Soft Computing

A novel X-shaped binary particle swarm optimization

--Manuscript Draft--

Manuscript Number:	SOCO-D-19-02761
Full Title:	A novel X-shaped binary particle swarm optimization
Article Type:	Original Research
Keywords:	Binary particle swarm optimization (BPSO); Transfer function; S-shaped, V-shaped and linear transfer functions; X-shaped transfer function
Abstract:	Definitive optimization algorithms are not able to solve high-dimensional optimization problems because the search space grows exponentially with the problem size and an exhaustive search will be impractical. Therefore, approximate algorithms are applied to solve them. A category of approximate algorithms are meta-heuristic algorithms. They have shown an acceptable efficiency to solve these problems. Among them, particle swarm optimization (PSO) is one of the well-known swarm intelligence algorithms to optimize continuous problems. A transfer function is applied in this algorithm to convert the continuous search space to the binary one. The role of the transfer function in binary PSO (BPSO) is very important to enhance the performance of BPSO. Several transfer functions have been proposed for BPSO such as S-shaped, V-shaped and linear transfer functions. However, BPSO algorithm can sometimes find local optima or show slow convergence speed in some problems because of using the velocity of PSO and these transfer functions. In this study, a novel transfer function called X-shaped BPSO (XBPSO) is proposed to increase exploration and exploitation of BPSO in the binary search space. The transfer function uses two functions and improved rules to generate a new binary solution. The proposed method has been run on 33 benchmark instances of the 0-1 multidimensional knapsack problem (MKP) and two discrete maximization functions with low and high dimensions. The results have been compared with some well-known BPSO and discrete meta-heuristic algorithms. The results showed that X-shaped transfer function considerably increased the solution accuracy and convergence speed in BPSO algorithm.

A novel X-shaped binary particle swarm optimization

Zahra Beheshti[1, 2]

1. Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran.
2. Big Data Research Center, Najafabad Branch, Islamic Azad University, Najafabad, Iran.

Email: z-beheshti@iaun.ac.ir

Abstract

Definitive optimization algorithms are not able to solve high-dimensional optimization problems because the search space grows exponentially with the problem size and an exhaustive search will be impractical. Therefore, approximate algorithms are applied to solve them. A category of approximate algorithms are meta-heuristic algorithms. They have shown an acceptable efficiency to solve these problems. Among them, particle swarm optimization (PSO) is one of the well-known swarm intelligence algorithms to optimize continuous problems. A transfer function is applied in this algorithm to convert the continuous search space to the binary one. The role of the transfer function in binary PSO (BPSO) is very important to enhance the performance of BPSO. Several transfer functions have been proposed for BPSO such as S-shaped, V-shaped and linear transfer functions. However, BPSO algorithm can sometimes find local optima or show slow convergence speed in some problems because of using the velocity of PSO and these transfer functions. In this study, a novel transfer function called X-shaped BPSO (XBPSO) is proposed to increase exploration and exploitation of BPSO in the binary search space. The transfer function uses two functions and improved rules to generate a new binary solution. The proposed method has been run on 33 benchmark instances of the 0-1 multidimensional knapsack problem (MKP) and two discrete maximization functions with low and high dimensions. The results have been compared with some well-known BPSO and discrete meta-heuristic algorithms. The results showed that X-shaped transfer function considerably increased the solution accuracy and convergence speed in BPSO algorithm.

Keywords—Binary particle swarm optimization (BPSO); Transfer function; S-shaped, V-shaped and linear transfer functions; X-shaped transfer function

1. Introduction

Kennedy and Eberhart introduced PSO algorithm in 1995 (Kennedy & Eberhart 1995). The algorithm is one of the well-known swarm intelligence algorithms due to a simple structure, high execution speed and a few numbers of parameters. PSO algorithm and its various versions apply to solve many optimization problems such as energy management (Collotta et al. 2017; Kanwar et al. 2017), controller design (Li et al. 2017; Bharti et al. 2017), neural networks training (Chatterjee et al. 2017; Beheshti et al. 2016; Beheshti et al. 2014), molecular docking (García-Nieto et al. 2018) and function optimization (Lin et al. 2018; Beheshti et al. 2014; Beheshti et al. 2013; Beheshti et al. 2016). Nevertheless, PSO has poor exploration and suffers from some disadvantages such as trapping in a local optimum and a premature convergence rate (Beheshti & Shamsuddin 2013). This is due to the fact that the velocity in PSO decreases step by step to be close to zero. In this case, if the best solution found by the swarm, the current position and the best personal position of particle are in the local optimum, the particle traps into the local optimum (Beheshti & Shamsuddin 2013). To overcome these shortcomings, many studies have been conducted such as the improvement of local topologies (Lynn et al. 2018; Marinakis et al. 2017), new strategy to tune parameters (Taherkhani & Safabakhsh 2016; Zhang et al. 2015; Ardizzon et al. 2015), and hybrid PSO with other algorithms and methods (Dong et al. 2018; Barisal et al. 2017; Kamboj 2016; Garg 2016).

Kennedy and Eberhart proposed BPSO algorithm, using a sigmoid function in 1997 (Kennedy & Eberhart 1997). BPSO has been employed to solve many discrete optimization problems such as feature selection (Sheikhan 2017; Kushwaha & Pant 2018; Manohar & Ganesan 2017; Wei et al. 2017), the 0-1 knapsack problem (Wang et al. 2018), network intrusion detection (Malik & Khan 2017) and task allocation (Sun et al. 2018; Yang et al. 2014); however, the BPSO encounters several disadvantages. The velocity of a particle in BPSO is computed as in the PSO; therefore, the algorithm can find a local optimum. Moreover, the sigmoid function creates some problems in BPSO. In PSO, a big value of velocity in the positive and the negative directions show that particles should have a great movement to reach the optimum position. In contrast, a small value indicates that a small movement needs to achieve the optimum solution. In addition, the zero velocity means that the new position should not be changed (Nezamabadi-pour & Maghfoori-Farsangi 2008). Meanwhile, these concepts are changed by using the S-shaped transfer function in BPSO. The value of velocity in the negative and the positive directions creates different values for the new position. Moreover, the zero value of velocity generates different values of zero or one with probability 0.5 for the new position. Additionally, some linear transfer functions (Wang et al. 2008; Bansal & Deep 2012) have been proposed to improve the performance of BPSO, but they are not able to solve the problem of trapping in local optima for many binary optimization problems (Mirjalili & Lewis 2013) and face drawbacks of the sigmoid function.

To overcome these problems, Nezamabadi-pour et al. (Nezamabadi-pour & Maghfoori-Farsangi 2008) proposed a V-shaped transfer function. Although V-shaped transfer functions cover some shortcomings of the S-shaped and the linear transfer functions, BPSO still shows weak results in some problems due to the poor exploration of velocity. A transfer function plays a main role to enhance the performance of BPSO (Mirjalili & Lewis 2013). In early steps, the exploration is very important to search promising regions and avoid trapping in local optima but during the later steps, the exploitation is more essential so that the probability of finding better solutions should be increased. In other words, a balance between exploration and exploitation is essentially to achieve a good result (Islam et al. 2017). For this aim, Islam et al. proposed a time-varying transfer function to balance the exploration and exploitation in BPSO (Islam et al. 2017). Liu and Li also suggested an adaptive inertia weight for S-shaped BPSO (Liu et al. 2016). They showed that a smaller inertia weight improves the exploration ability; while, a larger inertia weight enhances the exploitation capability. In other studies, several techniques applied to improve the efficiency of BPSO (Mafarja & Mirjalili 2017; Lin & Guan 2018a; Lin & Guan 2018b) but they increase the computational complexity of algorithms.

In this study, an X-shaped transfer function (XBPSO) is introduced to improve the performance of BPSO. The transfer function increases exploration and exploitation in BPSO. Two functions are used to generate different results. The best result is chosen and compared with the previous solution. If the new solution is better than the previous one, it will be selected as the next position; otherwise, a crossover operator applies on the new and previous solution. In this case, the best result of crossover operator is chosen as the new position. The proposed transfer function can be employed into all various PSO algorithms to convert the continuous search space to the binary one.

XBPSO has been compared with some various BPSO algorithms, binary bat algorithm (BBA) (Mirjalili et al. 2014), binary artificial bee colony using bitwise operations (Bin-ABC) (Jia et al. 2014), binary gravitational search algorithm (BGSA) (Rashedi et al. 2010) and hybrid whale optimization algorithm with simulated annealing (WOA-SA) (Mafarja & Mirjalili 2017) on 33 benchmark instances of the 0-1 MKP (Beasley 1990) and two maximization binary functions (Rashedi et al. 2010). The results showed that the efficiency of BPSO has been considerably improved by the proposed transfer function compared with the others in terms of solution accuracy and convergence speed.

The rest of the paper is organized as follows: Related works are presented in Section 2. Section 3 deals with the proposed transfer function in details. The results and discussion of the new method compared with several BPSO algorithms and binary swarm intelligence algorithms are provided in Section 4. Finally, conclusion and the future research directions are presented in Section 5.

2. Related works

In this section, the PSO, BPSO and some improved BPSO algorithms are described in details. PSO is a population-based algorithm. Particles in PSO have a velocity, V_i, and a position, X_i, in the D dimension search space as follows:

$$X_i = (x_i^1, x_i^2, ..., x_i^d, ..., x_i^D), \quad for \quad i = 1,2,...,N, \quad d = 1,2,...,D. \tag{1}$$

$$V_i = (v_i^1, v_i^2, ..., v_i^d, ..., v_i^D), \quad for \quad i = 1,2,...,N, \quad d = 1,2,...,D. \tag{2}$$

where N is population size or the number of particles.

Particles' velocities and positions are randomly initialized and updated as follows:

$$v_i^d(t+1) = w(t) * v_i^d(t) + C_1 * rand_1() * \left(pbest_i^d(t) - x_i^d(t)\right) + C_2 * rand_2() * \left(gbest^d(t) - x_i^d(t)\right), \tag{3}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1), \tag{4}$$

where *rand*( ) is a random number in (0,1). w is the inertia weight. C_1 and C_2 are acceleration coefficients. $pbest_i^d$ is the personal best position of the i^{th} particle in the d^{th} dimension. $gbest^d$ is the best position in the d^{th} dimension found so far by swarm.

The best position can be obtained by global or local topologies (Kennedy 1999). In the global topology, the particle moves based on the best solution found by the swarm as shown in (3). In the local topology, each particle has some neighbors. The best neighbor of particle is selected as *lbest*; therefore, the particle moves based on its personal best position and its best neighbor as follows:

$$v_i^d(t+1) = w(t) * v_i^d(t) + C_1 * rand_1() * \left(pbest_i^d(t) - x_i^d(t)\right) + C_2 * rand_2() * \left(lbest_i^d(t) - x_i^d(t)\right). \tag{5}$$

The local topology in the real search space shows a better result than the global topology for solving multimodal problems (Beheshti & Shamsuddin 2015). The particle's velocity in PSO is applied in BPSO. The next position is obtained based on the velocity and the following sigmoid transfer function.

$$Tf\left(v_i^d(t+1)\right) = sigmoid\ (v_i^d(t+1)) = \frac{1}{1+e^{-v_i^d(t+1)}}, \tag{6}$$

$$xb_i^d(t+1) = \begin{cases} 1 & if \quad rand() < Tf\left(v_i^d(t+1)\right) \\ 0 & if \quad rand() \geq Tf\left(v_i^d(t+1)\right) \end{cases}, \tag{7}$$

where $|v_i^d(t+1)| < v_{max}$ and v_{max} is set to a constant value. $xb_i^d(t+1)$ is the next position in the binary search space.

BPSO may converge prematurely and trap into local optima because of its poor exploration (Beheshti et al. 2015). These problems are related to the velocity of PSO and the sigmoid function. Therefore, the improvement of PSO and the transfer function of BPSO enhance the efficiency of BPSO.

A modified binary PSO (MBPSO) was proposed by Shen et al. (Shen et al. 2004) to select variables in MLR and PLS. In this algorithm, particles move based on the following rules:

$$xb_i^d(t+1) = \begin{cases} xb_i^d(t) & if \quad 0 < v_i \leq a \\ pbest_i^d(t) & if \quad a < v_i \leq \frac{1}{2}(1+a) \\ gbest^d(t) & if \quad \frac{1}{2}(1+a) < v_i \leq 1 \end{cases}, \tag{8}$$

where a decreases from 0.5 to 0.33 and the velocity v_i is a random number in range [0, 1].

In MBPSO, if v_i tends to be zero and $xb_i^d(t)$ is in the local optimum $xb_i^d(t+1)$ will remain in the local optimum. It means that the algorithm still tarps in local optima; hence, 10% of particles randomly changes their positions without any rule to avoid trapping in local optima. Fan et al. (FAN et al. 2007) introduced a modified sigmoid function for BPSO to solve the job-shop scheduling problem. They suggested a new method so that the best particle searches in the feasible dimension. In this algorithm, the next position is generated based on the standard BPSO rules and the new sigmoid function is defined as follows:

$$Tf\left(v_i^d(t+1)\right)=\frac{1}{(m-i)+e^{-v_i^d(t+1)}}, \tag{9}$$

where i is the row number of a particle and m is the total row numbers of a particle in the job-shop scheduling problem.

Since this algorithm uses the sigmoid function and BPSO rules, it faces disadvantages of the function as mentioned. Therefore, the algorithm still suffers from trapping in local optima. Lee et al. presented a modified BPSO by the genotype and phenotype concepts (Lee et al. 2008). The algorithm applies a mutation operator to improve the exploration ability. The new real position (genotype position) is obtained based on PSO algorithm, and then the following sigmoid function is used to generate the new binary position (phenotype position):

$$Tf\left(x_{g,i}^d(t+1)\right)=sigmoid(x_{g,i}^d(t+1))=\frac{1}{1+e^{-x_{g,i}^d(t+1)}}, \tag{10}$$

$$x_{p,i}^d(t+1)=\begin{cases}1 & if \quad rand()<Tf\left(x_{g,i}^d(t+1)\right)\\ 0 & if \quad rand()\geq Tf\left(x_{g,i}^d(t+1)\right)\end{cases}, \tag{11}$$

where x_p and x_g are the phenotype (binary) and genotype (real) positions, respectively.

