Information Sciences 512 (2020) 1503–1542

Contents lists available at ScienceDirect

Information Sciences

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

A time-varying mirrored S-shaped transfer function for binary particle swarm optimization

Zahra Beheshti [a,b]

[a] Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran
[b] Big Data Research Center, Najafabad Branch, Islamic Azad University, Najafabad, Iran

ARTICLE INFO

Article history:
Received 18 February 2018
Revised 8 October 2019
Accepted 14 October 2019
Available online 30 October 2019

Keywords:
Particle swarm optimization (PSO)
Binary particle swarm optimization (BPSO)
S-shaped and V-shaped transfer functions
Local and global topologies
the 0-1 multidimensional knapsack problem (MKP)
Time-varying mirrored S-shaped (TVMS) transfer function

ABSTRACT

Binary Particle swarm optimization (BPSO) is one of the most popular swarm intelligence algorithms to solve binary optimization problems. It has a few parameters, simple structure, and high execution speed. A transfer function is applied in BPSO to convert the continuous search space to the binary one. This algorithm and its variants can sometimes find local optima or exhibit slow convergence speed. Thus, many researchers have improved the structure of BPSO and its transfer function to overcome these shortcomings. In this study, a new time-varying mirrored S-shaped transfer function for BPSO (TVMS-BPSO) is introduced to enhance global exploration and local exploitation in the algorithm. The performance of the proposed transfer function has been compared with some well-known BPSO algorithms and binary meta-heuristic algorithms. These algorithms have been evaluated by CEC 2005 benchmark functions and set of 0–1 multidimensional knapsack problem (MKP) benchmark instances. The experimental results showed that the new transfer function significantly enhances the efficiency of BPSO for both local and global topologies in terms of solution accuracy and convergence speed.

1. Introduction

Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart in 1995 [22]. Due to its simple structure and low computational cost, PSO has been applied to solve wide range optimization problems. This algorithm has shown a good performance in many problems; however, due to poor exploration, it can sometimes find local optima and show slow convergence speed. Hence, many researchers have proposed several improved PSO algorithms to overcome these disadvantages [10,38,46].

A binary version of PSO (BPSO) using a sigmoid transfer function was introduced by Kennedy and Eberhart to solve discrete optimization problems [23]. BPSO uses the velocity of PSO; therefore, it faces the disadvantages of PSO [35]. The results of various transfer functions show that the role of an appropriate transfer function in BPSO is very important to enhance the performance of BPSO [3,18,33]. As a result, three categories of transfer functions namely S-shaped [23,33], V-shaped [3,4,33,35], and linear [1,47] have been introduced to convert the continuous search space to the binary one.

The S-shaped transfer functions apply the variants of sigmoid functions. In these transfer functions, if the velocity is positive, the next position will be zero or one. If a random number in the range [0,1] is not greater than the velocity, the next position will be one; otherwise, it will be zero. If the velocity has a negative value, the next position will be zero because the random number is positive. The BPSO with S-shaped transfer function encounters some shortcomings [35]. In

E-mail address: z-beheshti@iaun.ac.ir

https://doi.org/10.1016/j.ins.2019.10.029
0020-0255/

the standard PSO, the value of velocity in the negative and the positive directions shows that the particle, based on its previous position, should have a movement toward the best solution. If the velocity is zero in PSO, the next position will be equal to current position. In other words, the zero velocity shows that the new position should not be changed in PSO. But the new position in BPSO can be changed to zero or one by probability of 0.5. The linear transfer function faces these drawbacks, as well.

Nezamabadi-pour et al. proposed V-shaped transfer functions to cover these disadvantages [35]. There is no difference between the negative and positive velocity in the transfer function and a great movement is required to reach the optimum position. Also, if the velocity is zero, the next position will be the current position as PSO. The V-shaped transfer functions have shown better performances than the S-shaped transfer functions in solving many optimization problems [3,4,35]; however, due to the fact that they employ the velocity of PSO, they may trap in local optima. In PSO, if the best position found by all particles is a local optimum, all particles may converge to this position. According to the structure of V-shaped transfer functions, if the current position is a local optimum and the velocity tends to be zero, the new position will be the current position (local optimum).

In this study, a new time-varying mirrored S-shaped (TVMS) transfer function is introduced to improve the performance of BPSO. TVMS enhances global exploration and local exploitation in BPSO. In early steps, the proposed function provides stronger global exploration. In middle steps, it starts switching from exploration to exploitation and in final steps; the transfer function provides a low probability of changing bits that increases exploitation. The mirrored S-shaped functions also help to get better results. Sigmoid functions and their rules generate different results and the best result is selected. The proposed transfer function can be applied in all versions of BPSO algorithms to achieve better solutions as shown in the experimental results. The transfer function has been employed for the local topology of BPSO. The performance of global and local topologies of BPSO with the proposed transfer function has been evaluated by CEC 2005 benchmark functions [43] as well as 0–1 MKP benchmark instances [2]. The results have been compared with some various BPSO algorithms and several well-known binary swarm intelligence algorithms. The results showed that the efficiency of BPSO has been considerably improved by the proposed transfer function, compared with others, in terms of global optimality and convergence speed.

The rest of this study is organized as follows: A brief overview of PSO and BPSO algorithms is presented in Sections 2 and 3, respectively. TVMS-BPSO is described in great details in Section 4. The proposed transfer function is evaluated by CEC 2005 benchmark functions and 0–1 MKP benchmark instances and its results is compared with several BPSO algorithms and binary meta-heuristic algorithms in Section 5. Finally, concluding remarks and future research directions are presented in Section 6.

2. Particle swarm optimization for the continuous search space

PSO simulates the flocking behavior of birds to solve continuous optimization problems [22,38]. It is a population-based algorithm in which each particle (solution) of the swarm has a position,X_i, and a velocity, V_i, in the D-dimensional search space as follows:

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

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

where D is the number of dimensions)problem parameters(and N is the population size.

Every particle moves based on its personal best position and the swarm best position. Hence, the particle has the ability of flying towards a better space. The velocity and position of the i^{th} particle in the d^{th} dimension are computed 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 $w(t)$ is the inertia weight applied to make a balance between exploration and exploitation. C_1 and C_2 are the acceleration coefficients, *rand()* is a random number in [0,1]. Also, $pbest_i = (pbest_i^1, pbest_i^2, ..., pbest_i^D)$ is the personal best position of the i^{th} particle and $gbest = (gbest^1, gbest^2, ..., gbest^D)$ is the best position found by the swarm so far.

Clerc and Kennedy proposed a variant of PSO with constriction factor χ to enhance the convergence rate of PSO [16] as follows:

$$v_i^d(t+1) = \chi * \left[v_i^d(t) + C_1 * rand_1() * \left(pbest_i^d(t) - x_i^d(t)\right) + C_2 * rand_2() * \left(gbest^d(t) - x_i^d(t)\right)\right] \tag{5}$$

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \quad where \quad \varphi = C_1 + C_2, \quad \varphi > 4 \tag{6}$$

where C_1 and C_2 were set to 2.05 and χ was set to 0.729.

Although PSO algorithm was proposed in 1995, many researchers are still interested to improve the performance of PSO. As a result, various PSO algorithms have been suggested focusing on parameters control, multi-swarm and population topology, hybrid methods, and novel learning strategies.

As shown in (3), w, C_1 and C_2 are the parameters of PSO. In study [42], a linearly decreasing inertia weight was proposed to control exploration and exploitation in PSO. Taherkhani and Safabakhsh introduced an adaptive multi-dimensional inertia weight [44]. The inertia weight is determined in different dimensions for each particle.

In an early experience [17], C_1 and C_2 were set to constant values (equal to 2). After that, a time-varying acceleration coefficients PSO was proposed by Ratnaweera and Halgamuge [40]. The algorithm, called PSO-TVAC, uses the following equation to change C_1 and C_2 parameters in (3). The experimental results showed that the best ranges for C_1 and C_2 are 2.5–0.5 and 0.5–2.5, respectively.

$$C_j(t) = C_{j,min} + \frac{\left(C_{j,max} - C_{j,min}\right)}{T} \times t, \quad j = 1, 2, 3 \tag{7}$$

where $C_{min}$and C_{max} are two constant values. Additionally, t and T are current iteration and the maximum number of iterations, respectively.

Cheng and Yao introduced time-varying parameters for PSO, based on a novel operator [12]. The acceleration parameters C_1 and C_2 are adaptively determined, based on the value of the inertia weight in each dimension.

PSO uses two types of topologies to search, namely local and global topologies [23]. In the local PSO (LPSO), the next velocity is calculated based on the best position achieved by particle's neighbors *lbest*; whereas, in the global topology, the next velocity is computed using the best position obtained by all particles (*gbest*). The velocity of the local topology in the d^{th} dimension is computed as follows:

$$v_i^d(t+1) = w(t) * v_i^d(t) + C_1 * rand_1() * \left(pbest_i^d(t) - x_i^d(t)\right) + C_2 * rand_2() * \left(lbest_i^d(t) - x_i^d(t)\right) \tag{8}$$

LPSO has shown a better performance compared with PSO to solve multimodal optimization problems [5,6,10]. To have better solutions, several topological structures have been proposed such as ring, star, square and von Neumann topologies [24]. A fully informed PSO (FIPS) was introduced by Mendes et al. [32]. In this algorithm, the velocity of each particle is updated based on all of the particle's neighbors, not just the best neighbor. In another study, Liang and Suganthan suggested a dynamic multi-swarm PSO (DMS-PSO) with a dynamic neighborhood structure [27]. In this algorithm, first the small groups of particles are created then, the particles are regrouped so that the information obtained by particles is shared among new groups. Marinakiset et al. proposed a hybrid PSO algorithm with variable neighborhood search (VNS) algorithm [31]. The algorithm, called PSOLGENT, solves the constrained shortest path problem as an NP-hard problem. In this algorithm, the velocity of each particle is computed, based on the local and global topologies. Local search of VNS algorithm helps a better search in the search space of the problem. In other PSO algorithms such as unified PSO (UPSO) [37] and fusion global-local-topology PSO (FGLT-PSO) [7], both local and global topologies are combined to improve exploration and exploitation in PSO.

In some variant PSO algorithms, different meta-heuristic algorithms such as genetic algorithm, fruit fly optimization algorithm (FOA), gravitational search algorithm (GSA) and ant colony optimization (ACO) have been combined with PSO. Beheshti et al. [6] enhanced the performance of PSO, using Newton's laws of motion in centripetal acceleration PSO (CAPSO). They also improved CAPSO and used both local and global topologies to increase exploration and exploitation in PSO [8].

A social learning PSO (SL-PSO) was proposed by Cheng and Jin [13]. The algorithm applies social learning mechanisms in a way that every particle, except the best one, learns from better particles in the current sorted swarm. The population is sorted, based on the fitness values of particles. The algorithm also employs a dimension-dependent parameter method for parameter settings. In another study, Zhang et al. presented an improved SL-PSO algorithm using a differential mutation and a novel social learning PSO (DSPSO) to improve exploration and exploitation in SL-PSO algorithm [50].

Liang et al. presented a comprehensive learning PSO (CLPSO) [28]. In CLPSO, particles' positions are updated by learning from different historical personal best positions. To enhance the performance CLPSO, an improved CLPSO with a local optima topology (LOT) structure (CLPSO-LOT) was introduced by Zhang et al. [49]. The LOT sorts the dimensions of positions and generates a topology structure. Then, random elements from the topology are applied by the particle for learning. Moreover, a heterogeneous CLPSO algorithm (HCLPSO) was introduced by Lynn and Suganthan [30]. In this algorithm, the population is divided into two subpopulations to focus on exploration and exploitation. The comprehensive learning strategy is applied to create samples for both subpopulations. The samples of exploration-subpopulation are generated based on particles' personal best positions. In exploitation-subpopulation, samples are created based on the personal best positions of entire swarm. Also, some adaptive control parameters are applied in the sub groups to improve exploration and exploitation.

Wang et al. introduced a hybrid PSO algorithm with an adaptive learning strategy (ALPSO) [48]. In this algorithm, a self-learning based candidate generation strategy is used to enhance exploration. At first, all particles learn from the best particle (*gbest*). If the swarm is trapped into a local optimum, particles adjust their search direction and learn from a new particle to jump from this situation. In this algorithm, a tolerance based search direction adjustment mechanism has been designed to balance exploration and exploitation.

Tanweer et al. proposed a self-regulating PSO (SRPSO) algorithm [45]. This algorithm combines the best human learning strategies to find the optimum solution in PSO. Two learning strategies self-regulating inertia weight and self-perception are used in this algorithm. The best particle applies the self-regulating inertia weight to have a better exploration and the rest of particles use the self-perception to have an intelligent exploitation.

Jensi and Jiji introduced a levy flight method to update the particle's velocity in PSO [19]. The particle moves slowly towards its *pbest* and *gbest* to enhance the diversity of swarm for the global exploration. The levy walk is computed and applied to modify the particles' velocity.

A neighbor-based learning PSO with short-term and long-term memory was introduced for dynamic optimization problems [11]. In this algorithm, a neighbor-based learning strategy is incorporated into the particle's velocity. Besides, the worst replacement strategy is applied to update the particles. The worst particle's position is replaced by a better position newly created. The solutions from the most recent environment and the historical best solutions from previous environments are stored by the short-term and long-term memories, respectively. After detecting an environmental change, some particles' positions are replaced by some particles from the short-term memory, and the best member in the long-term memory is re-introduced to the active swarm along with its Gaussian neighborhood. Then, the other particles' positions are re-initialized. This algorithm and above various PSO algorithms have been introduced for the continuous search space. In the next section, the binary PSO and its variants are described for the discrete search space.

3. Particle swarm optimization for the binary search space

PSO algorithm has been designed for the continuous search space but many optimization problems are discrete (binary) optimization problems. Therefore, it is necessary to map the continuous search space to the binary one for solving these problems. A binary version of PSO (BPSO) was first introduced by Kennedy and Eberhart in 1997 [23]. The standard BPSO uses (3) to compute the next velocity. The algorithm applies a sigmoid transfer function to modify the continuous search space to the binary one as follows:

$$S\left(v_i^d(t+1)\right) = sigmoid\left(v_i^d(t+1)\right) = \frac{1}{1+e^{-v_i^d(t+1)}}. \tag{9}$$

Also, the new binary position of the i^{th} particle in the d^{th} dimension is calculated as follows:

$$xb_i^d(t+1) = \begin{cases} 1 & if \quad rand() < S\left(v_i^d(t+1)\right) \\ 0 & if \quad rand() \ge S\left(v_i^d(t+1)\right) \end{cases}, \tag{10}$$

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

Although BPSO has a simple structure, it suffers from some inherent disadvantages [35]. Some of the drawbacks are directly tied to the shortcomings of PSO and the others are related to the transfer function. PSO has poor exploration; therefore, it may trap into the local optimum. Since BPSO employs the velocity of PSO, it can sometimes find local optima or show slow convergence rate.

Another disadvantage of BPSO depends on the sigmoid function [35]. In the standard PSO, there is no difference between big values of velocity in positive or negative directions. A big absolute value of the velocity indicates that the current particle's position is not suitable and a great movement is required to reach the optimum position. Also, a small absolute value of the velocity shows that the current particle's position is close to the optimum solution and a small distance is needed to reach the optimum position. In BPSO, a value in the positive direction generates a bigger probability (probability of 1) and a value in the negative direction makes the probability of zero for the next particle position. In other words, the new solutions in different directions are obtained by different ways.

To overcome these drawbacks, many improved BPSO algorithms have been proposed so far. Shen et al. proposed a modified binary PSO (MBPSO) [41] which selects variables in MLR and PLS, based on the following rule. Ten percent of swarm randomly moves in the search space without following any rule to avoid entrapment by local optimum; however, this algorithm may find local optima in some cases.

$$xb_i^d(t+1) = \begin{cases} xb_i^d(t) & if \quad 0 < v_i \le a \\ pbest_i^d(t) & if \quad a < v_i \le \frac{1}{2}(1+a) \\ gbest^d(t) & if \quad \frac{1}{2}(1+a) < v_i \le 1 \end{cases}, \tag{11}$$

where v_i is a random number in [0,1] and a is a static probability changed from 0.5 to 0.33.

Lee et al. introduced a modified BPSO using the concepts of genotype and phenotype [26]. The binary and real positions are called phenotype and genotype, respectively. Moreover, a mutation operator is employed in this method to enhance exploration. The algorithm applies the binary position to update the velocity. It acquires the new position based on the real velocity and the current real position. Therefore, the new position has a real value, and it should be converted to a binary value by a sigmoid function as follows:

$$S\left(x_{g,i}^d(t+1)\right) = sigmoid\left(x_{g,i}^d(t+1)\right) = \frac{1}{1+e^{-x_{g,i}^d(t+1)}}, \tag{12}$$

$$x_{p,i}^{d}(t+1)=\begin{cases}1 & if \quad rand()<S\left(x_{g,i}^{d}(t+1)\right)\\ 0 & if \quad rand()\geq S\left(x_{g,i}^{d}(t+1)\right)\end{cases}, \tag{13}$$

where x_p and x_g are the phenotype and genotype positions, respectively.

Wang et al. presented a probability binary PSO (PBPSO) algorithm using a new strategy to obtain the new position [47]. In this algorithm, the following transfer function and rule are applied to determine the next binary position:

$$L\left(x_i^d(t+1)\right)=\frac{\left(x_i^d(t+1)-R_{\min}\right)}{(R_{\max}-R_{\min})}, \tag{14}$$

$$xb_i^d(t+1)=\begin{cases}1 & if \quad rand()\leq L\left(x_i^d(t+1)\right)\\ 0 & if \quad rand()>L\left(x_i^d(t+1)\right)\end{cases}, \tag{15}$$

where $x_i^d(t+1)$ is the next position in the continuous search space and *L(x)* is a linear function in (0,1). $[R_{\max}, R_{\min}]$ is a predefined range for the $L(x)$ function and $xb_i^d(t+1)$ is the next binary position.

They claimed that the computational complexity of BPSO reduced by this method, but the algorithm still traps in local optima when solving some optimization problems.

Nezamabadi-pour et al. introduced a new BPSO (NBPSO) to overcome the disadvantages of sigmoid function in BPSO [35] as follows:

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

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

where α is a constant to change the gradient of transfer function.

When the random number is less than the transfer function value, the next position is computed by changing the bits of current position from 0 to 1 or vice versa. Otherwise, the next position will be equal to the current position. The results revealed that NBPSO may get stuck into local optima due to the velocity of standard PSO. To solve this problem, an improved NBPSO (INBPSO) was introduced by Nezamabadi-pour et al. [35] as follows:

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

where A is a parameter to avoid the stagnation of the algorithm. When the algorithm falls into the local optimum, the *gbest* may not change during successive iterations. Therefore, the value of A increases so that the algorithm can get out of the local optimum.

An improved BPSO using Catfish effect (CatfishBPSO) was proposed by Chuang to improve the performance of BPSO [15]. The Catfish particles guide those particles that were trapped in local optima towards new search spaces to achieve better solutions. Mirjalili and Lewis introduced six new transfer functions which are divided into two categories, namely S-shaped and V-shaped transfer functions [33]. The performances of transfer functions were evaluated by the benchmark functions of CEC 2005 special session [13]. The best transfer function selected to be applied in some well-known versions of BPSO. The results showed that the following V-shaped transfer function (VBPSO8) has a better performance than the others on the tested functions.

$$S\left(v_i^d(t+1)\right)=\left|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}v_i^d(t+1)\right)\right|. \tag{19}$$

The next binary position is created based on the standard BPSO for S-shaped transfer functions. It is also generated based on NBPSO for V-shaped transfer functions. Although the V-shaped model shows a better performance in solving some problems, it may trap into local optima. If the best solution found by swarm is the local optimum, the second and third terms in (3) will be zero due to $pbest_i = gbest = x_i$. Also, the inertia weight is linearly decreased. Therefore, the next velocity becomes very near to zero and the next binary position will be the current binary position (the local optimum).

In another study, a memetic binary hybrid topology PSO (BHTPSO) was introduced by Beheshti et al. [3]. This algorithm combined local and global topologies to enhance exploration and exploitation in BPSO. In addition, a variant of BHTPSO, binary hybrid topology PSO quadratic interpolation (BHTPSO-QI), was proposed to improve the global searching ability. The algorithm applies (8) to update the particle's velocity. It also uses the following relations to compute the next position:

$$a_i^d(t+1)=v_i^d(t+1)+C(t)\times rand()\times\left(gbest^d(t)-xb_i^d(t)\right), \tag{20}$$

$$S\left(a_i^d(t+1)\right)=E+(1-E)\times\left|tanh\left(a_i^d(t+1)\right)\right|, \tag{21}$$

$$\begin{aligned} & if\ rand() < S\left(a_i^d(t+1)\right)\ then\ xb_i^d(t+1) = complement\ \left(xb_i^d(t)\right) \\ & \quad else\ xb_i^d(t+1) = xb_i^d(t),\ \ for\ i = 1, 2, ..., N. \end{aligned} \tag{22}$$

where C is a time-varying acceleration coefficient. E is obtained as follows:

$$E = erf\left(\frac{NF}{T}\right) = \frac{2}{\sqrt{\pi}} \int_0^{\frac{NF}{T}} e^{-t^2} dt, \tag{23}$$

where T is the maximum number of iterations and t is the current iteration. *erf* is an error function and *NF* is number of times failed to get better solution by the best particle.

In the proposed method, a new particle, $\tilde{X}$, is created by three different particles. If the fitness value of $\tilde{X}$is better than the *gbest*, $\tilde{X}$ will be the best solution (*gbest*). $\tilde{X}$ is generated as follows:

$$\tilde{X} = \left(xb_j^d\ \ xor\ \ xb_k^d\right)\ \ or\ \ \left(xb_k^d\ \ xor\ \ gbest^d\right)\ \ or\ \ \left(gbest^d\ \ xor\ \ xb_j^d\right),\quad j \neq k \neq gbest, \tag{24}$$

Moreover, several linear functions have been introduced for the binary search space. Bansal et al. proposed a binary version of PSO using a linear normalized transfer function [1]. The function and the next position were defined as follows:

$$L\left(x_i^d(t+1)\right) = \frac{\left(x_i^d(t+1) + v_i^d(t+1) + v_{\max}\right)}{1 + 2v_{\max}}, \tag{25}$$

$$xb_i^d(t+1) = \begin{cases} 1 & if \quad rand() < L\left(x_i^d(t+1)\right) \\ 0 & if \quad rand() \geq L\left(x_i^d(t+1)\right) \end{cases}. \tag{26}$$

Since this algorithm uses the velocity of PSO, it still suffers from the shortcoming of BPSO. A time-varying transfer function was introduced by Islam et al. [18]. The algorithm, namely TV_T-BPSO, was evaluated by combinatorial problems for low and high dimensions knapsack problems. The method applied the following transfer function and the next position is created based on S-shaped transfer functions:

$$S\left(v_i^d(t+1), \phi\right) = \frac{1}{1 + e^{-v_i^d(t+1)/\phi}}, \tag{27}$$

$$\phi = \phi_{max} - iter\left(\frac{\phi_{max} - \phi_{min}}{\max iter}\right), \tag{28}$$

where ϕ_{max} and ϕ_{min} are the control parameters of bound ϕ. *iter* is the current iteration and *maxiter* is the maximum number of iterations.

Although TV_T-BPSO has improved the balance between exploration and exploitation, it still faces the shortcomings of employing the sigmoid function as the base of transfer function. Kiran proposed the following relation in order to convert a continuous value to the binary value in the artificial bee colony (ABC) [25]:

$$xb_i^d(t+1) = round\left(\left|\ \ x_i^d(t+1) \mod 2\ \right|\right) \mod 2\ . \tag{29}$$

A new binary hybrid PSO with wavelet mutation (BHPSOWM) has been proposed by Jiang et al. [20]. In BHPSOWM, a mutation operator that is based on wavelet theory is applied in PSO to improve the quality of the best solution. The algorithm uses a sigmoid function to generate binary solutions; therefore, the algorithm encounters the mentioned disadvantages of S-shaped transfer functions.

Lin and Guan introduced a hybrid BPSO to solve the obnoxious *p*-median problem as an NP-hard problem [29]. The algorithm uses one of the three following relations to compute the new position. The selection of new position is based on a random number in the range [0, 1] and probabilities $prob_p$ and $prob_g$.Also, two tabu-based mutation operators and an iterated greedy local search are used to avoid the premature convergence and enhance exploitation. The $prob_p$ and $prob_g$ are set to constants less than one.

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

where $pbest_j^d(t)$ is the personal best position of the j^{th} particle ($j \neq i$). The particle j is randomly selected. Two operators, $\oplus$and ~, are defined as sum and difference operators, respectively.

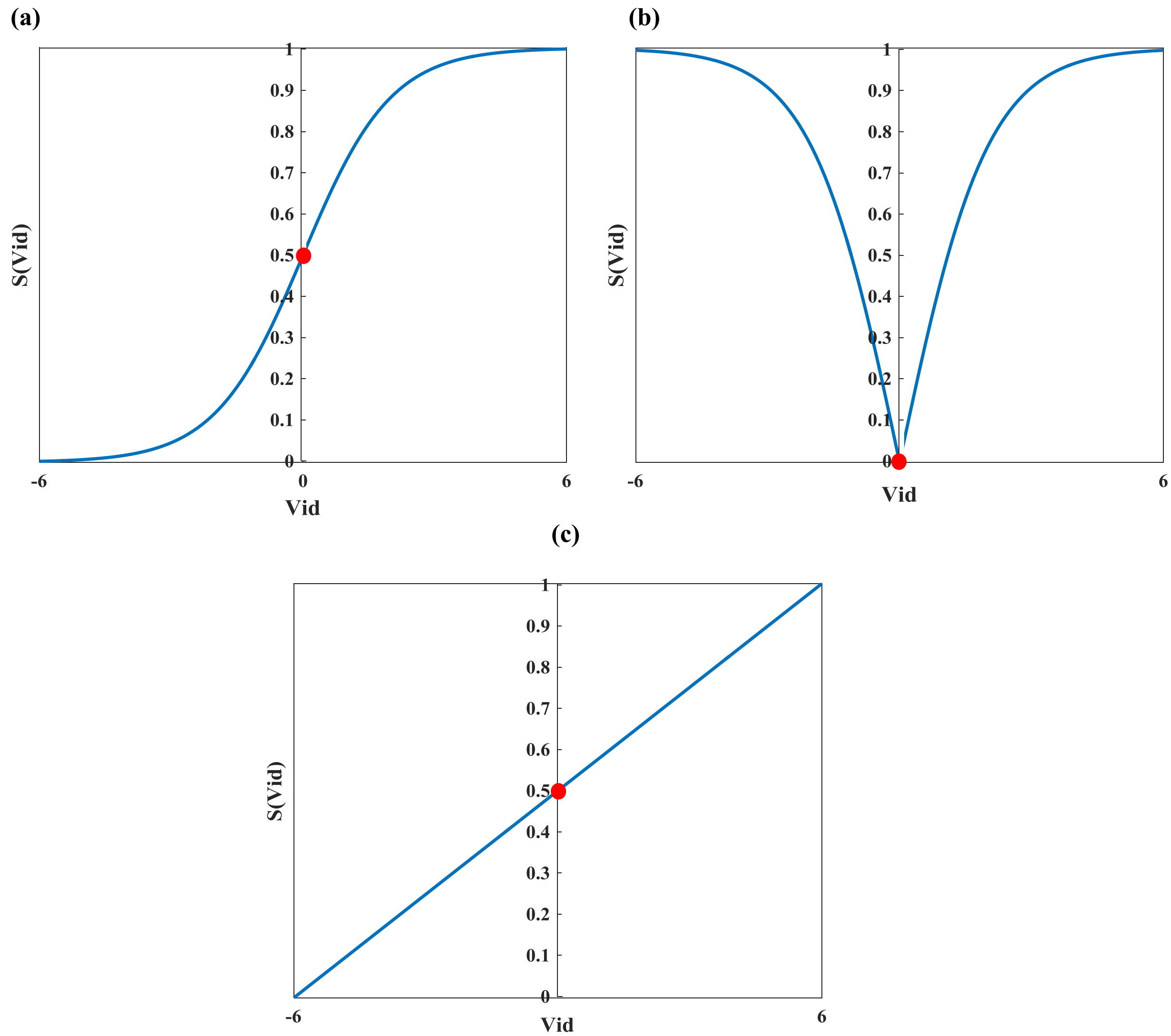


Fig. 1. (a) S-shaped, (b) V-Shaped and (c) Linear normalized transfer functions.

Jordehi introduced a new BPSO with a quadratic transfer function (QBPSO) for optimal scheduling of appliances in smart homes [21]. The algorithm obtains the new position based on a new transfer function as follows:

$$S\left(v_i^d(t+1)\right) = \begin{cases} \left(\dfrac{v_i^d(t)}{0.5v_{\max}^d}\right)^2 & if \quad v_i^d(t) < 0.5v_{\max}^d \\ 1 \quad if \quad v_i^d(t) \geq 0.5v_{\max}^d \end{cases} \tag{31}$$

$$xb_i^d(t+1) = Complement\left(xb_i^d(t)\right) \quad if \quad rand < \quad S\left(v_i^d(t+1)\right) \tag{32}$$

where $v_{\max}^d$is the maximum velocity in the d^{th} dimension.

4. TVMS-BPSO-The proposed method

The various BPSO algorithms with the S-shaped, V-shaped and linear transfer functions can sometimes find local optima or exhibit slow convergence speed [3,18,33,35]. Fig. 1 shows the general forms of S-shaped, V-Shaped and linear normalized transfer functions. Since, the velocity of PSO has poor exploration, this problem is led to a premature convergence rate. Therefore, a new transfer function should create a balance between exploration and exploitation to avoid local optima and to find the best solution.

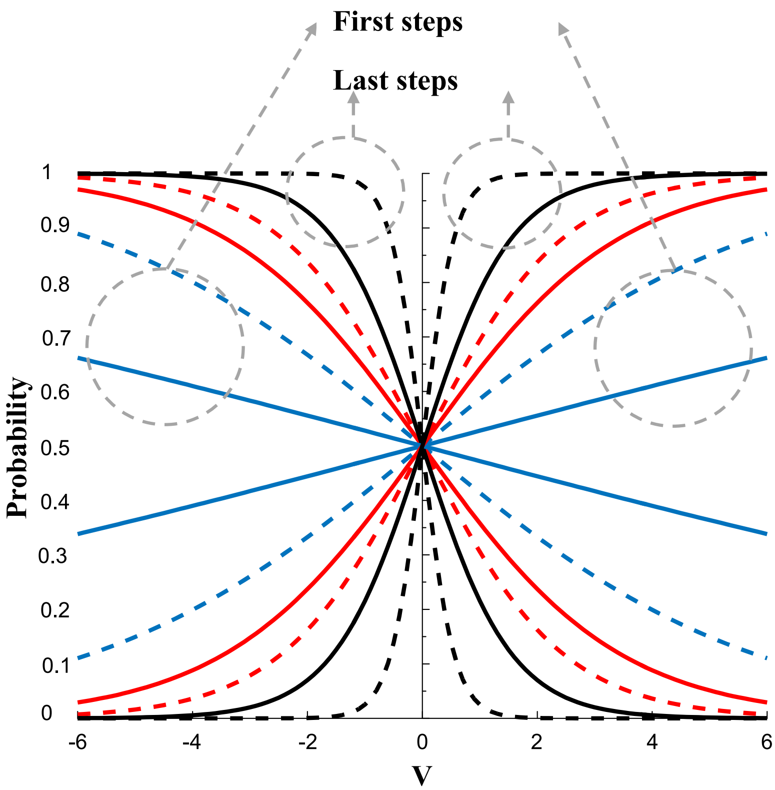


Fig. 2. The proposed transfer function with different values of control parameter σ.

In this section, a new time-varying mirrored sigmoid transfer function is introduced to enhance exploration and exploitation in BPSO. Fig. 2 shows the proposed transfer function. In the first steps, a strong exploration should be performed to avoid local optima. In the last steps, exploration should be switched to exploitation to search around good results. As observed in Fig. 2, exploration decreases from the first steps to the last steps and exploitation increases in the last repetitions.

The general structure of the proposed method has been shown in Fig. 3. As seen in this figure, two sigmoid functions are applied to convert the real results to the binary ones as follows:

$$S\left(v_i^d(t+1),\sigma\right) = \frac{1}{1+e^{\sigma\left(-v_i^d(t+1)\right)}}, \tag{33}$$

$$S'\left(v_i^d(t+1),\sigma\right) = \frac{1}{1+e^{\sigma\left(v_i^d(t+1)\right)}}, \tag{34}$$

where σ is a time-varying variable. It is initialized by σ_{max} and gradually decreased to $\sigma_{\min}$ in order to switch smoothly from exploration to exploitation. σ is defined as follows:

$$\sigma = (\sigma_{max} - \sigma_{\min})\left(\frac{iter}{\max iter}\right) + \sigma_{\min}. \tag{35}$$

The next binary positions of each transfer function are obtained by (36) and (37), respectively. Then, a greedy selection based on the objective function is done between P_i and P'_i as shown in (38). The best position is chosen as the next binary position $xb_i^d(t+1)$.

$$P_i^d(t+1) = \begin{cases} 1 & if \quad rand_1() < S\left(v_i^d(t+1),\sigma\right) \\ 0 & if \quad rand_1() \geq S\left(v_i^d(t+1),\sigma\right) \end{cases}, \tag{36}$$

$$P'^d_i(t+1) = \begin{cases} 1 & if \quad rand_2() > S'\left(v_i^d(t+1),\sigma\right) \\ 0 & if \quad rand_2() \leq S'\left(v_i^d(t+1),\sigma\right) \end{cases}, \tag{37}$$

$$xb_i(t+1) = \begin{cases} P_i(t+1) & if \quad f\left(P_i(t+1)\right) \quad is \quad better \quad than \quad f\left(P'_i(t+1)\right) \\ P'_i(t+1) & if \quad f\left(P'_i(t+1)\right) \quad is \quad better \quad than \quad f\left(P_i(t+1)\right) \end{cases}. \tag{38}$$

Algorithm 1. Pseudocode of the proposed TVMS-BPSO

Initialization phase
Initialize the control parameters: population size (N), C_1, C_2, σ_{max}, σ_{min}, $\sigma=\sigma_{min}$ (for first iteration) and certain stopping criteria.
Initialize the velocity of particles, $v_i^d=0$, $i=1..N$, $d=1..D$.
Initialize the position of particles randomly, xb_i^d, $i=1..N$, $d=1..D$, in the binary search spaces.
Evaluate all solutions.
$pbest_i^d=x_i^d, i=1..N, \quad d=1..D$.
Calculate the best solution ($gbest$).
Repeat
For each particle i Do
For each Dimension d Do
Calculate v_i^d using (3) for the global topology and (8) for local topology.
Compute $S\left(v_i^d(t+1),\sigma\right)$ using (33).
If $rand_1()<S\left(v_i^d(t+1),\sigma\right)$ ***then***
$P_i^d(t+1)=1$
else
$P_i^d(t+1)=0$
End If
Compute $S'\left(v_i^d(t+1),\sigma\right)$ using (34).
If $rand_2()>S'\left(v_i^d(t+1),\sigma\right)$ ***then***
$P_i'^d(t+1)=1$
else
$P_i'^d(t+1)=0$
End If
End For d
Compute the next position $xb_i(t+1)$:
If $f(P_i(t+1))$ is better than $f(P'_i(t+1))$ ***then***
$xb_i(t+1)=P_i(t+1)$
else
$xb_i(t+1)=P'_i(t+1)$
End If
End For i
Calculate the best solution ($gbest$).
Compute σ using (35).
UNTIL ***certain stopping criterion is met***
Return gbest

Fig. 3. The pseudocode of the proposed TVMS-BPSO.

To show how TVMS-BPSO algorithm achieves the best solution, all steps of the algorithm are described by an example as seen in Table 1. The Max-Ones function is selected for this purpose. This is a binary function to be maximized. The maximum value of the function depends on its dimension. For example, if the dimension is equal to 5, the best result will be 5. The function is defined as follows:

$$f(x)=\sum_{i=1}^{D}x_i \tag{39}$$

In this example, the population size and the maximum iteration are set to 4 and 3, respectively. The dimension is set to 5. Therefore, the velocity and the position of each particle are vectors with 5 dimensions.

Initialization step: Particles' velocities are set to zero and the particles' positions are randomly initialized. The best solution ($gbest$) is computed, based on the fitness values of initialized particles (The best fitness value=2).

Table 1
An example of solving Max-Ones function by TVSM-BPSO ($N = 4$, $D = 5$ and *Maximum iteration*=3).

Iteration	σ	Particle No	Particle's Velocity (V)					P					P'					Particle's Position (X)					Global best
Initialization	–	Particle #1	0	0	0	0	0	–					–					0	0	1	0	0	2
		Particle #2	0	0	0	0	0											1	0	0	0	0	
		Particle #3	0	0	0	0	0											0	1	0	0	0	
		Particle #4	0	0	0	0	0											0	0	0	1	1	
1	0.1	Particle #1	0	0	−0.219	1.1	1.415	1	0	0	1	1	0	1	0	0	1	1	0	0	1	1	3
		Particle #2	−0.781	0	0	0.867	1.024	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	
		Particle #3	0	−0.02	0	0.097	1.004	0	1	1	0	1	1	0	0	0	1	0	1	1	0	1	
		Particle #4	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	0	0	1	1	
2	0.55	Particle #1	0	0	−0.219	1.1	1.415	1	0	1	0	1	0	0	1	1	1	0	0	1	1	1	5
		Particle #2	0.203	−3.091	0	2.022	2.523	1	0	0	1	1	0	0	1	0	1	1	0	0	1	1	
		Particle #3	1.707	−1.805	−0.407	0.938	1.004	1	0	0	1	0	1	0	1	1	1	1	0	1	1	1	
		Particle #4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	**1**	**1**	**1**	**1**	**1**	
3	1	Particle #1	1.382	0.159	−1.156	1.1	1.415	0	1	0	1	1	1	1	0	0	1	1	1	0	0	1	5
		Particle #2	0.203	−2.20	1.354	2.022	2.523	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0	
		Particle #3	1.707	−0.430	−0.407	0.938	1.004	1	1	1	1	1	1	0	0	1	1	**1**	**1**	**1**	**1**	**1**	
		Particle #4	0	0	0	0	0	1	0	0	1	0	0	0	1	1	1	0	0	1	1	1	

Table 2
The parameter settings of tested algorithms.

BPSO, LBPSO, BPSO-bin, LBPSO-bin, INBPSO, LINBPSO, VBPSO8, LVBPSO8	MS-BPSO, MS-LBPSO	TV-BPSO, TV-LBPSO	TVMS-BPSO, TVMS-LBPSO, TVMS-VLBPSO	HTBPSO-QI	BBA	BGSA	GB-ABC
$w_{\max} = 0.9$ $w_{\min} = 0.4$ $C_1 = C_2 = 2$	$w = 1$ $C_1 = C_2 = 2$	$w = 1$ $C_1 = C_2 = 2$ $\phi_{max} = 5$ $\phi_{min} = 1$ $V_{\max} = 10$	$w = 1$ $C_1 = C_2 = 2$ $\sigma_{max} = 1$ $\sigma_{\min} = 0.1$ $V_{\max} = 10$	$w_{\max} = 0.6$ $w_{\min} = 0.2$ $C_{1,\max} = 2$ $C_{1,\min} = 0.5$ $C_{2,\max} = 2$ $C_{2,\min} = 1$ $C_{3,\max} = 1.5$ $C_{3,\min} = 0.5$	$F_{\max} = 2$ $F_{\min} = 0$ $A = 0.25$ $r = 0.5$ $\varepsilon =$ $[-1, 1]\gamma =$ 0.9 $a = 0.9$	$G_0 =$ $100k_{0\,\max} =$ N $k_{0\,\min} = 1$	$Th = 0.1D$

Iteration #1 ***(**$\sigma = 0.1$**):*** The new velocity is computed according to (3) for all particles. *Pi* and *Pi'* are obtained by (36) and (37), respectively. The best position between *Pi* and *Pi'* is selected as the next position by (38). The new best solution (*gbest*) is calculated based on these new particles' positions (The best fitness value=3).

Iteration #2 ($\sigma = 0.55$)***:*** The next particles' velocities and particles' positions are computed for the swarm. The best position is [1 1 1 1 1] and the best fitness value is 5. The particle #4 has achieved the best solution.

Iteration #3 ($\sigma = 1$)***:*** The algorithm is repeated until the stopping criterion (the *maximum number of iterations*) is met.

In the problem, the proposed method achieves the global optimum very fast. In the next section, the results of TVMS-BPSO are compared with state-of-the-art BPSO algorithms and some binary algorithms as well.

5. Experimental results and discussion

The proposed TVMS-BPSO algorithm with local and global topologies is compared with various BPSOs on CEC 2005 benchmark functions. In addition, the best algorithms are selected in this step to evaluate their performances with some well-known binary swarm intelligence algorithms on the 0–1 MKP benchmark instances. These results are presented in Sections 5.2 and 5.3.

As mentioned, the local topology shows a better efficiency compared with the global topology in many problems; thus, some BPSO algorithms with the local topology are implemented in Section 5.2. The performances of four S-shaped and four V-shaped transfer functions have been compared with each other on CEC 2005 benchmark functions by Mirjalili and lewis [33]. Among them, VBPSO8 (19) showed the best results; therefore, VBPSO8 and the local topology VBPSO8 (LVBPSO8) are selected in the experiment. Also, the proposed transfer function with a fixed value $\sigma(\sigma = 1)$ is applied in BPSO (MS-BPSO) and LBPSO (MS-LBPSO) so that the performance of the time-varying transfer function is cleared in this study. TV-BPSO [18], TV-LBPSO, INBPSO [35] and LINBPSO are chosen for the comparison. As described in Section 3, a new method (29) has been proposed by Kiran [25] to convert the continuous search space to the discrete one in the binary ABC. The method is also applied in the BPSO (BPSO-bin) and LBPSO (LBPSO-bin).

In Section 5.3, some well-known binary swarm intelligence algorithms have been chosen to solve 0–1 MKP benchmark instances such as binary gravitational search algorithm (BGSA) [39], binary hybrid topology particle swarm optimization quadratic interpolation (BHTPSO-QI) [3], Binary bat algorithm (BBA) [34] and artificial bee colony algorithm with genetic operators (GB-ABC) [36]. The best algorithms, based on their results from Section 5.2, have been selected for Section 5.3. Moreover, a variable neighbors BPSO with time-varying mirrored S-shaped transfer function (TVMS-VLBPSO) is applied to evaluate the proposed transfer function. The results of TVMS-VLBPSO and TVMS-LBPSO are compared with each other to show the role of local topology in the performance of BPSO. In TVMS-VLBPSO, the neighbors of each particle are changed per iteration. Each particle x_i finds a new solution based on the best neighbor from its near neighborhood in the algorithm. The information of near neighborhood is calculated by the Hamming distance (HD) between x_i and near neighbors. A near neighbor is defined as follows:

$$\begin{array}{l} if \quad HD_{ik} \le MeanHD_i \quad then \quad k \quad is \quad a \quad near \quad neighbor \quad of \quad i, \\ \qquad else \quad k \quad is \quad not \quad a \quad near \quad neighbor \quad of \quad i \end{array} \tag{40}$$

where $MeanHD_i$ is the average Hamming distance between i and other particles.

The distance between two neighbors i and k is computed based on Hamming distance as follows:

$$\begin{array}{l} HD_{ik} = \sum_{j=1}^{D} (x_{ij} - x_{kj}) \\ i, k \in \{1, 2, ..., N\}, \qquad j \in \{1, 2, ..., D\}. \end{array} \tag{41}$$

Table 3
The best solution achieved by TVMS-BPSO on some CEC 2005 functions with different values of w, C_1 and C_2.

Function No.	$W=0.9$–0.4, $C1=C2=2$	$W=1$, $C1=C2=2$	$W=0.9$–0.4, $C1=2.5$–0.5, $C2=0.5$–2.5	$K=0.729$, $C1=C2=2.05$
F1	−404.4383	**−446.97**	−387.2141	−176.7926
F2	−406.0657	**−438.98**	−384.3594	−181.5523
F3	256245.5093	**98287**	288093.6993	831363.379
F4	−379.9042	**−441.77**	−359.8661	−47.8106
F5	289.0189	**−241.97**	427.1282	1142.6469
F6	26608.7554	**503.45**	63492.8493	932100.5708
F7	350.2803	**266.25**	279.2058	389.5721
F8	−119.9698	**−120.05**	−119.9816	−119.8995
F9	−321.4709	**−328.55**	−321.3894	−317.0921
F10	−316.6379	**−325.6**	−317.115	−314.7437
F11	92.6191	**91.167**	92.745	93.1742
F12	−6.3943	**−238.13**	126.1142	1121.0191
F13	−129.0283	**−129.77**	−129.0012	−128.8447
F14	−298.5191	**−298.87**	−298.5324	−298.4165
F15	650.8381	**379.16**	644.2256	734.3625
F16	392.4665	**261.86**	412.626	414.8928
F17	424.8283	**276.62**	431.7592	456.6358
F18	1040.9231	**795.26**	1039.9108	1086.6817
F19	1024.2713	**737.95**	1054.2098	1094.2269
F20	1003.6173	**748.04**	1029.9527	1083.2416
F21	1559.2825	**1183.3**	1571.0258	1647.908
F22	1253.4124	**1138**	1268.0848	1345.2272
F23	1569.3373	**1271.1**	1590.0504	1655.2908
F24	1105.8885	**526.23**	1183.8923	1429.9303
F25	2107.0258	**2007.8**	2119.8243	2140.6466

Table 4
The results of functions achieved by TVMS-BPSO for different values of σ .

Function No. σ	F8	F8	F10	F10	F25	F25
0.01	−119.9557	−119.9390	−309.9281	−312.8649	2159.1821	2172.1226
0.02	−120.6322	−119.9548	−315.6055	−314.0571	2166.9891	2179.6723
0.03	−121.6686	−119.9175	−316.4295	−318.7756	2142.8119	2176.6815
0.04	−122.4442	−119.9098	−311.9856	−317.1489	2122.0798	2107.7887
0.05	−119.9708	−119.9262	−314.0304	−320.9332	2116.3920	2134.6600
0.06	−119.9735	−119.9586	−315.1207	−318.6501	2093.4844	2126.6118
0.07	−119.8819	−119.9104	−320.7575	−317.5448	2115.8212	2108.3428
0.08	−119.9807	−119.9557	−319.3405	−323.2278	2100.6859	2104.4754
0.09	−122.4001	−119.8965	−316.5014	−317.4313	2093.9501	2090.0655
0.1	−119.9752	**−121.6934**	−317.8969	−317.6542	2083.7640	2078.5430
0.14	**−128.9610**	−119.9755	−326.4695	−325.1045	2069.1585	2059.2847
0.2	−119.9562	−119.9559	−327.8170	−327.9123	2027.8582	2035.3040
0.3	−119.9527	−119.9627	−318.9417	−325.8113	1996.9452	2002.4808
0.4	−119.9815	−119.9614	−322.2314	−325.7731	1988.2775	1983.3925
0.5	−119.9913	−119.9522	−319.9014	−327.3937	1984.4718	1975.3279
0.6	−119.8646	−119.8763	**−328.9960**	−317.0760	1990.9568	1980.8515
0.7	−119.8478	−119.9647	−320.5213	−319.0555	1990.3480	1980.7978
0.8	−119.8825	−119.9015	−325.8652	−320.3386	1980.3532	1975.2823
0.9	−119.9391	−119.8992	−316.6491	−319.5188	1976.3472	1985.2341
0.91	−119.8996	−119.8805	−305.3470	−316.0833	1981.8810	**1970.8766**
0.98	−119.9507	−119.8229	−325.3033	−320.2227	**1974.0571**	1973.7776
1	−119.9258	−119.8907	−310.7832	**−328.9983**	1981.1081	1979.0480
1.1	−119.9578	−119.7886	−311.7171	−326.9786	1977.1598	1980.1110
1.2	−119.8987	−119.9343	−320.5557	−327.1565	1980.4103	1976.5116
1.3	−119.9942	−119.9040	−328.9916	−327.3621	1987.1147	1974.6740
1.4	−119.8488	−119.5705	−317.6549	−326.2246	1976.8226	1977.6369
1.5	−119.9663	−119.6678	−303.8462	−326.8653	1984.0809	1978.0868

5.1. