**	**0.2335**	**0.7956**	**0.2899**	**0.5101**	N/A
	F21	F22	F23	F24	F25					
BPSO1	**0.3183**	5.48e-04	**0.0959**	**0.3871**	N/A					
BPSO2	0.0108	2.14e-09	1.95e-04	9.84e-09	0.0014					
BPSO3	6.05e-07	2.87e-11	1.01e-07	2.59e-11	4.92e-08					
BPSO4	6.01e-08	2.87e-11	2.25e-08	2.59e-11	2.87e-11					
BPSO5	0.0905	1.14e-05	0.0388	0.0091	**0.6938**					
BPSO6	**0.7958**	0.0079	**0.2572**	2.41e-04	**0.5096**					
BPSO7	**0.3871**	**0.6949**	**0.1624**	N/A	**0.0624**					
BPSO8	N/A	N/A	N/A	**0.5962**	**0.2308**					

the BPSO algorithms with s-shaped transfer functions have steady and monotonous convergence curves similar to the previous benchmark functions. However, the v-shaped transfer functions give BPSO an accelerated convergence behavior. This accelerated convergence helps the BPSO algorithms with v-shaped transfer functions to surpass the BPSO algorithms with s-shaped transfer functions after completing almost half of the iterations (250). The effects of slopes and saturations of the s-shaped and v-shaped transfer functions are also similar to the previous benchmark functions, i.e. the v-shaped transfer functions saturated with lower absolute values of velocities have slower convergence while the s-shaped transfer functions with early saturation converge much faster than those with late saturation.

According to Table 3 and 4, and Fig. 9, it can be stated that algorithms using the v-shaped family of transfer functions is capable of finding global solution with good convergence speed in dealing with composite benchmark functions.

To summarize, statistically speaking, the BPSO algorithms with the v-shaped family of transfer functions have the best results for twenty-one out of twenty-five benchmark functions, showing significant improvement in eleventh of them. In contrast, the BPSO algorithms using the s-shaped family transfer functions have the best results on only four benchmark functions, giving significant results in none of them. Among all algorithms, the BPSO8 algorithm with V4 transfer function performs better in 13 out of 25 benchmark functions; around half of the test functions. Therefore, these results show that the family of s-shaped transfer functions with their method of position updating is not suitable for the BPSO algorithm. However, the proposed new v-shaped family of transfer functions with their special method of updating positions is useful for the BPSO algorithm in terms both of avoiding local minima and convergence rate.

In the following subsection, a comparative study with some recent modifications of BPSO is provided in order to validate the performance of the proposed new approach.

5.4. Comparative study

In this section, the BPSO8 algorithm which showed the highest performance among the proposed modified transfer function-based BPSO algorithms is named VPSO and compared with some of the most recent, modified versions of BPSO algorithms such as MPSO1 [19], PBSO [20], and MPSO2 [21]. In order to provide a comprehensive comparative study, some recent high performance versions of continuous PSO such as CLPSO [34], FIPS [35], and DMS-PSO [36,37] are also converted to binary algorithms and compared with the VPSO algorithm. These new binary algorithms, which use both s-shaped and v-shaped transfer functions, have been equipped with S2 and V4 transfer functions to be able to work in binary spaces. Those using the S2 transfer function are named MBPSO1, MBPSO2, BCLPSO, BFIPS, and BDMS-PSO, whereas those using V4 are named MVPSO1, MVPSO2, VCLPSO, VFIPS, and VDMS-PSO. It should be noted that PBSO does not use transfer functions. The minimization and statistical results of

Table 5
Comparison of VPSO, PBSO, MBPSO1, MBPSO2, BCLPSO, BFIPS, and BDMS-PSO on all benchmark functions over 30 independent runs.

	F1	F2	F3	F4	F5
PBSO [20]	−400.77 ± 32.66 (5.04)	−302.88 ± 91.22 (15.7)	−290.51 ± 32.55 (32.7)	−299.11 ± 45.65 (1.19)	−211.66 ± 98.55 (11.3)
MBPSO1 [19]	−426.99 ± 18.34 (2.34)	−297.79 ± 52.43 (44.6)	−201.32 ± 54.33 (55.5)	−247.73 ± 90.22 (11.8)	−225.35 ± 24.14 (5.12)
MBPSO2 [21]	−425.97 ± 13.11 (0.88)	−408.33 ± 301.7 (0.79)	−320.58 ± 77.12 (11.2)	−395.71 ± 436.0 (2.51)	−201.81 ± 24.57 (22.4)
BCLPSO [32]	−420.41 ± 48.84 (1.02)	−311.36 ± 114.7 (21.4)	−333.22 ± 46.15 (1.07)	−301.40 ± 96.97 (5.32)	−220.69 ± 30.01 (1.85)
BFIPS [33]	−419.14 ± 40.02 (4.30)	−345.23 ± 45.81 (22.5)	−350.51 ± 30.03 (22.9)	−346.25 ± 41.07 (0.68)	−248.44 ± 87.94 (0.97)
BDMS − PSO [34,35]	−428.98 ± 43.88 (0.81)	−410.54 ± 71.98 (1.49)	−402.29 ± 29.67 (0.27)	−421.88 ± 22.53 (0.88)	−255.65 ± 32.67 (**0.41**)
	F6	F7	F8	F9	F10
PBSO	501.55 ± 27.87 (99.5)	−100.25 ± 45.33 (**1.68**)	−119.83 ± 0.072 (10.5)	−300.79 ± 1.787 (29.1)	−298.25 ± 1.361 (**19.5**)
MBPSO1	532.47 ± 34.11 (87.7)	−90.155 ± 14.63 (44.6)	−119.91 ± 0.042 (22.2)	−307.58 ± 0.288 (12.3)	−299.07 ± 2.782 (22.1)
MBPSO2	412.01 ± 92.38 (0.99)	−128.74 ± 20.39 (3.89)	−119.90 ± 0.045 (31.2)	−308.53 ± 2.205 (24.1)	−303.18 ± 1.919 (26.8)
BCLPSO	483.68 ± 19.43 (1.42)	−104.59 ± 17.67 (11.3)	−119.88 ± 0.063 (28.9)	−301.45 ± 1.572 (11.6)	−300.83 ± 2.452 (13.3)
BFIPS	450.32 ± 30.18 (**0.58**)	−129.49 ± 18.88 (2.05)	−119.87 ± 0.038 (**9.74**)	−304.91 ± 1.476 (26.5)	−301.36 ± 2.728 (19.6)
BDMS-PSO	410.44 ± 65.87 (21.4)	−130.23 ± 22.15 (9.48)	**−119.98 ± 0.092** (19.1)	**−315.97 ± 4.867 (7.77)**	−304.72 ± 2.627 (22.7)
	F11	F12	F13	F14	F15
PBSO	93.154 ± 0.624 (2.96)	−301.72 ± 0.813 (19.3)	−127.24 ± 0.732 (1.55)	−270.68 ± 92.72 (11.6)	513.13 ± 100.1 (221)
MBPSO1	93.130 ± 0.508 (1.67)	−313.25 ± 0.132 (15.3)	−127.44 ± 0.192 (1.98)	−298.67 ± 13.72 (0.43)	320.67 ± 127.4 (98.2)
MBPSO2	93.135 ± 0.364 (0.89)	−432.17 ± 0.710 (6.04)	−129.60 ± 0.738 (0.30)	−298.42 ± 20.14 (**0.19**)	644.49 ± 53.50 (323)
BCLPSO	93.164 ± 0.621 (0.30)	−312.10 ± 0.440 (22.6)	−128.59 ± 0.341 (1.08)	−298.48 ± 47.11 (0.25)	493.09 ± 88.88 (84.1)
BFIPS	93.148 ± 0.279 (1.53)	−352.32 ± 0.141 (8.82)	−129.11 ± 0.236 (**0.03**)	**−298.50 ± 97.12** (0.53)	350.98 ± 54.43 (77.2)
BDMS-PSO	**92.481 ± 0.612** (0.93)	−440.57 ± 0.835 (4.36)	**−129.52 ± 0.521** (0.08)	−298.78 ± 56.87 (0.81)	302.46 ± 43.26 (65.8)
	F16	F17	F18	F19	F20
PBSO	353.36 ± 34.75 (37.7)	397.24 ± 62.74 (55.5)	370.23 ± 111.1 (241)	432.35 ± 100.1 (153)	410.23 ± 53.23 (291)
MBPSO1	308.03 ± 28.06 (38.6)	325.17 ± 26.81 (62.4)	282.39 ± 147.6 (88.4)	358.28 ± 129.4 (119)	395.72 ± 221.6 (211)
MBPSO2	426.27 ± 67.34 (84.9)	494.46 ± 70.06 (95.3)	331.14 ± 176.4 (241)	416.10 ± 146.2 (361)	300.18 ± 176.8 (169)
BCLPSO	389.08 ± 54.67 (83.9)	379.45 ± 42.92 (77.3)	323.06 ± 173.2 (199)	441.28 ± 120.2 (228)	393.76 ± 192.8 (272)
BFIPS	326.89 ± 30.48 (37.2)	342.02 ± 60.91 (83.0)	205.46 ± 218.7 (78.5)	393.56 ± 115.4 (249)	368.02 ± 221.5 (122)
BDMS-PSO	290.15 ± 43.35 (17.3)	**220.52 ± 43.47 (11.6**)	180.28 ± 120.1 (95.2)	261.46 ± 56.32 (**53.1**)	198.12 ± 352.2 (57.3)
	F21	F22	F23	F24	F25
PBSO	1235.3 ± 124.1 (500)	1025.24 ± 194.2 (579)	1145.3 ± 123.3 (534)	980.36 ± 299.27 (348)	1006.2 ± 230.2 (610)
MBPSO1	1001.3 ± 167.0 (672)	1019.46 ± 229.8 (655)	1011.8 ± 164.6 (463)	938.48 ± 309.21 (387)	976.39 ± 250.1 (597)
MBPSO2	1491.9 ± 84.34 (723)	1172.67 ± 75.47 (472)	1521.4 ± 64.79 (511)	1206.5 ± 181.79 (527)	1135.4 ± 141.5 (763)
BCLPSO	1191.4 ± 88.68 (844)	1202.98 ± 133.5 (602)	1250.2 ± 99.36 (451)	1064.4 ± 153.59 (514)	1055.7 ± 224.4 (743)
BFIPS	1101.5 ± 70.07 (652)	1217.70 ± 166.1 (518)	1045.1 ± 66.05 (388)	882.08 ± 220.17 (351)	865.32 ± 240.6 (623)
BDMS-PSO	972.34 ± 122.2 (366)	**530.456 ± 65.34 (187**)	**860.34 ± 45.34** (392)	833.34 ± 122.35 (**370**)	844.35 ± 210.2 (592)

Table 6
P-values calculated for Wilcoxon's rank-sum test on unimodal, multimodal, and composite benchmark functions.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
VPSO	N/A	N/A	N/A	N/A	N/A	N/A	N/A	**0.5823**	**0.8833**	NA
PBSO	2.70e-11	2.91e-11	2.91e-11	2.91e-11	2.91e-11	2.91e-11	5.34e-09	0.0232	**0.0542**	0.0009
MBPSO1	5.43e-05	3.29e-11	2.91e-11	5.41e-08	2.91e-11	2.91e-11	2.91e-11	0.0168	0.0237	0.0082
MBPSO2	3.55e-05	5.48e-05	2.91e-11	2.91e-11	4.90e-07	2.43e-03	7.41e-06	**0.3754**	**0.7327**	0.0291
BCLPSO	3.01e-05	2.91e-11	2.91e-11	2.91e-11	2.91e-11	2.91e-11	7.73e-08	**0.2198**	**0.5173**	0.0346
BFIPS	1.65e-05	2.91e-11	2.91e-11	2.91e-11	6.344-06	3.34e-08	5.33e-06	**0.2536**	**0.5278**	0.0426
BDMS-PSO	7.43e-02	3.24e-04	6.71e-05	7.87e-05	2.12e-04	**0.0621**	**2.13e-04**	N/A	**N/A**	0.0111
	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
VPSO	**0.08315**	N/A	6.23e-08	2.91e-11	N/A	N/A	3.52e-05	N/A	N/A	N/A
PBSO	1.72e-06	2.91e-11	3.81e-09	2.91e-11	2.91e-11	2.91e-11	2.91e-11	2.91e-11	2.91e-11	2.91e-11
MBPSO1	1.04e-06	2.91e-11	9.15e-08	**0.0684**	7.43e-08	8.34e-07	6.45e-09	2.91e-11	2.91e-11	2.91e-11
MBPSO2	8.41e-05	2.3e-05	0.0094	**0.0564**	2.91e-11	2.91e-11	2.91e-11	2.91e-11	2.91e-11	2.91e-11
BCLPSO	2.46e-06	2.91e-11	6.24e-07	0.0384	6.34e-09	9.85e-10	2.91e-11	2.91e-11	2.91e-11	2.91e-11
BFIPS	4.85e-06	2.91e-11	0.0075	N/A	2.32e-08	9.35e-07	2.91e-11	2.91e-11	2.91e-11	2.91e-11
BDMS-PSO	N/A	1.95e-04	N/A	**0.0753**	7.23e-07	2.22e-06	N/A	0.0001	7.53e-05	2.52e-04
	F21	F22	F23	F24	F25					
VPSO	N/A	0.0213	7.92e-04	N/A	N/A					
PBSO	2.91e-11	2.91e-11	2.91e-11	1.05e-11	2.91e-11					
MBPSO1	8.34e-08	2.91e-11	2.91e-11	9.34e-10	6.34e-07					
MBPSO2	2.91e-11	2.91e-11	2.91e-11	2.91e-11	2.91e-11					
BCLPSO	4.23e-06	2.91e-11	2.91e-11	2.91e-11	2.91e-11					
BFIPS	3.23e-06	2.91e-11	2.91e-11	4.57e-05	9.24e-05					
BDMS-PSO	**7.34e-04**	**N/A**	**N/A**	0.1973	**3.23e-04**					

Table 7
Comparison of VPSO PBSO, MVPSO1, MVPSO2, VCLPSO, VFIPS, and VDMS-PSO, on all benchmark functions over 30 independent runs.

	F1	F2	F3	F4	F5
MVPSO1	−428.01 ± 54.08 (0.75)	−300.72 ± 80.10 (22.8)	−300.48 ± 27.12 (12.1)	−298.49 ± 22.46 (22.4)	−287.11 ± 27.45 (2.29)
MVPSO2	−429.89 ± 37.21 (3.48)	−415.13 ± 29.66 (0.76)	−377.88 ± 77.43 (1.34)	−410.81 ± 40.71 (0.18)	**−302.46 ± 42.98** (0.74)
VCLPSO	−427.55 ± 17.74 (1.22)	−417.06 ± 35.32 (1.51)	−407.41 ± 24.83 **(0.02)**	−388.60 ± 40.35 (12.2)	−275.21 ± 12.76 (5.98)
VFIPS	−423.77 ± 13.17 (2.74)	−421.57 ± 18.46 (0.82)	−420.89 ± 25.05 (1.57)	−399.34 ± 18.86 (9.24)	−296.48 ± 16.58 (1.82)
VDMS-PSO	−430.09 ± 32.43 (0.09)	**−429.91 ± 23.64 (0.08)**	−431.29 ± 44.65 (0.07)	−435.10 ± 33.78 (0.98)	−300.84 ± 55.43 **(0.25)**
	F6	F7	F8	F9	F10
MVPSO1	422.03 ± 27.92 (1.37)	−146.82 ± 86.74 (8.55)	−119.92 ± 0.0327 (12.6)	−321.81 ± 2.359 (1.39)	−315.05 ± 2.836 (9.38)
MVPSO2	407.35 ± 23.69 (8.38)	−155.10 ± 92.88 **(2.40)**	−119.88 ± 0.0283 (35.1)	−318.43 ± 3.497 (7.59)	−313.05 ± 4.097 (10.6)
VCLPSO	452.18 ± 17.84 (12.5)	−106.83 ± 124.6 (76.5)	−119.92 ± 0.0251 (29.3)	**−327.62 ± 0.404 (0.04)**	**−322.26 ± 2.549** (2.22)
VFIPS	411.80 ± 49.64 (0.62)	−155.37 ± 17.86 (3.78)	−119.90 ± 0.0317 **(11.8)**	−320.66 ± 2.203 (11.4)	−317.11 ± 4.173 **(0.82)**
VDMS-PSO	406.76 ± 94.47 **(0.52)**	−154.89 ± 43.75 (3.37)	**−119.95 ± 0.0536** (33.3)	−319.74 ± 3.464 (16.1)	−309.78 ± 2.753 (22.2)
	F11	F12	F13	F14	F15
MVPSO1	92.243 ± 0.331 (2.53)	−418.06 ± 51.06 (0.10)	−129.23 ± 0.2793 (0.07)	−298.51 ± 0.144 (0.90)	317.753 ± 87.51 (66.9)
MVPSO2	92.064 ± 0.421 (1.21)	−447.97 ± 72.35 (0.26)	−129.03 ± 0.7208 (0.01)	−298.46 ± 0.115 (0.21)	577.509 ± 74.78 (391)
VCLPSO	92.304 ± 0.392 (0.78)	−350.14 ± 74.76 (11.3)	−129.62 ± 0.1062 **(0.02)**	**−298.59 ± 0.126 (0.01)**	273.599 ± 48.53 (58.3)
VFIPS	**91.887 ± 0.484** (0.04)	−413.69 ± 40.87 (9.58)	−129.62 ± 0.3574 (0.06)	−298.48 ± 0.126 (0.87)	319.279 ± 94.28 (89.2)
VDMS-PSO	91.687 ± 0.314 (1.18)	−450.56 ± 31.98 (0.35)	**−129.78 ± 0.4268** (0.05)	−298.36 ± 0.149 (0.64)	278.373 ± 54.26 (66.7)
	F16	F17	F18	F19	F20
MVPSO1	292.28 ± 47.51 (6.11)	300.222 ± 47.40 (65.6)	257.353 ± 151.8 (126)	212.974 ± 292.1 (188)	370.030 ± 165.5 (111)
MVPSO2	405.25 ± 58.18 (43.6)	459.047 ± 56.78 (85.5)	241.371 ± 89.56 (201)	330.967 ± 181.7 (261)	206.797 ± 176.2 (135)
VCLPSO	287.41 ± 26.89 (26.3)	282.304 ± 33.29 (11.3)	257.285 ± 161.8 (197)	324.124 ± 121.8 (105)	275.621 ± 221.3 (168)
VFIPS	299.40 ± 57.11 (38.4)	313.040 ± 45.70 (21.1)	188.165 ± 151.1 (57.7)	287.613 ± 171.3 (246)	215.715 ± 220.8 (91.6)
VDMS-PSO	272.65 ± 27.62 (53.3)	**202.354 ± 56.10 (8.08)**	169.022 ± 99.32 **(22.3)**	211.546 ± 111.8 **(88.7)**	133.600 ± 211.8 (83.2)
	F21	F22	F23	F24	F25
MVPSO1	993.01 ± 100.5 (237)	569.508 ± 209.6 (226)	893.157 ± 112.29 (309)	829.363 ± 293.5 (393)	932.305 ± 206.8 (588)
MVPSO2	1194.5 ± 110.3 (443)	554.682 ± 151.3 (238)	948.291 ± 81.471 (392)	983.605 ± 151.6 (573)	965.620 ± 166.0 (634)
VCLPSO	918.94 ± 25.16 (274)	599.745 ± 254.6 (297)	823.302 ± 12.055 (323)	814.276 ± 248.9 (306)	938.973 ± 194.9 (645)
VFIPS	976.25 ± 98.12 (301)	429.633 ± 49.72 **(138)**	725.195 ± 160.31 (396)	805.557 ± 175.6 (383)	836.454 ± 111.3 (508)
VDMS-PSO	911.58 ± 83.42 **(179)**	**419.228 ± 55.56** (165)	**723.049 ± 33.664 (283)**	**781.399 ± 111.4 (332)**	825.739 ± 189.4 **(558)**

these algorithms and VPSO are provided in Tables 5–8 for unimodal, multimodal, and composite benchmark functions. The best results are highlighted in boldface.

As can be seen in Table 5, the VPSO algorithm outperforms the other s-shaped based algorithms on all the unimodal benchmark functions in terms of the mean and standard deviation of the results.

Table 8
P-values calculated for Wilcoxon's rank-sum test on unimodal, multimodal, and composite benchmark functions.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
VPSO	N/A	**0.0958**	N/A	N/A	0.0355	N/A	N/A	**0.8463**	0.0036	0.0064
MVPSO1	0.0002	2.91e-11	2.91e-11	2.91e-11	5.24e-05	8.44e-05	5.78e-05	**0.6274**	**0.0699**	0.0151
MVPSO2	0.0003	4.72e-04	8.05e-09	8.24e-05	N/A	**0.3167**	0.0074	**0.6205**	0.0404	0.0139
VCLPSO	2.66e-05	7.42e-04	3.63e-06	4.47e-07	6.69e-06	3.55e-08	8.33e-09	**0.7274**	N/A	N/A
VFIPS	4.12e-05	**0.0735**	5.72e-04	5.33e-07	0.0009	0.0048	0.0071	**0.8735**	**0.0538**	0.0284
VDMS-PSO	0.0017	N/A	0.0241	**0.0282**	**0.0976**	**0.5884**	0.0068	N/A	0.0453	0.0055
	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
VPSO	0.0024	N/A	8.55e-04	3.73e-05	N/A	N/A	7.45e-06	N/A	N/A	N/A
MVPSO1	0.0085	2.84e-06	**0.0944**	**0.4240**	6.75e-06	9.43e-05	3.67e-07	2.63e-04	1.76e-04	4.85e-08
MBVPSO2	0.0247	0.0023	**0.0863**	**0.3905**	2.91e-11	2.91e-11	2.91e-11	5.72e-06	3.46e-09	0.0022
VCLPSO	0.0061	2.91e-11	**0.3452**	N/A	0.0008	3.04e-04	5.72e-05	2.13e-04	6.31e-09	4.30e-04
VFIPS	N/A	7.08e-06	**0.2839**	**0.4297**	1.55e-06	5.86e-05	**0.0630**	0.0056	7.08e-05	0.0045
VDMS-PSO	**0.1984**	0.0057	N/A	**0.2628**	0.0005	5.46e-04	N/A	**0.1501**	5.35e-04	**0.0935**
	F21	F22	F23	F24	F25					
VPSO	N/A	4.24e-04	7.34e-04	2.34e-06	N/A					
MVPSO1	4.76e-07	7.56e-05	5.45e-05	5.11e-06	5.01e-06					
MVPSO2	2.91e-11	3.22e-05	7.45e-07	6.44e-09	6.34e-07					
VCLPSO	0.0232	2.91e-11	9.44e-04	3.66e-05	2.90e-06					
VFIPS	7.39e-07	**0.0687**	**0.0599**	7.34e-05	**0.0794**					
VDMS-PSO	**0.0624**	N/A	N/A	N/A	**0.1663**					

According to the *p*-values in Table 6, VPSO offers much significant improvement in all the unimodal benchmark functions in comparison with the other algorithms. Therefore, this proves that the proposed v-shaped family of transfer functions could significantly improve the performance of BPSO in finding the global solution of unimodal benchmark functions.

According to results of F_6 to F_{14} in Table 5, the VPSO algorithm has the best performance among the algorithms on four of the multimodal benchmark functions. The *p*-values of F_6 to F_{14} in Table 6 indicate significantly improved performance for the VPSO algorithm in two of the multimodal functions compared to the other algorithms as well. So it appears that the VPSO algorithm with its transfer function (V4) can be better than the conventional BPSO in avoiding local minima.

As reflected in the remaining functions of Table 5, the recorded results on the composite benchmark function are variable. The VPSO algorithm shows better results in eight out of 11 composite benchmark functions. The BDMS-PSO algorithm outperforms the other algorithms on the F_{17}, F_{22}, and F_{23} benchmark functions. In addition, as shown in Table 6, the VPSO algorithm yields significantly greater improvements for seven of the composite benchmark functions than the other algorithms.

The results of Tables 7 and 8 indicate that the performance of all the BPSO algorithms is improved with the v-shaped transfer function. VPSO algorithm has the best results for F_1, F_3, F_4, F_6, F_7, F_{12}, F_{15}, F_{16}, F_{18}, F_{19}, F_{20}, F_{21}, and F_{25}. It is worth mentioning that the results of Table 8 vary in comparison with Table 6. This is the outcome of increasing the performance of all algorithms, which results in rejection of the null hypothesis in 16 of the functions.

Overall, it is evident that the VPSO algorithm has the best results (considering *p*-values) on fourteen functions in comparison with other s-shaped based algorithm and seven functions compared to the v-shaped based algorithm**s.** Therefore, it appears that the proposed v-shaped family of transfer functions, especially the V4 function, has merit for use in binary algorithms. Additionally, these results indicate how important the role of the transfer function in BPSO is, such that by selecting a suitable function, the performance of BPSO can be increased markedly.

6. Conclusion

In this paper, six new modified transfer function-based BPSO are proposed utilizing different transfer functions. Six new transfer functions divided into two families, s-shaped and v-shaped, are introduced and evaluated. In order to evaluate the performance of all the modified transfer function-based versions, 25 benchmark functions were employed, and the results compared. The highest performance algorithm was VPSO, when compared with six recent high performance modifications of BPSO. Furthermore, the Wilcoxon's rank-sum nonparametric statistical test was carried out at 5% significance level to judge whether the results of the VPSO algorithm differ from the best results of the other algorithms in a statistically significant way. The results show that the new introduced v-shaped family of transfer functions with their own method of updating position vectors can significantly improve the performance of the original BPSO in terms of avoiding local minima and convergence rate. The results also show that the proposed new v-shaped family of transfer functions, especially the V4 function, has merit for use in binary algorithms.

For future studies, it would be interesting to investigate the impact of the new v-shaped family of transfer function on other binary algorithms like Binary Gravitational Search Algorithm and Binary Magnetic Optimization Algorithm. In addition, the use of mathematical functions with variety of slopes and saturations could be explored as new transfer functions for BPSO.

References

[1] R. Eberhart, J. Kennedy, A new optimizer using particles swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995.
[2] R. Eberhart, J. Kennedy, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Network, Perth, Australia, 1995.
[3] S. Chandra, R. Bhat, H. Singh. A pso based method for detection of brain tumors from Mri. In Nature and Biologically Inspired Computing,. NaBIC 2009, Proceeding of the IEEE world Congress on Nature, 666–671, 2009.
[4] M. Mathiyalagan, U. Dhepthie, S. Sivanandam, Grid Scheduling Using Enhanced PSO Algorithm 2 (2) (2010) 140–145.

[5] E. Masehian, D. Sedighizadeh, A multi-objective PSO-based algorithm for robot path planning, in: Proceedings of the IEEE International Conference on Industrial Technology, Vi a del Mar, 2010.
[6] J.J. Liang, P.N. Suganthan, C.C. Chan, V.L. Huang, Wavelength detection in FBG sensor network using tree search DMS-PSO, IEEE Photonics Technology Letters 18 (2006) 1305–1307.
[7] M. Fayk, H. El Nemr, M. Moussa, Particle swarm optimisation based video abstraction, Journal of Advanced Research 1 (2) (2010) 163–167.
[8] Z. Geem, J. Kim, G. Loganathan, A new heuristic optimization algorithm: harmony search, Journal of Simulation 76 (2001) 60–68.
[9] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[10] M. Tayarani-N, M. Akbarzadeh-T, Magnetic optimization algorithms a new synthesis, in: IEEE Congress on Evolutionary Computation, 2008.
[11] E. Rashedi, S. Nezamabadi, S. Saryazdi, GSA: a gravitational search algorithm, Information Sciences 179 (13) (2009) 2232–2248.
[12] Wang Ling, Yin Xu, Yunfei Mao, Minrui Fei, A Discrete Harmony Search Algorithm Life System Modeling and Intelligent Computing, Kang Li, Xin Li, Shiwei Ma George W. Irwin, (Eds.) vol. 98, Springer Berlin Heidelberg, 2010, 37–43.
[13] Wang Ling, Xiping Fu, Muhammad Menhas, Minrui Fei, A Modified Binary Differential Evolution Algorithm Life System Modeling and Intelligent Computing, Kang Li, Minrui Fei, Li Jia George Irwin, (Eds.) vol. 6329, Springer Berlin / Heidelberg, 2010, pp. 49–57.
[14] S. Mirjalili, S.Z. Mohd Hashim, BMOA: binary magnetic optimization algorithm, International Journal of Machine Learning and Computing, 2, 3, 204–208, 2012.
[15] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, BGSA: binary gravitational search algorithm, Natural Computing 9 (3) (2009) 727–745.
[16] J. Kennedy, R. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceedings of the IEEE International Conference on Computational Cybernetics and Simulation, 1997.
[17] G. Luh, C. Lin, A binary particle swarm optimization for continuum structural topology optimization, Applied Soft Computing 11 (2011) 2833–2844.
[18] P. Yin, A discrete particle swarm algorithm for optimal polygonal approximation of digital curves, Journal of Visual Communication and Image Representation 15 (2004) 241–260.
[19] Q. Shen, J.H. Jiang, C.X. Jiao, G. Shen, R.Q. Yu, Modified particle swarm optimization algorithm for variable selection in MLR and PLS modeling: QSAR studies of antagonism of angiotensin II antagonists, European Journal of Pharmaceutical Sciences, Amsterdam, Netherlands 22 (2004) 145–152.
[20] L. Wang, X. Wang, F.J Zhen, L. Zhen, A novel probability binary particle swarm optimization algorithm and its application, Journal of Software 3 (9) (2008) 28–35.
[21] S. Lee, S. Soak, S. Oh, W. Pedrycz, M. Jeon, Modified binary particle swarm optimization, Progress in Natural Science 18 (9) (2008) 1161–1166.
[22] L. Chuang, S. Tsai, C. Yand, Improved binary particle swarm optimization using catfish effect for feature selection, Expert Systems with Applications 38 (2011) 12699–12707.
[23] Y. Xiaohui, Y. Yanbin, W. Cheng, Z. Xiaopan, An improved PSO approach for profit-based unit commitment in electricity market, in: Transmission and Distribution Conference and Exhibition, 2005.
[24] P. Avishek, J. Maitib, Development of a hybrid methodology for dimensionality reduction in Mahalanobis-Taguchi system using Mahalanobis distance and binary particle swarm optimization, Expert Systems with Applications 37 (2) (2010) 1286–1293.
[25] X. Zenga, Y. Li, J. Qina, A dynamic chain-like agent genetic algorithm for global numerical optimization and feature selection, Neurocomputing 72 (4–6) (2009) 1214–1228.
[26] M. Tasgetiren, P.N. Suganthan, Q.K. Pan, A discrete particle swarm optimization algorithm for the generalized traveling salesman problem, in: Proceedings of the 9th Annual Conference on Genetic and evolutionary computation (GECCO '07), New York, NY, USA, 2007.
[27] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Tech. Rep, Nanyang Technological University, vol. 2005005, Singapore, 2005.
[28] Z. Yang, J. Zhang, K. Tang, X. Yao, A. Sanderson, An adaptive coevolutionary differential evolution algorithm for large-scale optimization, in: Evolutionary Computation, 2009, CEC'09, IEEE Congress on 2009.
[29] A. Eiben, S. Smit, Parameter tuning for configuring and analyzing evolutionary algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 19–31.
[30] R.C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in: Congress on Evolutionary Computation 2001, Seoul, Korea, 2001.
[31] J. Derrac, G.S.D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.
[32] García Salvador, Daniel Molina, Manuel Lozano, Francisco Herrera, A Study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case Study On the cec'2005 special session on real parameter optimization, Journal of Heuristics 15 (6) (2009) 617–644.
[33] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (6) (1945) 80–83.
[34] J. Liang, A. Qin, P. Suganthan, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computations 10 (3) (2006) 281–295.
[35] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better,, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 204–210.
[36] J. Liang, P. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in: Swarm Intelligence Symposium IEEE, 2005.
[37] J. Liang, P. Suganthan, Dynamic multi-swarm particle swarm optimizer with local search, in: IEEE Congress on Evolutionary Computation, 2005.