In this algorithm, the sigmoid function and the next position are based on the standard BPSO, therefore; the algorithm may find local optima or exhibit slow convergence speed. Wang et al. (Wang et al. 2008) introduced a new probability BPSO (PBPSO) based on a linear transfer function. Although they changed the transfer function, this modification cannot solve the problem of trapping in local optima for a wide range of binary optimization problems (Mirjalili & Lewis 2013). This transfer function was defined as follows:

$$Tf\left(x_i^d(t+1)\right)=\frac{\left(x_i^d(t+1)-R_{\min}\right)}{\left(R_{\max}-R_{\min}\right)}, \tag{12}$$

$$xb_i^d(t+1)=\begin{cases}1 & if \quad rand()\leq Tf\left(x_i^d(t+1)\right)\\ 0 & if \quad rand()>Tf\left(x_i^d(t+1)\right)\end{cases}, \tag{13}$$

where $L(x)$ is a linear transfer function in (0,1). $[R_{\max}, R_{\min}]$ is a predefined range for obtaining the probability value.

A linear normalized transfer function is introduced by Bansal and Deep (Bansal & Deep 2012) to enhance the exploration ability of BPSO. The transfer function is defined as follows:

$$Tf\left(x_i^d(t+1)\right) = \frac{\left(x_i^d(t+1) + v_i^d(t+1) + v_{\max}\right)}{1 + 2v_{\max}}, \tag{14}$$

The next position is generated based on the following rule:

$$xb_i^d(t+1) = \begin{cases} 1 & if \quad rand() < Tf\left(x_i^d(t+1)\right) \\ 0 & if \quad rand() \geq Tf\left(x_i^d(t+1)\right) \end{cases}. \tag{15}$$

As mentioned before, S-shaped and linear transfer functions encounter some disadvantages. For solving these shortcomings, several V-shaped families transfer functions have been proposed so far (Nezamabadi-pour & Maghfoori-Farsangi 2008; Mirjalili & Lewis 2013; Beheshti et al. 2015; Beheshti 2018). A new V-shaped transfer function was introduced by Nezamabadi-pour et al. for BPSO (Nezamabadi-pour & Maghfoori-Farsangi 2008). In this algorithm, the next position was generated using a new rule. If a random number is less than the value of $Tf\left(v_i^d(t+1)\right)$, the next position is generated by changing bits of the current position from zero to one or vice versa; otherwise, the next position will be the current position as follows:

$$Tf\left(v_i^d(t+1)\right) = \left|\tanh\left(\alpha . v_i^d(t+1)\right)\right|, \tag{16}$$

$$xb_i^d(t+1) = \begin{cases} Complement\left(xb_i^d(t)\right) & if \quad rand() < Tf\left(v_i^d(t+1)\right) \\ xb_i^d(t) & if \quad rand() \geq Tf\left(v_i^d(t+1)\right) \end{cases}, \tag{17}$$

where α is a constant value.

The results of the proposed method showed that the algorithm may get stuck into local optima because of the poor exploration in PSO. Nezamabadi-pour et al. (Nezamabadi-pour & Maghfoori-Farsangi 2008) introduced an improved V-shaped transfer function to avoid the premature convergence and the stagnation of algorithm as follows:

$$Tf\left(v_i^d(t+1)\right) = A + (1-A) * \left|\tanh\left(\alpha v_i^d(t+1)\right)\right|, \tag{18}$$

A is a parameter and is computed as follows:

$$A = k\left(1 - e^{-\frac{F}{T}}\right), \tag{19}$$

where k and T are set to constant values and F is a failure counter.

A failure occurs when the best solution found by the swarm is not improved per iteration; in this case, F is increased by one. If the failure does not happen, the F will be zero. The failure counter can prevent getting stuck in local optima in some cases.

Mirjalili and Lewis (Nezamabadi-pour & Maghfoori-Farsangi 2008) evaluated the performance of S-Shaped and V-shaped transfer functions using the CEC 2005 benchmark functions. Moreover, they introduced some S-shaped and V-shaped families of transfer functions to convert the continuous search space to the binary one. The results indicated that the following V-shaped transfer function performs better than the others for benchmark functions.

$$Tf\left(v_i^d(t+1)\right) = \left|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}v_i^d(t+1)\right)\right|. \tag{20}$$

Beheshti et al. (Beheshti et al. 2015) proposed a memetic binary hybrid topology PSO (BHTPSO) and a quadratic interpolation BHTPSO (BHTPSO-QI) to improve the exploration and exploitation abilities of BPSO. These algorithms apply both global and local topologies to obtain the next position in the binary search space: The next velocity is computed based on the global and the local topologies and the next position is created as follows:

$$a_i^d(t+1) = v_i^d(t+1) + C_3(t) \times rand() \times (gbest^d(t) - xb_i^d(t)), \tag{21}$$

$$Tf\left(a_i^d(t+1)\right) = E + (1-E) \times \left|tanh\left(a_i^d(t+1)\right)\right|, \tag{22}$$

$$\begin{aligned} &if \quad rand() < Tf\left(a_i^d(t+1)\right) \quad then \quad xb_i^d(t+1) = complement\ (xb_i^d(t)) \\ &\qquad else \quad xb_i^d(t+1) = xb_i^d(t), \quad for \quad i = 1,2,\ldots,N. \end{aligned}, \tag{23}$$

where C_3 in (21) is computed as follows:

$$C_3(t) = C_{3,min} + \frac{\left(C_{3,max} - C_{3,min}\right)}{T} \times t \ . \tag{24}$$

where C_{min} and C_{max} are lower and upper bound of C_3. Parameters t and T are the current iteration and the maximum iteration, respectively.

E in (22) is calculated as follows:

$$E = erf\left(\frac{NF}{T}\right) = \frac{2}{\sqrt{\pi}} \int_0^{\frac{NF}{T}} e^{-t^2} dt \ , \tag{25}$$

where *erf* is an error function. *NF* is set to zero and it will be increased when the best solution is not improved per iteration.

In BHTPSO-QI, three different particles create a new particle ($\tilde{X}$). If the fitness function of $\tilde{X}$ is better than that of *gbest*, $\tilde{X}$ is known as the best particle. $\tilde{X}$ is generated as follows:

$$\tilde{X} = (xb_j^d \quad xor \quad xb_k^d) \quad or \quad (xb_k^d \quad xor \quad gbest^d) \quad or \quad (gbest^d \quad xor \quad xb_j^d), \quad j \neq k \neq gbest \ . \tag{26}$$

Although V-shaped transfer functions have been shown better performances than S-shaped and linear transfer functions in many problems, BPSO algorithms with the V-shaped family of transfer functions may get stuck in local optima. If the current position funded by a particle is the local optimum and the velocity is zero, the next position will be the current position. Therefore, if this particle is the best particle (*gbest*), all particles will be converged to this position and trap in the local optimum. The linear, S-shaped and V-shaped transfer functions have been shown in Fig. 1.

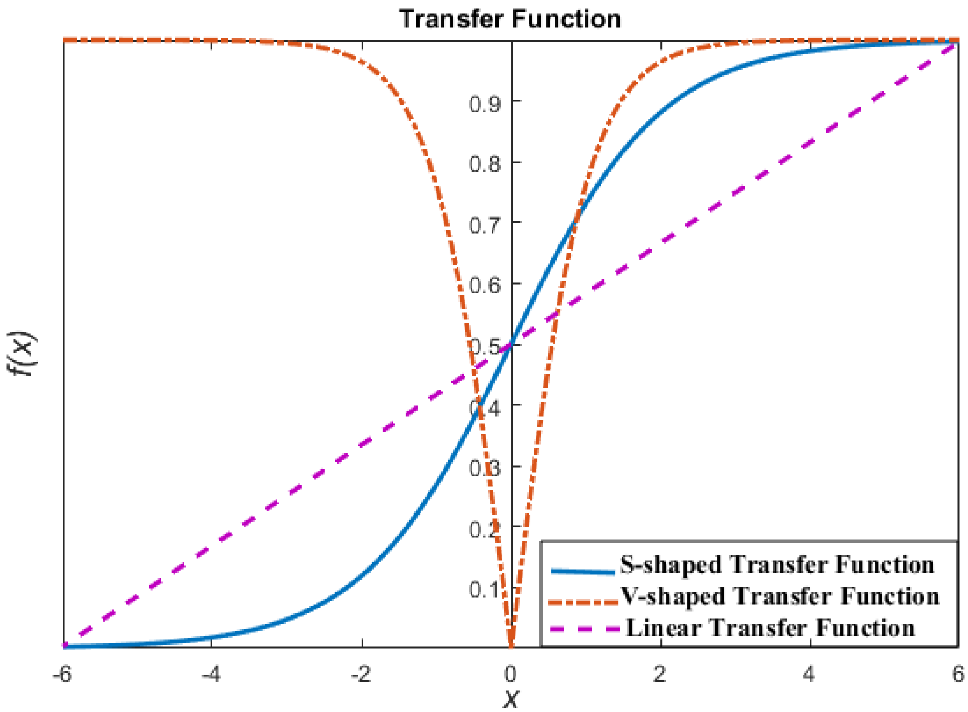


Fig. 1. The S-shaped, V-shaped and linear transfer functions

Kiran (Kiran 2015) proposed the following relation to convert a continuous value to the binary value in the artificial bee colony (ABC):

$$xb_i^d(t+1) = round\left(\left|x_i^d(t+1) \mod 2\right|\right) \mod 2. \tag{27}$$

Yuan et al. proposed an improved BPSO combined with a lambda-iteration method to solve the unit commitment (UC) problem (Yuan et al. 2009). The method applies the logical operators in PSO. The velocity and the position are binary strings and the following operators are applied to generate the next position:

$$vb_i^d(t+1) = \omega_1 \otimes \left(pbest_i^d(t) \oplus xb_i^d(t)\right) + \omega_2 \otimes \left(gbest^d(t) \oplus xb_i^d(t)\right), \tag{28}$$

$$xb_i^d(t+1) = xb_i^d(t) \oplus vb_i^d(t+1), \tag{29}$$

where $\otimes$ is "AND" operator, $\oplus$ is "XOR" operator, + is "OR" operator. ω_1 and ω_2 are random binary integer numbers in [0, 1].

In this algorithm, if the velocity is zero and the current position is the local optimum, the next position will remain in the local optimum position because of using the XOR operator in (29). A hybrid BPSO was presented by Lin and Guan to solve the obnoxious *p*-median problem as an NP-hard problem (Lin & Guan 2018a). The method obtains the new position based on the following rules.

$$xb_i^d(t+1)=\begin{cases} xb_i^d(t)\oplus\left(pbest_i^d(t)\sim xb_i^d(t)\right) & if \quad 0\le rand()<prob_p \\ xb_i^d(t)\oplus\left(gbest^d(t)\sim xb_i^d(t)\right) & if \quad prob_p\le rand()<prob_p+prob_g \\ xb_i^d(t)\oplus\left(pbest_j^d(t)\sim xb_i^d(t)\right) & if \quad prob_p+prob_g\le rand()<1 \end{cases}, \tag{30}$$

where the notations of $\oplus$ and ~ are defined as sum and difference operators, respectively. $prob_p$ and $prob_g$ are set to constant values less than one. $pbest_j^d(t)$ is the personal best position of particle $j\neq i$ and the particle *j* is randomly chosen.

In this proposed algorithm, if all if the *gbest* found by the *ith* particle, the current position *i*, $pbest_i$ and *gbest* are in a local optimum position, the second terms in (30) will be zero. Therefore, the next position will be the current position (the local optimum).

Islam et al. (Islam et al. 2017) introduced a time-varying transfer function for BPSO. The following parameter has been applied to balance the exploration and exploitation abilities of BPSO:

$$Tf\left(v_i^d(t+1),\varphi\right)=\frac{1}{1+e^{-v_i^d(t+1)/\varphi}}, \tag{31}$$

$$\varphi=\varphi_{max}-iter\left(\frac{\varphi_{max}-\varphi_{min}}{max\_iteration}\right), \tag{32}$$

where φ_{max} and φ_{min} control the bound of φ. *iter* and *max_iteration* are the current iteration and maximum iteration, respectively.

Although this transfer function improves the exploration and the exploitation of BPSO, it faces disadvantages of the S-shaped transfer functions. In another study, Liu et al. (Liu et al. 2016) analyzed the value of *w* in BPSO to have better performance. In PSO, *w* is decreased step by step to balance the exploration and the exploitation abilities. In the early stages of the run, *w* has a bigger value to search new spaces and in the later stages of the run, it has a smaller value to switch from the exploration to the exploitation. They indicated that if C_1 and C_2 are set to a constant value; a smaller value of *w* improves the exploration ability in BPSO; whereas, a larger value of *w* encourages the exploitation. They proposed the following *w* for changing the inertia weight in BPSO:

$$w = \begin{cases} \underline{w} + \dfrac{\pi.\left(\overline{w} + \underline{w}\right)}{\rho.\overline{\pi}} & if \quad \pi \le \rho.\overline{\pi} \\ \overline{w} & if \quad \rho.\overline{\pi} < \pi \le .\overline{\pi} \end{cases} \tag{33}$$

where $\overline{w}$ and $\underline{w}$ are the upper and lower bound of w, respectively. ρ is the fraction of iteration for updating w. π and $\overline{\pi}$ stand the current iteration and the maximum iteration.

Although the above algorithms have improved the performance of BPSO in some tested problems, a new transfer function still requires covering the shortcomings of transfer functions. The new method should improve the exploration ability of BPSO and have the advantages of transfer functions. In the next section, a new transfer function is introduced for this aim.

3. X-shaped BPSO – The proposed method

In this section, X-shaped BPSO (XBPSO) is introduced for the binary search space. In PSO, an absolute big value of velocity shows that a great movement needed to achieve the best solution. The small velocity indicates that the particle is close to the best result. The zero value of velocity illustrates that the current position is not changed ($x_i^d(t+1) = x_i^d(t)$). In BPSO, the V-shaped transfer functions and their rules have been designed based on these concepts in PSO. When the velocity is zero, the next position will be equal to the current position. If the current position is the local optimum, the next position will remain in the position. Since the velocity of PSO has a poor exploration, the BPSO with the transfer function also traps in local optima. In other transfer functions, the S-shaped transfer functions, the next position is modified without considering the current position. In other words, a big value of velocity in the positive direction increases the probability of the next position to be one value. It is vice versa for the negative direction. The next position will be zero, if the velocity shows a big value in the negative direction (Nezamabadi-pour & Maghfoori-Farsangi 2008; Islam et al. 2017). When the velocity is zero, the next position will be 1 or 0 with the probability of 0.5. The velocity must be zero when the optimum solution is found; however, the next position will be changed using this transfer function. This problem is led to the divergence of swarm in the last stages of the run (Nezamabadi-pour & Maghfoori-Farsangi 2008). Linear transfer functions also face the drawbacks of S-shaped transfer functions because of using the same velocity and rules.

In this section, the advantages and disadvantages of proposed transfer functions are considered to introduce a new transfer function. Two functions are applied for this purpose to improve exploration and exploitation of BPSO as shown in Fig. 2.

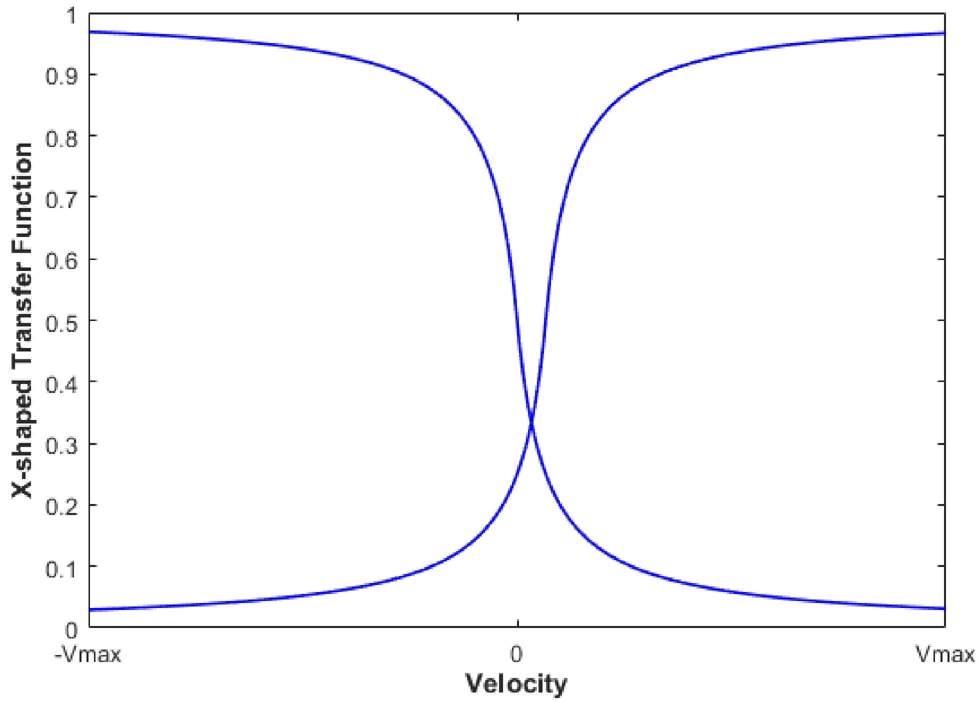


Fig. 2. The X-shaped transfer function

In X-shaped BPSO (XBPSO), the new position is generated as follows:

$$S_1\left(v_i^d(t+1)\right)=\frac{-v_i^d(t+1)}{1+\left|-v_i^d(t+1)\right|*0.5}+0.5 \tag{34}$$

$$y_i^d=\begin{cases}1 & if \quad rand_1()>S_1\left(v_i^d(t+1)\right)\\ 0 & if \quad rand_1()\le S_1\left(v_i^d(t+1)\right)\end{cases} \tag{35}$$

$$S_2\left(v_i^d(t+1)\right)=\frac{v_i^d(t+1)-1}{1+\left|v_i^d(t+1)-1\right|*0.5}+0.5 \tag{36}$$

$$z_i^d=\begin{cases}1 & if \quad rand_2()<S_2\left(v_i^d(t+1)\right)\\ 0 & if \quad rand_2()\ge S_2\left(v_i^d(t+1)\right)\end{cases}, \tag{37}$$

Two new positions y_i and z_i are created by two functions. The best position is selected as follows:

$$P_i(t+1)=\begin{cases}y_i & if \quad f(y_i) \quad is \quad better \quad than \quad f(z_i)\\ z_i & if \quad f(z_i) \quad is \quad better \quad than \quad f(y_i)\end{cases}, \tag{38}$$

where *f(.)* is the fitness function.

If $P_i(t+1)$ is better than $xb_i(t)$, then the next position will be $P_i(t+1)$; otherwise, $P_i(t+1)$ and $xb_i(t)$ are chosen as two parents and a crossover operator is run on them. The result of crossover is two children (*Child1*, *Child2*) and the best one is selected as the next position. The crossover operator can be a single, double or uniform point crossover chosen based on a roulette weal function.

$$
\begin{array}{l}
if \quad f\left(P_i(t+1)\right) \quad is \quad better \quad than \quad f\left(xb_i(t)\right) \\
\qquad\qquad xb_i(t+1) = P_i(t+1) \\
else \\
\qquad [Child1, \quad Child2] = crossover\left(P_i(t+1) \quad , \quad xb_i(t)\right) \cdot \\
\qquad xb_i(t+1) = best\left(Child1, \quad Child2\right) \\
endif
\end{array}
\tag{39}
$$

For example, if v_i is [-0.8 -3.0 1.0 6.0 0 -5.0 4.5 2.4 -3.1], then $S_1(v_i)$ and $S_2(v_i)$ will be as follows:

$S_1(v_i) = [0.7222 \quad 0.8750 \quad 0.2500 \quad 0.0714 \quad 0.5000 \quad 0.9167 \quad 0.0909 \quad 0.1471 \quad 0.8780]$,

$S_2(v_i) = [0.1786 \quad 0.1000 \quad 0.5000 \quad 0.9167 \quad 0.2500 \quad 0.0714 \quad 0.8889 \quad 0.7917 \quad 0.0980]$.

$rand_1$ and $rand_2$ are in (0,1); hence, y_i and z_i with different values of $rand_1$ and $rand_2$ can be generated as follows:

$y_i = [0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1]$,

$z_i = [1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0]$.

The pseudo code of XBPSO has been shown in Fig. 3. The crossover operator improves the exploration ability. Moreover, some parts of the good solution are inherited by the children so that the ability of exploitation is enhanced. Algorithms are usually compared with each other based on the time complexity. It describes the amount of time that the algorithm takes to be run. The time complexity is commonly expressed using big *O* notation. The proposed method has *O(T*N*D)*, where *T* is maximum iteration, *N* is the population size and *D* is the dimension. The time complexity of BPSO is also *O(T*N*D)*; therefore, the time complexity in XBPSO is not increased compared to BPSO algorithm, and this is one of the advantages of the proposed algorithm.

Algorithm: The pseudo code of XBPSO

1. Initialize particles' velocities with zero, particles' positions randomly in the binary search spaces and $y_i^d = [\ ], i = 1,2,\ldots,N.\ d = 1,2,\ldots,D.\ \ and\ \ z_i^d = [\ ], i = 1,2,\ldots,N.\ d = 1,2,\ldots,D$
2. *pbest=x*
3. Evaluate all positions, $x_i^d, i = 1,2,\ldots,N.\ d = 1,2,\ldots,D.$
4. Compute the best solution (*gbest*) found so far by swarm
5. **Repeat**
6.   ***For i=1 to N Do***
7.   ***For d=1 to D Do***
8.   Compute v_i^d (for global topology using (3) and for local topology using (5))
9.   Calculate $S_1\left(v_i^d(t+1)\right) = \dfrac{-v_i^d(t+1)}{1+\left|-v_i^d(t+1)\right| * 0.5} + 0.5$
10.   ***If*** $rand_1() > S_1\left(v_i^d(t+1)\right)$ ***then*** $y_i^d = 1$
11.   ***else*** $y_i^d = 0$
12.   ***End If***
13.   Calculate $S_2\left(v_i^d(t+1)\right) = \dfrac{v_i^d(t+1)-1}{1+\left|v_i^d(t+1)-1\right| * 0.5} + 0.5$
14.   ***If*** $rand_2() < S_2\left(v_i^d(t+1)\right)$ ***then*** $z_i^d = 1$
15.   ***else*** $z_i^d = 0$
16.   ***End If***
17.   ***End For d***
18.   **Calculate the next position** $xb_i(t+1)$**:**
19.   ***If*** $f(y_i)$ **is better than** $f(z_i)$ ***then*** $P_i(t+1) = y_i$
20.   ***else*** $P_i(t+1) = z_i$
21.   ***End If***
22.   ***If*** $f(P_i(t+1))$ **is better than** $f(xb_i(t))$ ***then*** $xb_i(t+1) = P_i(t+1)$
23.   ***else***
24. $[Child_1\ ,\ Child_2] = crossover\left(P_i(t+1), xb_i(t)\right)$
25. $xb_i(t+1) = best\left(Child_1\ ,\ Child_2\right)$
26.   ***End If***
27.   ***End For i***
28.   Calculate the best solution (*gbest*).
29. **UNTIL** *certain stopping criteria* ***is met***
30. ***Return gbest***

Fig. 3. The pseudo code of XBPSO

To illustrate how to obtain XBPSO a solution in the binary search space, *Max-ones* function is selected to trace the algorithm. *Max-ones* function is a maximization binary function (Rashedi et al. 2010). In this function, the best result is equal to its dimension (D). It is defined as follows:

$$Max-ones=\sum_{i=1}^{D} x_i \tag{40}$$

To trace the algorithm, the population size (N), and the dimension (D) are set to 3 and 6, respectively. In the initialization step, all particles' velocities (v_i^d, $i=1,2,.3$, $d=1,2,3,4,5,6$) are set to zero and particles' positions (x_i^d, $i=1,2,.3$, $d=1,2,3,4,5,6$) are randomly initialized in the binary search spaces. Then, their fitness values are computed as follows:

Initialization Step:

$x_1=[0\quad 0\quad 1\quad 0\quad 0\quad 0]$, $\quad f(x_1)=1$

$x_2=[1\quad 0\quad 0\quad 0\quad 0\quad 1]$, $\quad f(x_2)=2$

$x_3=[0\quad 1\quad 0\quad 0\quad 0\quad 0]$, $\quad f(x_3)=1$

The 2th particle has the best fitness value in this step (*f*(*gbest*)=*2*). At Iteration #1, the values of y_i^d and z_i^d as well as their fitness values are obtained as follows:

Iteration #1:

$y_1=[1\quad 0\quad 0\quad 0\quad 0\quad 0]$, $\quad f(y_1)=1$

$y_2=[1\quad 0\quad 1\quad 0\quad 0\quad 1]$, $\quad f(y_2)=3$

$y_3=[1\quad 0\quad 0\quad 0\quad 1\quad 0]$, $\quad f(y_3)=2$

$z_1=[1\quad 1\quad 0\quad 0\quad 0\quad 1]$, $\quad f(z_1)=3$

$z_2=[1\quad 1\quad 1\quad 1\quad 0\quad 1]$, $\quad f(z_2)=5$

$z_3=[1\quad 0\quad 1\quad 0\quad 0\quad 1]$, $\quad f(z_3)=3$

During the first iteration, p_i, $i=1,2,3.$ and $f(p_i)$ are computed as follows:

$p_1=[1\quad 1\quad 0\quad 0\quad 0\quad 1]$, $\quad f(p_1)=3$

$p_2=[1\quad 1\quad 1\quad 1\quad 0\quad 1]$, $\quad f(p_2)=5$

$p_3=[1\quad 0\quad 1\quad 0\quad 0\quad 1]$, $\quad f(p_3)=3$

Since $f\left(P(t+1)\right)$ is better than $f\left(xb(t)\right)$ for all cases, the next positions will be as follows:

$x_1 = [1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1], \quad f(x_1) = 3$

$x_2 = [1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1], \quad f(x_2) = 5$

$x_3 = [1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1], \quad f(x_3) = 3$

Therefore, the 2th particle achieves the best fitness value in this step (*f*(*gbest)=5*). Then, y_i and z_i are updated at iteration #2 as follows:

Iteration #2:

$y_1 = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0], \quad f(y_1) = 5$

$y_2 = [1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0], \quad f(y_2) = 3$

$y_3 = [1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1], \quad f(y_3) = 5$

$z_1 = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1], \quad f(z_1) = 4$

$z_2 = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1], \quad f(z_2) = 3$

$z_3 = [0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1], \quad f(z_3) = 3$

p_i also is modified as follows:

$p_1 = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0], \quad f(p_1) = 5$

$p_2 = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1], \quad f(p_2) = 3$

$p_3 = [1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1], \quad f(p_3) = 5$

Therefore, the next position of particles 1 and 3 are modified as follows:

$x_1 = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0], \quad f(x_1) = 5$

$x_3 = [1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1], \quad f(x_3) = 5$

Since the next position of particle 2 is not better than its current position, the crossover operator is done on p_2 and x_2. Then, the next position will be changed as follows:

$x_2 = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1], \quad f(x_2) = 6$

Therefore, the 2th particle archives the global best solution (*f*(*gbest)=6*).

4. Results and Discussion

The proposed method has been implemented and compared with various well-known BPSO algorithms, and binary swarm intelligence algorithms such as BGSA (Rashedi et al. 2010), BBA (Mirjalili et al. 2014), Bin-ABC (Jia et al. 2014) and WOA-SA (Mafarja & Mirjalili 2017). The performances of algorithms are evaluated by the 0-1 MKP benchmarks (Beasley 1990) and two discrete benchmark functions (Rashedi et al. 2010). BGSA and BBA apply a V-shaped transfer function. Bin-ABC uses the bitwise operation in ABC algorithm to create a binary ABC algorithm.

Mirjalili and Lewis showed that the V-shaped transfer function defined in (20) (Mirjalili & Lewis 2013) has a better performance than the other proposed S-shaped and V-shaped transfer functions in solving CEC 2005 benchmark functions. Hence, the transfer function has been selected and used in BPSO. The BPSO is referred to as VBPSO in this study. Moreover, the transfer function defined in (27) has been applied in BPSO (PSO-bin) to compare the ability of transfer functions. The standard BPSO and two other improved BPSO algorithms, BHTPSO-QI (Beheshti et al. 2015) and TV-BPSO (Islam et al. 2017), are employed in the comparison. In addition, the local topology of BPSO, VBPSO and XBPSO are implemented to evaluate the performance of transfer functions.

4.1. Benchmark instances

4.1.1. Benchmark instances for 0-1 MKP

As shown in Table 1, thirty-three benchmark instances of the 0-1 MKP (Beasley 1990) have been selected to test the efficiency of algorithms. In this table, n and m are the number of objects and knapsacks, respectively. The best-known profit for each benchmark has been shown in the table.

The 0-1 MKP consists of m knapsacks and n objects. Each object i has a weight w_i and a profit p_i where $i=1,2,\ldots,n$. The aim in the problem is to select a subset of objects in such a way that the total profits are maximized. Moreover, the total weight of these selected items does not exceed knapsack capacities. The problem is defined as follows:

$$\begin{aligned}
&Maximize && \sum_{i=1}^{n} p_i x_i \\
&Subject\ \ to && \sum_{i=1}^{n} w_{ij} x_i \le C_j , \\
& && x_i \in \{0,1\}, \quad 1 \le i \le n, \quad 1 \le j \le m
\end{aligned} \tag{41}$$

where C_j is the capacity of the j^{th} knapsack.

Some of them are infeasible solutions; when, algorithms generate solutions for the 0-1 MKP. Therefore, a penalty function requires decreasing the probability of selecting these infeasible solutions. The following penalty function (Beheshti et al. 2013) is applied to avoid choosing infeasible solutions:

$$Penalty = \frac{\sum_{i=1}^{n} p_i x_i}{\eta + \underset{j=1..m}{Max}\left(\sum_{i=1}^{n} w_{ji} x_i - C_j\right)}, \tag{42}$$

where η is a positive value. It is set to 100 (Beheshti et al. 2013) in the experimental results.

Table 1. The 0-1 MKP benchmarks (Beasley 1990)

Benchmark No.	Benchmark Name	Best Known	n	m
1.	mknapcb1-5.100-00	24381	100	5
2.	mknapcb1-5.100-01	24274	100	5
3.	mknapcb1-5.100-02	23551	100	5
4.	mknapcb1-5.100-03	23534	100	5
5.	mknapcb1-5.100-04	122319	100	5
6.	mknapcb2-5.250-00	59312	250	5
7.	mknapcb2-5.250-01	61472	250	5
8.	mknapcb2-5.250-02	62130	250	5
9.	mknapcb2-5.250-03	59446	250	5
10.	mknapcb2-5.250-04	58951	250	5
11.	mknapcb3-5.500-00	120130	500	5
12.	mknapcb3-5.500-01	117837	500	5
13.	mknapcb3-5.500-02	121109	500	5
14.	mknapcb3-5.500-03	120798	500	5
15.	mknapcb3-5.500-04	122319	500	5
16.	mknapcb4-10.100-00	23064	100	10
17.	mknapcb4-10.100-01	22801	100	10
18.	mknapcb4-10.100-02	22131	100	10
19.	mknapcb4-10.100-03	22772	100	10
20.	mknapcb4-10.100-04	22751	100	10
21.	mknapcb5-10.250-00	59187	250	10
22.	mknapcb5-10.250-01	58662	250	10
23.	mknapcb5-10.250-02	58094	250	10
24.	mknapcb5-10.250-03	61000	250	10
25.	mknapcb5-10.250-04	58092	250	10
26.	mknapcb6-10.500-00	117726	500	10
27.	mknapcb6-10.500-01	119139	500	10
28.	mknapcb6-10.500-02	119159	500	10
29.	mknapcb6-10.500-03	118802	500	10
30.	mknapcb6-10.500-04	116434	500	10
31.	mknapcb8-30.250-29	150038	250	30
32.	mknapcb9-30.500-28	303605	500	30
33.	mknapcb9-30.500-29	301021	500	30

4.1.2. Maximization benchmark functions

Max-ones and *Royal-road* functions are two binary functions. They are binary in nature and should be maximized (Rashedi et al. 2010). In *Max-ones*, the best result is equal to its dimension (*D*). The best result in *Royal-road* function is equal to *D/8*. The low and high dimensions of *Max-ones* and *Royal-road* functions are considered to test the efficiency of algorithms. *Royal-road* function is defined as follows:

$$Royal-road = \sum_{i=1}^{D/8}\left(\prod_{j=8(i-1)+1}^{8i} x_j\right) \tag{43}$$

4.2. Parameter settings

As mentioned before, the parameter *w* has a key role in the performance of BPSO. The value of *w* in the standard BPSO, VBPSO, XBPSO and PSO-bin is set to different values (Islam et al. 2017; Liu et al. 2016; Shi & Eberhart 1998) according to Table 2. Firstly, *w* is linearly decreased from 0.9 to 0.4 (Shi & Eberhart 1998) per iteration as follows:

$$w = w_{max} - iter\left(\frac{w_{max} - w_{min}}{max\_iteration}\right), \tag{44}$$

where w_{max} and w_{min} control the bound of w. *iter* and *max_iteration* are the current iteration and maximum iteration, respectively.

Table 2. Parameter settings of algorithms

Algorithm	Parameter
XBPSO, BPSO, VBPSO, PSO-bin, LVBPSO (Shi & Eberhart 1998)	$w_{max}=0.9$, $w_{min}=0.4$, $C1=2$, $C2=2$
XBPSO1, BPSO1, VBPSO1, PSO-bin1, LBPSO1 (Liu et al. 2016)	$\rho=1$, $\underline{w}=0.4$, $\overline{w}=1$, $\pi = Current \quad iteration$ $\overline{\pi} = Maximum\ iteration$, $C1=2$, $C2=2$
XBPSO2, BPSO2, VBPSO2, PSO-bin2, LXBPSO2 (Islam et al. 2017)	$w=1$, $C1=2$, $C2=2$
BHTPSO-QI (Beheshti et al. 2015)	$w_{max}=0.6$, $w_{min}=0.2$, $C_{1,max}=2$, $C_{1,min}=0.5$, $C_{2,max}=2$, $C_{2,min}=1$, $C_{3,max}=1.5$, $C_{3,min}=0.5$
TV-BPSO (Islam et al. 2017)	$\varphi_{max}=5$, $\varphi_{min}=1$, $w=1$, $C1=2$, $C2=2$
BBA (Mirjalili et al. 2014)	$F_{max}=2$, $F_{min}=0$, $A=0.25$, $r=0.5$, $\varepsilon=[-1,\ 1]$, $\gamma=0.9$, $a=0.9$
BGSA (Rashedi et al. 2010)	$G_0=100$, $k_{0\max}=N$ $k_{0\min}=1$
WOA-SA (Mafarja & Mirjalili 2017)	*Maximum number of Sub-iteration=10, Temperature Reduction Rate=0.99*
Bin-ABC (Jia et al. 2014)	*r=0.5, Limit=100*

Table 3. The maximum and average best profit for the 0-1 MKP benchmarks with different *w*

Benchmark / Algorithm	mknapcb1-5.100-00		mknapcb1-5.100-01		mknapcb1-5.100-02		mknapcb1-5.100-03		mknapcb1-5.100-04	
Profit	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average
XBPSO	21227	20245.1	20767	19965.9	20168	19142.5	20649	19919	20894	19926.7
XBPSO1	23334	22572.3	22918	22227.7	22640	21695.5	22286	21709	23084	22243
XBPSO2	**24184**	**23714.8**	**24082**	**23464.2**	**23457**	**22994.7**	**23245**	**22788.6**	**23617**	**23293.8**
BPSO	21125	20247.8	20903	20096.9	20400	19319.2	20899	19986	20739	20058.3
BPSO1	**23531**	**23064.5**	**23482**	**22867.8**	**22994**	**22242.6**	**23009**	**22206.9**	**23420**	**22788.6**
BPSO2	23334	22577.9	23279	22326.7	22816	21912.9	22623	21846.1	23161	22345.5
VBPSO	**23610**	**22789.7**	**22977**	**22351**	**22498**	**21905.9**	**22930**	**22166.1**	**23262**	**22537.3**
VBPSO1	22812	21601.1	22703	21336.7	22158	20540.8	21959	21147.8	22093	20885.4
VBPSO2	18646	4138.27	19379	3687.21	18566	5280.92	19521	3719.06	19816	7877.1
PSO-bin	**23302**	**22315.5**	**22768**	**22005.3**	**22137**	**21212.9**	**22296**	**21571.6**	**22722**	**21866.1**
PSO-bin1	23109	21338.7	22395	21206.6	21578	20400.4	21973	21006.9	22641	20949
PSO-bin2	22203	21208.9	22255	20992.3	21618	20388.8	21888	20960.6	22206	21160.6

Secondly, *w* is linearly increased from 0.4 to 1 based on (33) (Liu et al. 2016). These algorithms with the parameter settings are referred to as XBPSO1, BPSO1, PSO-bin1 and VBPSO1. Thirdly, *w* is set to 1 (constant value) (Islam et al. 2017). These algorithms are referred to as XBPSO2, BPSO2, PSO-bin2, and VBPSO2. In algorithms, *C1* and *C2* are set to 2 (Eberhart & Shi 2001; Mirjalili & Lewis 2013). To find the optimal value for *w* in these algorithms, the results on five MKP benchmarks and two maximization functions have been presented in Tables 3 and 4.

Table 4. The average and standard deviation (±SD) of the best solution for maximization benchmark functions with different values of *w*

Function / Algorithm	Max-ones Dimension=100	Max-ones Dimension=200	Royal Road Dimension=80	Royal Road Dimension=120
XBPSO	83.3±0.823	151.3±2.409	4.033±0.556	5.2±0.633
XBPSO1	**100±0**	186.4±1.265	6.933±1.015	9.2±1.229
XBPSO2	**100±0**	**200±0**	**7±1.174**	**11.5±0.85**
BPSO	87.6±1.793	155.17±1.984	3.833±0.461	7.633±1.377
BPSO1	**100±0**	**200±0**	**6.833±0.913**	8.4±1.3499
BPSO2	**100±0**	**200±0**	4.833±1.262	**8.467±1.358**
VBPSO	**99.667±0.607**	**195.07±2.180**	**5.733±1.015**	7±0.8165
VBPSO1	95.7±1.705	179.37±4.081	4.033±0.964	**7.233±1.194**
VBPSO2	76.933±1.999	137.5±2.432	2.833±0.648	4.667±0.844
PSO-bin	**99.033±1.066**	**190.07±2.716**	**4.933±0.980**	3.467±0.629
PSO-bin1	93.333±2.139	172.67±4.544	4.633±0.850	**6.167±1.177**
PSO-bin2	96.3±1.803	180.2±4.923	3.367±0.669	5.833±1.234

The BPSO algorithms are run, and their results are averaged over 30 independent runs. The maximum iteration and the population size are set to 1000 and 40 (Islam et al. 2017; Wang et al. 2008), respectively. In tables, the best results for each algorithm have been shown in bold. As seen, XBPSO2, BPSO1, VBPSO and PSO-bin have better results as compared with the other algorithms. The parameter settings of other algorithms in Table 2 are based on their references.

4.3. Results on 0-1 MKP benchmarks

Different sets of the 0-1 MKP benchmarks have been selected to evaluate the performance of algorithms. These benchmarks have been extensively applied in the literature (Beheshti et al. 2015; Beheshti et al. 2013; Zouache et al. 2016; Abdel-Basset et al. 2018). XBPSO2, BPSO1, VBPSO, and PSO-bin, which provide better solutions in Table 3 and Table 4, are selected and compared their performance with BHTPSO-QI, TV-BPSO, BGSA, WOA-SA, BBA and Bin-ABC algorithms.

The best, average, worst and standard deviation (STD) of the best solution in the last iteration are reported in Table 5. It can be observed that XBPSO2 significantly outperforms other algorithms on the set of mknapcb1 benchmarks. In other words, XBPSO2 achieves the maximum profit among the other algorithms. These benchmarks are easier than the others to solve, and the proposed method can obtain the best results with the smallest STD as compared with other algorithms. In contrast, BBA and Bin-ABC obtain the worst results in benchmarks.

In the second set of benchmarks, mknapcb2-xx, XBPSO2 shows better average results than others, particularly in mknapcb2-5.250-04 and mknapcb2-5.250-04. In two benchmarks, the performance of XBPSO2 has increased with the number of dimensions of the problems. Moreover, BHTPSO-QI provides the better solutions in terms of the best profit in 5.250-00 and 5.250-01. In addition, the proposed method performs well in terms of the average best profit on the third set of benchmarks which are more difficult than the previous ones. However, TV-BPSO achieves the best solution compared to the others on 5.500-04 benchmark.

In other high-dimensional benchmarks, SS_BPSO obtains better results as compared with other algorithm in the majority of benchmarks; however, TV-BPSO shows the better performance on 10.500-00, 10.500-01, 10.500-03 and 10.500-04. In other cases, especially in the most difficult benchmarks, 30.250-29, 30.500-28 and 30-500-29, XBPSO2 performs much better solution than the others. The results show that the proposed method performs well on the optimization of set of 0-1 MKP instances. Additionally, BHTPSO-QI provides better solutions than other V-shaped BPSO algorithms and TV-BPSO presents better results than other S-shaped BPSO algorithms in the table.

Table 5. The best, average, worst and standard deviation (STD) of the best solution obtained by compared algorithms for the 0-1 MKP benchmarks

Benchmark / *Algorithm*	mknapcb1-5.100-00				mknapcb1-5.100-01				mknapcb1-5.100-02			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	**24184**	**23714.8**	**23343**	250.703	**24082**	**23464.2**	**22839**	301.106	**23457**	**22994.7**	**22176**	282.353
BPSO1	23531	23064.5	22190	311.761	23482	22867.8	22196	320.815	22994	22242.6	21677	325.72
VBPSO	23610	22789.7	21800	420.719	22977	22351	21317	389.443	22498	21905.9	21038	385.258
PSO-bin	23302	22315.5	21189	507.39	22768	22005.3	20745	468.509	22137	21212.9	20211	553.463
BHTPSO-QI	24067	23252	21905	554.701	23751	23248.7	22294	333.219	23250	22637.9	21779	303.714
TV-BPSO	23595	23150.2	22499	276.995	23536	22989.8	22287	349.155	23115	22407.1	21695	363.483
BGSA	23301	22631.7	21888	417.263	23286	22183.8	21240	524.116	22583	21743.2	20937	488.824
WOA-SA	23393	22320.5	21212	515.161	22577	21906	20653	538.708	21995	21224.2	20203	509.963
BBA	21985	20554.1	19714	513.151	21280	20284.6	19554	445.284	20707	19587.1	19048	451.479
Bin-ABC	21321	20649.2	20026	354.138	21289	20614.8	20072	322.542	20613	19575.7	18964	333.444

Benchmark / *Algorithm*	mknapcb1-5.100-03				mknapcb1-5.100-04				mknapcb2-5.250-00			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	**23245**	**22788.6**	**22465**	175.343	**23617**	**23293.8**	**22845**	199.044	56172	**55659.3**	**55148**	352.059
BPSO1	23009	22206.9	21183	367.184	23420	22788.6	22109	286.109	54058	52389.9	50755	1075.18
VBPSO	22930	22166.1	21310	331.301	23262	22537.3	21603	378.312	54155	52354.2	50330	998.76
PSO-bin	22296	21571.6	20883	339.397	22722	21866.1	20756	434.893	51780	50548.5	48905	973.881
BHTPSO-QI	23168	22394.9	21110	461.461	23547	23006	22126	359.487	**56812**	55291.5	54267	760.786
TV-BPSO	22981	22458.7	21072	382.359	23399	22846.9	21781	382.832	56553	54732.7	52803	1241.79
BGSA	22522	21973.1	21419	330.119	22892	22292.2	21704	286.849	53045	52179.6	51317	504.312
WOA-SA	22064	21603.3	20995	289.378	23209	22011	20906	514.029	51786	50097.5	48434	1090.06
BBA	20980	20185.1	19275	409.003	21323	20204.2	19669	409.745	46871	45979.5	44867	646.275
Bin-ABC	21219	20320.6	19823	306.907	21070	20202.9	19677	358.729	48242	47478.3	46742	523.337

Benchmark / *Algorithm*	mknapcb2-5.250-01				mknapcb2-5.250-02				mknapcb2-5.250-03			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	58901	**57861.9**	**57017**	566.388	**59638**	**58516.2**	**57209**	745.955	**56834**	**56191.3**	**55470**	440.771
BPSO1	56718	54168.6	51607	1404.49	56650	54717.2	53565	953.701	54374	52975.1	51666	836.12
VBPSO	55664	54699.3	53350	751.861	56560	55138	53537	857.486	54248	52867.6	50681	1017.6
PSO-bin	53548	52127	50297	875.464	54795	52703.4	51702	862.306	53203	51904.7	50670	780.901
BHTPSO-QI	**59057**	56364.9	50993	2212.62	59382	57837.5	56223	1238.81	56686	54735	52745	1371.1
TV-BPSO	57791	56751	55724	691.712	58730	57355	55445	1019.12	56013	54911.1	53475	621.146
BGSA	55587	54248.8	52762	759.586	56017	54859.3	54020	736.498	53947	52520.6	50797	1040.3
WOA-SA	55461	52366.5	50816	1475.95	54845	52839.9	51174	1113.78	51195	50405.5	49530	499.915
BBA	48746	47550.8	45569	898.945	50061	49032.7	47570	847.944	48294	47498.3	45503	886.072
Bin-ABC	49784	49375.1	48846	310.072	50680	50456.8	50289	140.995	49920	49332.2	48729	462.412

Table 5. *Continued.*

Benchmark / ***Algorithm***	**mknapcb2-5.250-04**				**mknapcb3-5.500-00**				**mknapcb3-5.500-01**			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	**56354**	**55551.8**	**54352**	592.816	109174	**107939**	**107194**	699.98	107061	**105386**	**104148**	927.081
BPSO1	53441	52027.1	50563	834.236	104301	102180	99970	1123.83	102227	99233.5	96675	1607.55
VBPSO	54215	52853.5	51850	685.694	105149	102924	100819	1478.28	103217	100453	98152	1399.44
PSO-bin	51515	50598.7	49241	796.549	102005	98952.1	96071	1848.76	99881	96592.2	93010	2102.07
BHTPSO-QI	55706	54500.1	52867	1061.98	**111160**	102784	95038	5353.76	**108619**	102563	98137	3679.04
TV-BPSO	55606	54609	52977	807.587	109747	108370	106697	854.709	107327	105193	102615	1365.33
BGSA	54889	52410.5	51314	1102.82	105763	103187	101669	1471.34	101421	99054	97088	1212.42
WOA-SA	51648	50101.1	48776	790.436	102492	98499.5	96177	1809.35	98797	96270	94532	1453.12
BBA	47499	46740.6	45999	556.218	95401	91047.7	88198	2383.3	92580	90651.7	87002	1662.42
Bin-ABC	48349	48048.3	47557	256.311	97476	96233.4	94982	799.566	95135	94503.7	93985	413.127

Benchmark / ***Algorithm***	**mknapcb3-5.500-02**				**mknapcb3-5.500-03**				**mknapcb3-5.500-04**			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	110462	**109045**	**107862**	870.735	109746	**108203**	**106249**	991.097	**111822**	109690	106630	1509.63
BPSO1	104829	102724	99992	1495.58	103289	101176	99055	1360.59	105725	103201	101656	1220.93
VBPSO	106226	103566	100889	1456.17	106507	103389	100025	1844.48	106077	103661	100752	1931.02
PSO-bin	102103	100863	99263	874.677	101404	99229.7	96010	1794.45	100815	99032.4	96070	1412.24
BHTPSO-QI	**111591**	105878	99482	4483.84	**112464**	106124	98236	4902.58	112691	108849	104260	3160.97
TV-BPSO	110086	108520	106388	1123.03	111028	108365	105876	1580.96	111397	**109775**	**107713**	1283.1
BGSA	106493	103184	99521	2327.5	103724	102140	99345	1288.64	105905	103394	101362	1531.29
WOA-SA	100771	99318.1	96573	1405.72	102487	98132.4	96175	1866.87	101840	98934.4	95844	1976.7
BBA	95003	92590.4	90900	1188.8	95144	92841.9	91034	1375.05	95303	83272	79.937	29268.2
Bin-ABC	100219	97902.7	96428	1123.82	97830	96765.5	96016	654.678	99270	97525.5	96556	818.256

Benchmark / ***Algorithm***	**mknapcb4-10.100-00**				**mknapcb4-10.100-01**				**mknapcb4-10.100-02**			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	**22745**	**22099.6**	21603	356.359	**22302**	**21893.7**	**21017**	353.771	**21760**	**21336.7**	**20707**	288.906
BPSO1	22172	21576.5	20977	379.405	21502	20990.2	20172	397.461	21379	20556.6	19934	541.307
VBPSO	22303	21514	20825	425.447	21792	21217.2	20886	315.263	20885	20309	19688	359.334
PSO-bin	21471	20879.8	20051	454.146	21075	20541.2	20006	355.264	20531	20034.7	18913	513.852
BHTPSO-QI	22387	22027.5	**21616**	267.213	22084	21424.2	19821	667.009	21314	20956.5	20188	355.894
TV-BPSO	22505	21750.6	21442	346.227	21971	21445.9	20934	262.003	21112	20827.6	19974	357.092
BGSA	21505	21081.6	20413	353.587	21488	20778.1	20444	329.819	20861	20267.2	19485	403.506
WOA-SA	21570	20751.5	20175	517.755	21328	20490.3	19985	442.963	20454	19936.8	19326	408.723
BBA	19941	19212	18639	408.632	19478	18763	18118	491.334	19162	18471.4	17817	398.146
Bin-ABC	19802	19413.8	19023	245.962	19436	19127.3	18845	205.517	18994	18632.6	18281	216.098

Table 5. *Continued.*

Benchmark / Algorithm	mknapcb4-10.100-03				mknapcb4-10.100-04				mknapcb5-10.250-00			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	22024	**21860.6**	**21547**	147.7	**22302**	**21933.1**	**21332**	288.709	**55664**	**55067.2**	**54215**	463.129
BPSO1	21601	21139.4	20620	363.938	21618	21162.5	20719	314.628	53271	51638.5	50688	1089.06
VBPSO	21710	21297.8	20863	274.825	21578	20884.2	20214	429.239	53750	52003.6	50554	963.791
PSO-bin	21351	20578.7	19867	468.876	21133	20314.1	19547	507.132	52274	50455.2	49312	918.615
BHTPSO-QI	**22065**	21403.3	20306	620.706	22003	21388	20353	583.924	54805	52577.2	50180	1591.85
TV-BPSO	21770	21604.8	21305	147.552	22051	21349.9	20964	338.872	55238	54229.6	52746	933.143
BGSA	21315	20727.6	20196	335.324	21564	20712.3	19529	692.606	54109	51861.8	50280	1115.94
WOA-SA	21168	20435.9	19508	450.622	21186	20186.3	19233	531.298	51986	49713.3	48313	1057.71
BBA	20079	19127.8	18472	435.961	19374	18713.5	18037	416.057	48433	46060.2	44443	1171.78
Bin-ABC	20330	19633.2	19240	331.253	19514	19116.1	18777	239.362	49312	47965.4	47337	521.274
Benchmark / Algorithm	**mknapcb5-10.250-01**				**mknapcb5-10.250-02**				**mknapcb5-10.250-03**			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	**55717**	**54640.5**	**53003**	827.123	**54864**	**53966.8**	**52798**	591.743	**58401**	**56928.9**	**55457**	808.777
BPSO1	52543	51307.2	49821	874.004	51559	50713.4	49708	645.159	54690	53476	52405	698.922
VBPSO	52551	51374.1	49598	861.975	51895	50397.5	49166	899.577	54905	53138.5	51952	1079.62
PSO-bin	52415	49966.4	48489	1369.89	50075	48983.1	47695	831.736	53755	51751.4	48953	1547.14
BHTPSO-QI	55138	53308.7	50890	1616.35	54646	52771.7	50153	1544.42	58213	55968	52919	1823.49
TV-BPSO	55306	53988.4	52528	885.648	54525	52843.1	51585	812.932	56672	55975.6	54847	652.182
BGSA	53105	51953.3	50134	1020.78	51852	50744.6	49425	796.403	54593	53807.3	52942	640.356
WOA-SA	51306	50013.8	48436	872.643	49536	48663.1	46524	871.685	52570	51556.1	50725	687.932
BBA	47323	45977	44696	896.412	45641	44650.3	43803	659.92	48548	46845.4	45866	737.787
Bin-ABC	48290	47719.3	47118	383.872	47369	46766	46226	411.576	49653	49179.7	48637	303.096
Benchmark / Algorithm	**mknapcb5-10.250-04**				**mknapcb6-10.500-00**				**mknapcb6-10.500-01**			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	**55150**	**54074.7**	**52944**	660.429	107016	104517	101431	1683.7	107460	105613	**104038**	1182.3
BPSO1	51409	50447.9	49749	548.155	101511	99160.6	95914	1403.34	102787	100005	97220	1829.69
VBPSO	52339	51291.7	50728	584.018	101951	100228	97080	1458.48	102857	100532	98090	1251.05
PSO-bin	51622	49669.3	47359	1340.77	98155	96683.1	93884	1350.45	99518	97978	95861	1132.7
BHTPSO-QI	53871	51245.1	48495	1834.62	104231	100366	94773	3051.55	106408	100861	95897	3565.95
TV-BPSO	54289	53612.1	52450	540.21	**107985**	**105619**	**103995**	1224.13	**107908**	**105650**	103535	1249.05
BGSA	52396	50726.7	49222	1000.66	102224	100849	98803	1137.17	102319	101280	99839	708.546
WOA-SA	50266	48970.5	47563	990.431	99803	97394.5	93874	1690.65	98971	96479.2	91943	2230.76
BBA	46833	45317.2	44599	749.214	92056	89816	84273	2248.37	92843	90617	86420	1733.51
Bin-ABC	48095	47588.8	46901	361.842	96315	95265.7	94183	699.838	96924	95796.2	95047	686.444

Table 5. *Continued*.

Benchmark / Algorithm	mknapcb6-10.500-02				mknapcb6-10.500-03				mknapcb6-10.500-04			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	107357	**105678**	**104285**	1084.78	105819	105033	**104024**	667.68	104163	102421	99441	1320.89
BPSO1	100757	99340.3	98170	885.804	100235	99240	97336	823.681	99752	98223.1	97022	880.359
VBPSO	103654	101123	98466	1811.5	102089	99665.3	96312	1627.56	100524	98268.7	95534	1481.55
PSO-bin	99411	97408.8	95994	1176.18	98075	96991.6	95630	955.124	97874	94956.7	92474	1691.03
BHTPSO-QI	**110196**	103854	97980	4557.16	106115	100946	97969	2573.29	**108345**	100738	96755	3698.82
TV-BPSO	107843	106016	103073	1594.59	**107211**	**105444**	102177	1510.35	105473	**104274**	**102930**	956.485
BGSA	102921	101087	99530	1137.99	102712	100835	99427	1148.2	102250	99604	96569	1812.33
WOA-SA	99291	96916.4	93970	1716.55	100199	97145.2	94657	1576.26	98173	95959	94711	1105.5
BBA	93546	90267.4	86128	2227.62	91720	89197.4	86551	1795.91	91085	88865.2	85996	1600.2
Bin-ABC	96906	96066.7	95109	681.978	95689	94731.3	93782	617.468	94610	93642.9	92555	675.563
Benchmark / Algorithm	**mknapcb8-30.250-29**				**mknapcb9-30.500-28**				**mknapcb9-30.500-29**			
	Best	Average	Worst	STD	Best	Average	Worst	STD	Best	Average	Worst	STD
XBPSO2	**147500**	**146916**	**146340**	367.957	**294229**	**292830**	**291690**	887.599	**290925**	**289684**	**288828**	681.057
BPSO1	144846	143984	143329	434.547	289620	286218	284384	1409.03	286716	284961	282992	1419.79
VBPSO	145285	144383	143434	562.422	290940	287813	286236	1472.61	286970	285103	282546	1494.18
PSO-bin	143300	142242	140856	614.683	285826	284242	282173	1228.91	283363	282119	280264	981.881
BHTPSO-QI	147325	145901	143681	1105.75	292174	289404	286354	2037.59	290725	286148	281599	3360.23
TV-BPSO	146493	145742	144649	579.507	293985	292294	289528	1456.21	291377	289119	286872	1374.16
BGSA	145230	143768	142595	962.404	289094	286661	284624	1684.94	286949	284689	283101	1144.2
WOA-SA	143771	142269	141265	874.39	287117	283677	280543	1841.3	284382	281850	278158	1721.35
BBA	135313	132401	130309	1601.52	246235	243302	238872	2684.5	253117	241642	237395	4739.66
Bin-ABC	138574	137967	137449	385.132	278991	277830	276864	761.531	276392	275508	274859	508.903

To compare the results statistically, a non-parametric statistical test, Friedman test, (Derrac et al. 2011) is conducted at the significance level of 0.05. Table 6 illustrates the results of Friedman test ranks based on the average error obtained by algorithms on the 0-1 MKP benchmarks. As observed in this table, XBPSO2 algorithm has minimum sum error compared to other algorithms. Moreover, TV-BPSO algorithm has the second rank in order to achieve the lower error. The results show that BBA and Bin-ABC return the worst results.

The progress of the average best solutions on 10-100-04 and 10-250-02 MKP benchmark instances has been shown in Fig. 4. As shown in this figure, XBPSO2 tends to find the best solution faster than the other algorithms, also BBA and Bin-ABC algorithms show the slow convergence rate.

To evaluate the performance of algorithms in solving the 0-1 MKP benchmarks, the average errors obtained by algorithms are computed as follows (Beheshti et al. 2013):

$$AE = \frac{1}{N} \sum_{i=1}^{N} \frac{t_i - y_i}{t_i} \times 100, \quad (45)$$

where N is the number of benchmarks. t_i and y_i are the maximum profit and the average profit obtained by the i^{th} benchmark, respectively.

Table 6. The average Friedman ranks of error for the 0-1 MKP benchmarks

Benchmark \ Algorithm	XBPSO2	BPSO1	VBPSO	PSO-bin	BHTPSO-QI	TV-BPSO	BGSA	WOA-SA	BBA	Bin-ABC
mknapcb1-5.100-00	1.333	3.933	5	6.633	3.133	3.4	5.867	6.833	9.4	9.467
mknapcb1-5.100-01	1.633	3.6	5.7	6.733	2.2	3.3	5.967	6.9	9.733	9.233
mknapcb1-5.100-02	1.333	3.967	5.267	7.067	2.567	3.267	5.567	7.067	9.467	9.433
mknapcb1-5.100-03	1.6	4.45	4.333	6.933	3.317	3.067	5.233	7.067	9.6	9.4
mknapcb1-5.100-04	1.433	3.7	4.967	7.2	2.6	3.533	5.967	6.6	9.567	9.433
mknapcb2-5.250-00	1.5	4.9	5	7.3	2.1	2.4	5.4	7.4	10	9
mknapcb2-5.250-01	1.4	5.1	4.6	7.4	2.8	2.6	4.9	7.2	10	9
mknapcb2-5.250-02	1.5	5.2	4.7	7.3	2	2.7	5.3	7.3	10	9
mknapcb2-5.250-03	1.1	4.8	5	6.7	3	2.5	5.1	7.8	10	9
mknapcb2-5.250-04	1.4	5.3	4.4	7.3	2.4	2.5	5.1	7.6	10	9
mknapcb3-5.500-00	2.2	4	4.6	6.9	1.8	3.2	6.3	7	9.7	9.3
mknapcb3-5.500-01	1.6	4.9	4.1	6.6	3.1	3.1	5.8	6.8	9.7	9.3
mknapcb3-5.500-02	1.6	4.5	5.2	6.2	2.8	3.2	5.6	6.9	9.4	9.6
mknapcb3-5.500-03	1.5	4.4	3.9	6.7	3.5	2.7	6.2	7.4	9.9	8.8
mknapcb3-5.500-04	1.4	3.8	5.2	6.8	3	3.2	5.7	7	9.9	9
mknapcb4-10.100-00	2	5	4.7	7.1	4.5	1.4	4.2	7.6	10	8.5
mknapcb4-10.100-01	1.8	5.4	4	7.7	3.2	1.9	4.9	7.3	10	8.8
mknapcb4-10.100-02	1.9	5.1	4.7	6.5	3.8	1.8	4.7	7.9	10	8.6
mknapcb4-10.100-03	1.9	5.4	4.3	7.1	3.1	1.9	4.8	7.8	10	8.7
mknapcb4-10.100-04	2	5.1	4.6	7.5	2.5	1.9	5.1	7.6	10	8.7
mknapcb5-10.250-00	1.2	4.7	4.7	6.7	4.2	2.1	4.9	7.5	9.9	9.1
mknapcb5-10.250-01	1.5	5.1	5.3	7.1	3.3	2	4.6	7.1	10	9
mknapcb5-10.250-02	1.3	4.9	5.2	7.2	2.3	2.7	4.8	7.7	10	8.9
mknapcb5-10.250-03	1.5	5.2	5.8	6.9	2.3	2.5	4.4	7.4	10	9
mknapcb5-10.250-04	1.2	5.35	4.1	6.5	4.9	1.9	4.95	7.3	10	8.8
mknapcb6-10.500-00	1.8	5.8	4.7	7.6	4.1	1.3	4	7.2	10	8.5
mknapcb6-10.500-01	1.7	5.2	4.8	6.5	4.8	1.6	4	7.8	10	8.6
mknapcb6-10.500-02	2	5.7	4.3	7.4	3.3	1.7	4.2	7.9	10	8.5
mknapcb6-10.500-03	1.7	5.6	5.1	7.2	4	1.4	3.7	7.5	10	8.8
mknapcb6-10.500-04	2.1	4.9	5	7.8	3.7	1.5	4.3	7	10	8.7
mknapcb8-30.250-29	1.2	4.9	4.3	7.4	2.4	2.8	5.6	7.4	10	9
mknapcb9-30.500-28	1.6	5.3	4.1	7.4	3.3	1.7	5.3	7.3	10	9
mknapcb9-30.500-29	1.6	4.6	4.8	7.2	3.8	2.1	4.6	7.3	10	9
Sum	**52.532**	159.8	156.467	232.566	103.817	78.867	167.051	241.467	326.267	296.166
Rank	**1**	5	4	7	3	2	6	8	10	9

(a)
(b)

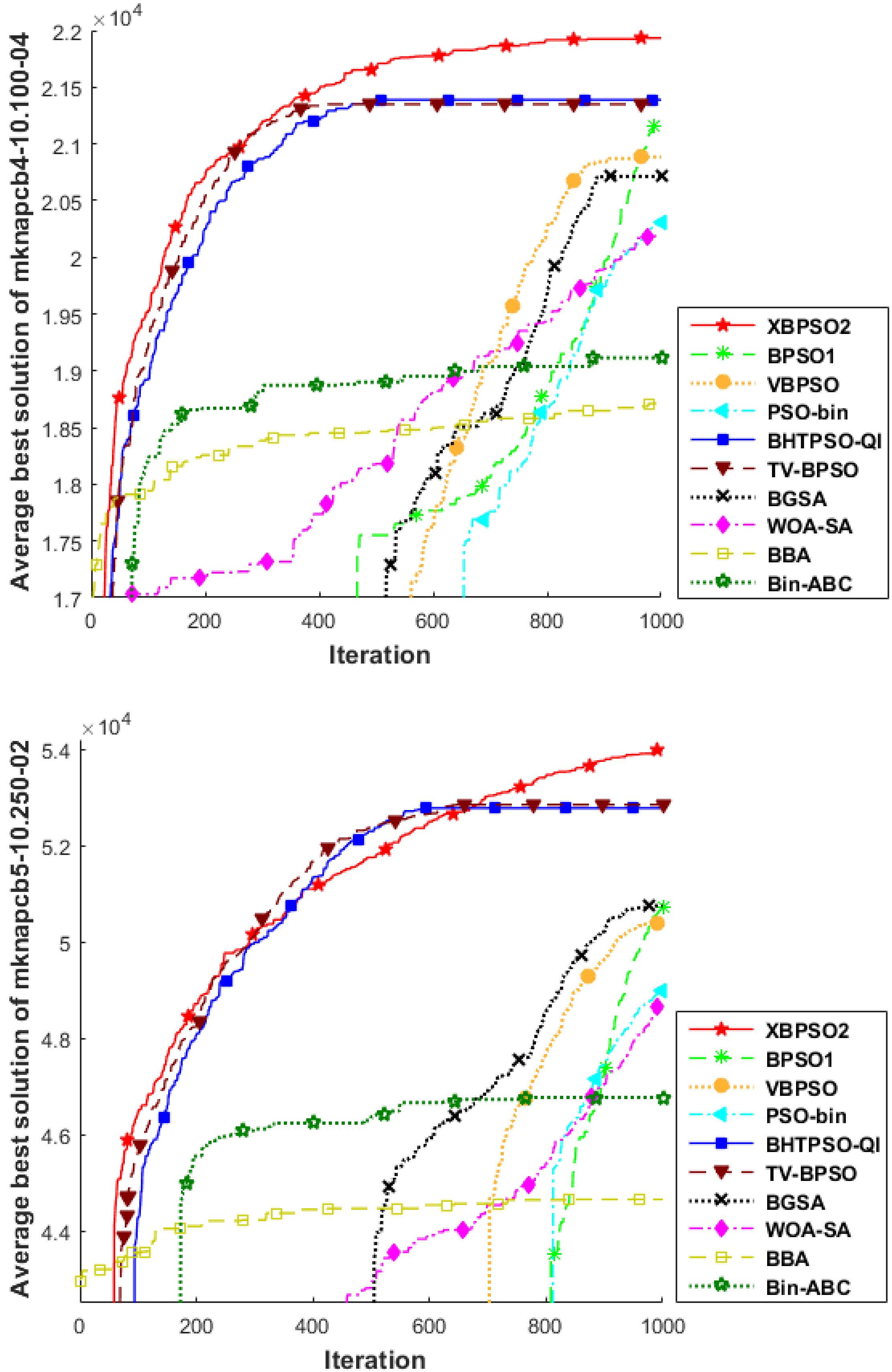


Fig. 4. Convergence curves for the proposed algorithm and other binary algorithms on (a) 10-100-04 and (b) 10-250-02 MKP benchmark instances

Fig. 5 shows the average error of algorithms on the 0-1 MKP benchmark instances. As seen, XBPSO2 shows the minimum error, 8.9%, compared to other binary algorithms. After the proposed method, TV-BPSO illustrates a lower error. The maximum error belongs to BBA with 22.67%. In other words, BBA cannot tune itself and has fallen into local optima in these benchmarks.

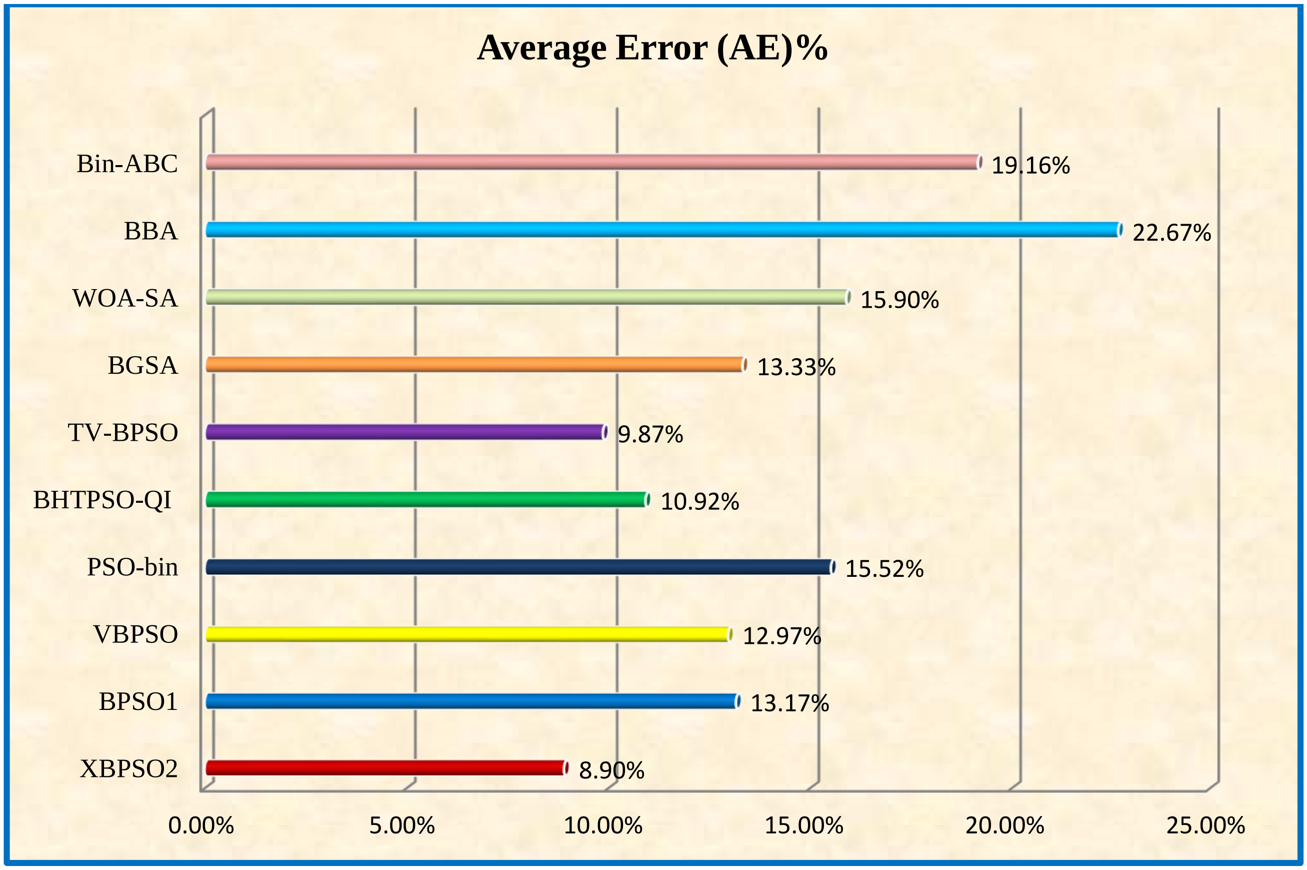


Fig. 5. The average error of algorithms on the 0-1 MKP benchmark instances

4.4. The results of binary benchmark functions

The best algorithms from the previous section are selected and applied for solving maximization functions. XBPSO2, BPSO1, VBPSO, BHTPSO-QI, TV-BPSO, and BGSA are chosen to compare their performances on these functions. Moreover, the local topology of XBPSO2 (LXBPSO2), BPSO1 (LBPSO1) and VBPSO (LVBPSO) are implemented to evaluate the efficiency of transfer functions. A ring topology with two neighbours is considered for the local topology.

The average and standard deviation of the best results for maximization benchmark functions have been demonstrated in Table 7. In the table, different low and high dimensions are considered for Max-ones (dimensions 150, 250, 300, and 350) and Royal-road (dimensions 32, 40, 56, and 96) functions.

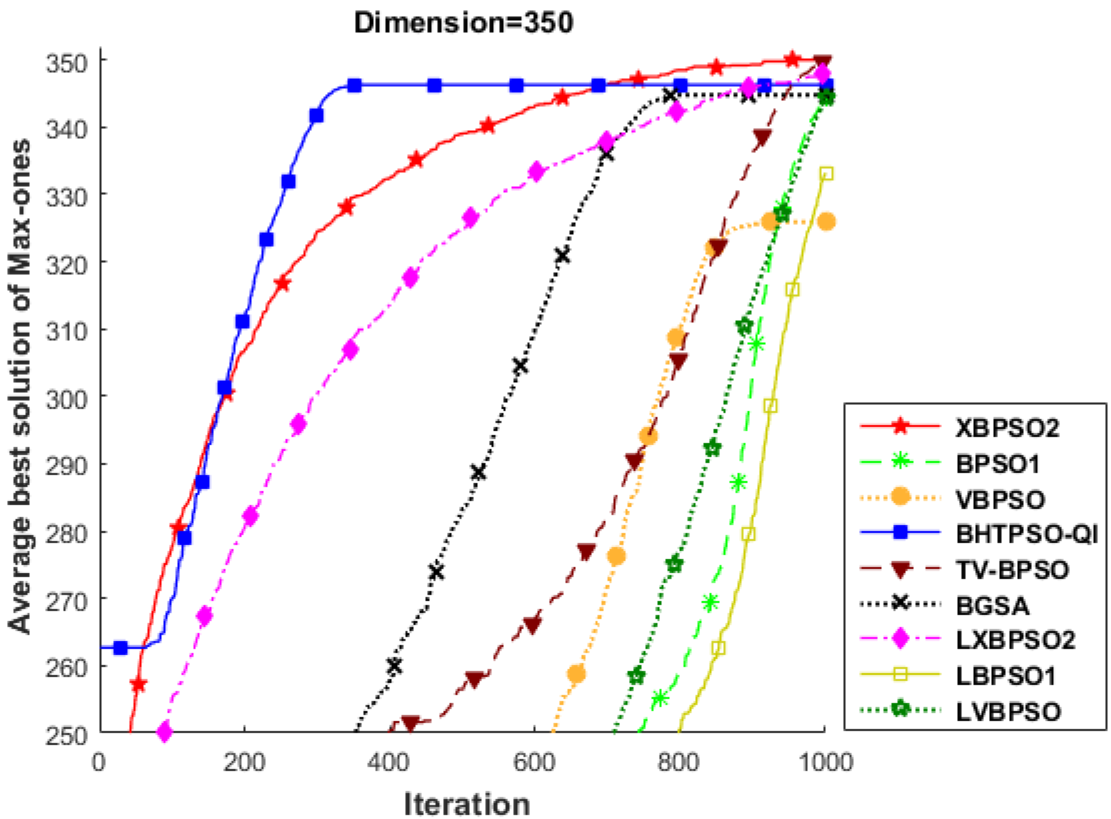
As observed in Table 7, XBPSO2 achieves the global optimum in Max-ones function for both low and high dimensions. LXBPSO2 also shows a good performance in the function. BPSO1 has better results than LBPSO1 in this function; while, LVBPSO provides better solution than VBPSO in this case. In Royal-road, BGSA achieves the best solution in all cases. LXBPSO2 shows better results than XBPSO2 in this function. It acquires the global optimum for dimensions 32, 40 and 56. Although TV-BPSO shows a good performance in solving the 0-1 MKP, it cannot tune itself in this function and does not obtain the global optimum in low and high dimensions. In this function, LVBPSO also has better results than VBPSO. In summary, XBPSO2 and LXBPSO2 perform much better solution than the other algorithms in these functions.

Table 7. The average and standard deviation (STD) of the best solution obtained by algorithms for maximization benchmark functions

Function / Algorithm	Max-ones Dimension=150	Max-ones Dimension=250	Max-ones Dimension=300	Max-ones Dimension=350
XBPSO2	**150±0**	**250±0**	**300±0**	**350±0**
BPSO1	**150±0**	249.8±0.42164	298.2±1.3166	344.3±1.3375
VBPSO	147.9±0.9944	240.6±2.2706	281.9±4.0947	325.9±5.1951
BHTPSO-QI	**150±0**	249.4±0.84327	297.8±1.6865	346.3±1.7029
TV-BPSO	**150±0**	**250±0**	**300±0**	349.7±0.48305
BGSA	149.9±0.3162	248.7±1.0593	298.3±1.8886	344.8±1.9322
LXBPSO2	**150±0**	**250±0**	299.8±0.4216	347.9±0.99443
LBPSO1	**150±0**	247±1.1547	290±1.4142	333.2±1.9889
LVBPSO	**150±0**	**250±0**	299.7±0.4831	344.5±2.1213

Function / Algorithm	Royal-road Dimension=32	Royal-road Dimension=40	Royal-road Dimension=56	Royal-road Dimension=96
XBPSO2	3.9±0.3162	4.9±0.3162	6.3±0.67495	9.30±823
BPSO1	3.6±0.5164	4.6±0.5164	5.3±0.67495	7.2±1.3166
VBPSO	3.1±0.9944	3.7±0.8233	4.6±0.96609	7.2±1.2293
BHTPSO-QI	3.3±0.4831	4.2±0.7888	5.5±0.97183	8.1±1.1972
TV-BPSO	3.9±0.3162	4.5±0.5271	5.7±0.67495	7.7±0.8233
BGSA	**4±0**	**5±0**	**7±0**	**12±0**
LXBPSO2	**4±0**	**5±0**	**7±0**	11.7±0.483
LBPSO1	**4±0**	4.7±0.4831	6.1±0.56765	6.9±0.5677
LVBPSO	3.9±0.3162	4.6±0.6992	5.5±0.84984	7.9±0.9944

(a)



(b)

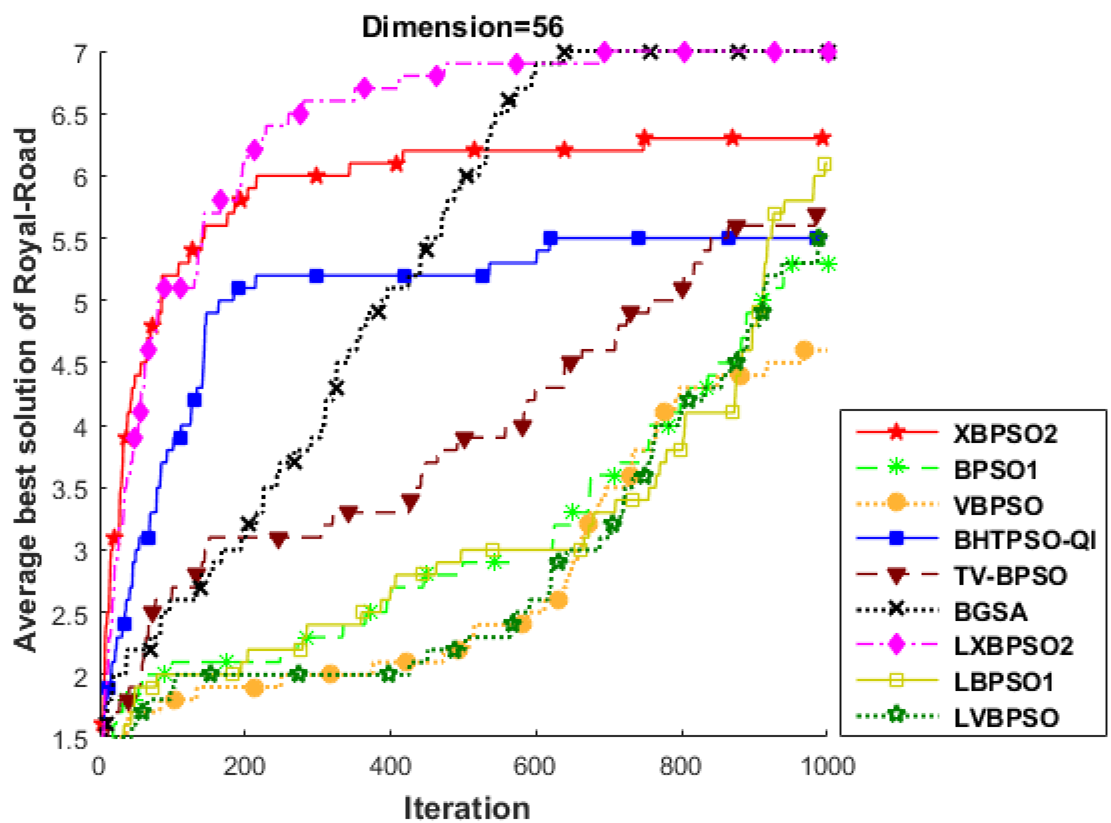


Fig. 6. Convergence curves for the proposed algorithm and other binary algorithms on (a) Max-Ones (*Dimension=350*) and (b) *Royal-road (Dimension=56)* functions

The superior convergence rate of XBPSO2 and LXBPSO2 on Max-Ones (*Dimension=350*) and Royal-Road (*Dimension=56*) functions has been shown in Fig. 6. According to this figure, the proposed transfer function increases the performance of BPSO to find the best solution in maximization functions.

Moreover, algorithms are compared based on the mean absolute error (MAE) on benchmark functions. MAE is computed as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} t_i - y_i \, , \tag{46}$$

where N is the number of benchmarks. t_i and y_i are the global optimum and the average best solution achieved by the i^{th} benchmark, respectively.

Fig. 7 demonstrates MAE achieved by algorithms on benchmark functions. As observed in this figure, LXBPSO2 and XBPSO2 illustrate minimum MAE on functions. It indicates that the global and local topologies of XBPSO have good performances for solving high dimensional functions. VBPSO provides the maximum MAE on these functions; however, local topology of VBPSO (LVBPSO) performs considerably better than VBPSO for solving these functions due to more exploration capability.

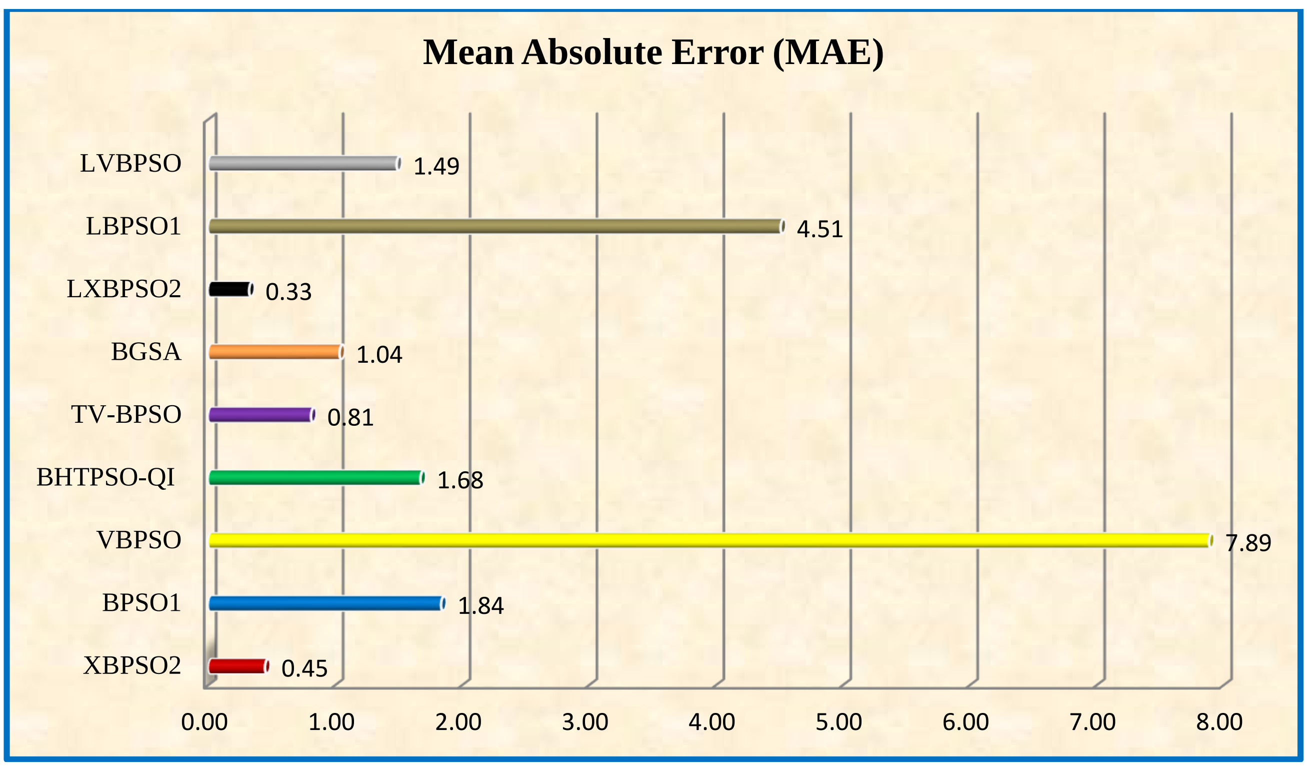


Fig. 7. The MAE of algorithms for maximization benchmark functions

5. Conclusion

In this study, a new X-shaped BPSO (XBPSO) has been introduced to improve the performance of BPSO. The standard BPSO encounters some disadvantages such as trapping into local optima and premature convergence due to its poor exploration. The proposed method employs two functions and applies enhanced rules to create a new binary solution. The best result obtained by the rules is compared to the previous solution. If the result is better than the previous one, the next position will be changed to the result; otherwise, the crossover operator is applied on the previous and the new solution. The best child from the operator is selected as the next position. Therefore, the exploration and the exploitation of BPSO will be enhanced by this transfer function. The proposed transfer function has been applied for global and local topologies of BPSO. Thirty-three benchmark instances of the 0-1 MKP and two maximization functions have been used to evaluate the performance of XBPSO. The results have been compared with some well-known BPSO algorithms and BGSA, WOA-SA, BBA as well as Bin-ABC. The results show that the proposed method outperforms the other methods for both global and local topologies. Moreover, the results of Friedman test demonstrate that the new transfer function significantly performs better than previous transfer functions. Additionally, the effect of different values of inertia weight (w) on the exploration ability of BPSO has been considered in this study. The results show that the standard BPSO provides better results, if the inertia weight is set to a small value in the first steps and gradually increases during the running of algorithm to achieve high performance searching. Meanwhile, the parameter w in VBPSO should be decreased linearly during the running of algorithm to have better results. In the proposed method, the best results obtained when w is set to one.

The new transfer function has a simple structure and shows better results than previous transfer functions but XBPSO may sometimes find local optima or exhibit slow convergence speed because of using the PSO's velocity. The local topology of XBPSO (LXBPSO) performs better than XBPSO because more exploration is performed by the swarm in LXBPSO.

For future studies, the proposed transfer function can be used in other improved PSO and continuous meta-heuristic algorithms to map the real search space to the binary one. Moreover, other binary optimization problems can be solved by XBPSO to evaluate the performance of X-shaped transfer function.

Compliance with Ethical Standards

The author declares that she has no conflict of interest.

This article does not contain any studies with human participants or animals performed by the author.

Informed consent was obtained from all individual participants included in the study.

References

Abdel-Basset, M. et al., 2018. A modified flower pollination algorithm for the multidimensional knapsack problem: human-centric decision making. *Soft Computing*, 22(13), pp.4221–4239. Available at: https://doi.org/10.1007/s00500-017-2744-y.

Ardizzon, G., Cavazzini, G. & Pavesi, G., 2015. Adaptive acceleration coefficients for a new search diversification strategy in particle swarm optimization algorithms. *Information Sciences*, 299, pp.337–378. Available at: http://www.sciencedirect.com/science/article/pii/S0020025514011669.

Bansal, J.C. & Deep, K., 2012. A modified binary particle swarm optimization for knapsack problems. *Applied Mathematics and Computation*, 218(22), pp.11042–11061.

Barisal, A.K., Panigrahi, T.K. & Mishra, S., 2017. A Hybrid PSO-LEVY flight algorithm based fuzzy PID controller for automatic generation control of multi area power systems: Fuzzy based hybrid pso for automatic generation control. *International Journal of Energy Optimization and Engineering (IJEOE)*, 6(2), pp.42–63.

Beasley, J.E., 1990. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society*, 41(11), pp.1069–1072. Available at: http://dx.doi.org/10.1057/jors.1990.166.

Beheshti, Z., Firouzi, M., et al., 2016. A new rainfall forecasting model using the CAPSO algorithm and an artificial neural network. *Neural Computing and Applications*, 27(8), pp.2551–2565.

Beheshti, Z., 2018. BMNABC: Binary Multi-Neighborhood Artificial Bee Colony for High-Dimensional Discrete Optimization Problems. *Cybernetics and Systems*, 49(7–8), pp.452–474. Available at: https://doi.org/10.1080/01969722.2018.1541597.

Beheshti, Z. et al., 2014. Enhancement of artificial neural network learning using centripetal accelerated particle swarm optimization for medical diseases diagnosis. *Soft Computing*, 18(11).

Beheshti, Z., Shamsuddin, S.M., et al., 2016. Improved centripetal accelerated particle swarm optimization. *International Journal of Advances in Soft Computing and its Applications*, 8(2), pp.1–26.

Beheshti, Z. & Shamsuddin, S.M., 2013. A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl*, 5(1), pp.1–35.

Beheshti, Z. & Shamsuddin, S.M., 2015. Non-parametric particle swarm optimization for global optimization. *Applied Soft Computing Journal*, 28.

Beheshti, Z., Shamsuddin, S.M. & Hasan, S., 2015. Memetic binary particle swarm optimization for discrete optimization problems. *Information Sciences*, 299, pp.58–84.

Beheshti, Z., Shamsuddin, S.M. & Sulaiman, S., 2014. Fusion global-local-topology particle swarm optimization for global optimization problems. *Mathematical Problems in Engineering*, 2014, pp.1–19.

Beheshti, Z., Shamsuddin, S.M. & Yuhaniz, S.S., 2013. Binary Accelerated Particle Swarm Algorithm (BAPSA) for discrete optimization problems. *Journal of Global Optimization*, 57(2), pp.549–573.

Beheshti, Z., Shamsuddin, S.M. & Yuhaniz, S.S., 2013. Binary Accelerated Particle Swarm Algorithm (BAPSA) for discrete optimization problems. *Journal of Global Optimization*, 57(2).

Beheshti, Z., Shamsuddin, S.M.H. & Hasan, S., 2013. MPSO: Median-oriented Particle Swarm Optimization. *Applied Mathematics and Computation*, 219(11), pp.5817–5836. Available at: http://www.sciencedirect.com/science/article/pii/S0096300312012805.

Bharti, O.P., Saket, R.K. & Nagar, S.K., 2017. Controller design for doubly fed induction generator using particle swarm optimization technique. *Renewable Energy*, 114, pp.1394–1406.

Chatterjee, S. et al., 2017. Particle swarm optimization trained neural network for structural failure prediction of multistoried RC buildings. *Neural Computing and Applications*, 28(8), pp.2005–2016.
Collotta, M., Pau, G. & Maniscalco, V., 2017. A fuzzy logic approach by using particle swarm optimization for effective energy management in IWSNs. *IEEE Transactions on Industrial Electronics*, 64(12), pp.9496–9506.
Derrac, J. et al., 2011. 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), pp.3–18.
Dong, J., Zhang, L. & Xiao, T., 2018. A hybrid PSO/SA algorithm for bi-criteria stochastic line balancing with flexible task times and zoning constraints. *Journal of Intelligent Manufacturing*, 29(4), pp.737–751.
Eberhart, R.C. & Shi, Y., 2001. Particle swarm optimization: developments, applications and resources. *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, 1, pp.81–86.
FAN, K., ZHANG, R. & XIA, G., 2007. Solving a Class of Job-Shop Scheduling Problem based on Improved BPSO Algorithm. *Systems Engineering - Theory & Practice*, 27(11), pp.111–117. Available at: http://www.sciencedirect.com/science/article/pii/S1874865108600678.
García-Nieto, J. et al., 2018. Multi-objective ligand-protein docking with particle swarm optimizers. *Swarm and Evolutionary Computation*. Available at: http://www.sciencedirect.com/science/article/pii/S2210650217304467.
Garg, H., 2016. A hybrid PSO-GA algorithm for constrained optimization problems. *Applied Mathematics and Computation*, 274, pp.292–305.
Islam, M.J., Li, X. & Mei, Y., 2017. A time-varying transfer function for balancing the exploration and exploitation ability of a binary PSO. *Applied Soft Computing*, 59, pp.182–196. Available at: http://www.sciencedirect.com/science/article/pii/S1568494617302326.
Jia, D., Duan, X. & Khan, M.K., 2014. Binary Artificial Bee Colony optimization using bitwise operation. *Computers & Industrial Engineering*, 76, pp.360–365. Available at: http://www.sciencedirect.com/science/article/pii/S0360835214002654.
Kamboj, V.K., 2016. A novel hybrid PSO--GWO approach for unit commitment problem. *Neural Computing and Applications*, 27(6), pp.1643–1655.
Kanwar, N. et al., 2017. Simultaneous allocation of distributed energy resource using improved particle swarm optimization. *Applied Energy*, 185, pp.1684–1693.
Kennedy, J., 1999. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. pp. 1931–1938.
Kennedy, J. & Eberhart, R.C., 1997. A Discrete Binary Version of the Particle Swarm Algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. Washington, DC, USA: IEEE Computer Society, pp. 4104–4108.
Kennedy, J. & Eberhart, R.C., 1995. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*. Perth, Australia, IEEE Service Center, Piscataway, NJ, pp. 1942–1948.
Kiran, M.S., 2015. The continuous artificial bee colony algorithm for binary optimization. *Applied Soft Computing*, 33, pp.15–23.
Kushwaha, N. & Pant, M., 2018. Link based BPSO for feature selection in big data text clustering. *Future Generation Computer Systems*, 82, pp.190–199. Available at: http://www.sciencedirect.com/science/article/pii/S0167739X17321854.
Lee, S. et al., 2008. Modified binary particle swarm optimization. *Progress in Natural Science*, 18(9), pp.1161–1166. Available at: http://www.sciencedirect.com/science/article/pii/S1002007108002189.
Li, X. et al., 2017. Optimal fractional order PID controller design for automatic voltage regulator

system based on reference model using particle swarm optimization. *International Journal of Machine Learning and Cybernetics*, 8(5), pp.1595–1605.

Lin, A. et al., 2018. Global genetic learning particle swarm optimization with diversity enhancement by ring topology. *Swarm and Evolutionary Computation*. Available at: http://www.sciencedirect.com/science/article/pii/S2210650218302918.

Lin, G. & Guan, J., 2018a. A hybrid binary particle swarm optimization for the obnoxious p-median problem. *Information Sciences*, 425, pp.1–17. Available at: http://www.sciencedirect.com/science/article/pii/S0020025516316334.

Lin, G. & Guan, J., 2018b. Solving maximum set k-covering problem by an adaptive binary particle swarm optimization method. *Knowledge-Based Systems*, 142, pp.95–107. Available at: http://www.sciencedirect.com/science/article/pii/S0950705117305580.

Liu, J., Mei, Y. & Li, X., 2016. An analysis of the inertia weight parameter for binary particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 20(5), pp.666–681.

Lynn, N., Ali, M.Z. & Suganthan, P.N., 2018. Population topologies for particle swarm optimization and differential evolution. *Swarm and Evolutionary Computation*, 39, pp.24–35. Available at: http://www.sciencedirect.com/science/article/pii/S2210650217308805.

Mafarja, M.M. & Mirjalili, S., 2017. Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. *Neurocomputing*, 260, pp.302–312. Available at: http://www.sciencedirect.com/science/article/pii/S092523121730807X.

Malik, A.J. & Khan, F.A., 2017. A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection. *Cluster Computing*. Available at: https://doi.org/10.1007/s10586-017-0971-8.

Manohar, L. & Ganesan, K., 2017. Diagnosis of Schizophrenia Disorder in MR Brain Images Using Multi-objective BPSO Based Feature Selection with Fuzzy SVM. *Journal of Medical and Biological Engineering*. Available at: https://doi.org/10.1007/s40846-017-0355-9.

Marinakis, Y., Migdalas, A. & Sifaleras, A., 2017. A hybrid Particle Swarm Optimization – Variable Neighborhood Search algorithm for Constrained Shortest Path problems. *European Journal of Operational Research*, 261(3), pp.819–834. Available at: http://www.sciencedirect.com/science/article/pii/S0377221717302357.

Mirjalili, S. & Lewis, A., 2013. S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. *Swarm and Evolutionary Computation*, 9(Supplement C), pp.1–14. Available at: http://www.sciencedirect.com/science/article/pii/S2210650212000648.

Mirjalili, S., Mirjalili, S.M. & Yang, X.-S., 2014. Binary bat algorithm. *Neural Computing and Applications*, 25(3), pp.663–681. Available at: https://doi.org/10.1007/s00521-013-1525-5.

Nezamabadi-pour, H. & Maghfoori-Farsangi, M., 2008. Binary particle swarm optimization: challenges and new solutions. *Journal of Computer Society of Iran (CSI) On Computer Scien ce and Engineering*, 6, pp.21–32.

Rashedi, E., Nezamabadi-Pour, H. & Saryazdi, S., 2010. BGSA: binary gravitational search algorithm. *Natural Computing*, 9(3), pp.727–745.

Sheikhan, M., 2017. Improvement of Prosody Modeling Using Semantic Role Labeling, Hybrid Feature Selection and BPSO--PSO--Optimized RNN. *National Academy Science Letters*, 40(3), pp.171–175. Available at: https://doi.org/10.1007/s40009-017-0545-6.

Shen, Q. et al., 2004. 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*, 22(2–3), pp.145–152.

Shi, Y. & Eberhart, R.C., 1998. Parameter selection in particle swarm optimization. In V. W. Porto et al., eds. *Evolutionary Programming VII*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 591–600.

Sun, Z., Liu, Y. & Tao, L., 2018. Attack localization task allocation in wireless sensor networks based on multi-objective binary particle swarm optimization. *Journal of Network and Computer*

Applications, 112, pp.29–40. Available at: http://www.sciencedirect.com/science/article/pii/S1084804518301103.

Taherkhani, M. & Safabakhsh, R., 2016. A novel stability-based adaptive inertia weight for particle swarm optimization. *Applied Soft Computing*, 38, pp.281–295. Available at: http://www.sciencedirect.com/science/article/pii/S1568494615006195.

Wang, J. et al., 2018. A Hybrid BPSO-GA Algorithm for 0-1 Knapsack Problems. In P. Krömer et al., eds. *Proceedings of the Fourth Euro-China Conference on Intelligent Data Analysis and Applications*. Cham: Springer International Publishing, pp. 344–351.

Wang, L. et al., 2008. A novel probability binary particle swarm optimization algorithm and its application. *Journal of software*, 3(9), pp.28–35.

Wei, J. et al., 2017. A BPSO-SVM algorithm based on memory renewal and enhanced mutation mechanisms for feature selection. *Applied Soft Computing*, 58, pp.176–192. Available at: http://www.sciencedirect.com/science/article/pii/S1568494617302430.

Yang, J. et al., 2014. Task Allocation for Wireless Sensor Network Using Modified Binary Particle Swarm Optimization. *IEEE Sensors Journal*, 14(3), pp.882–892.

Yuan, X. et al., 2009. An improved binary particle swarm optimization for unit commitment problem. *Expert Systems with Applications*, 36(4), pp.8049–8055. Available at: http://www.sciencedirect.com/science/article/pii/S0957417408007720.

Zhang, L. et al., 2015. A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques. *Applied Soft Computing*, 28, pp.138–149. Available at: http://www.sciencedirect.com/science/article/pii/S1568494614005808.

Zouache, D., Nouioua, F. & Moussaoui, A., 2016. Quantum-inspired firefly algorithm with particle swarm optimization for discrete optimization problems. *Soft Computing*, 20(7), pp.2781–2799. Available at: https://doi.org/10.1007/s00500-015-1681-x.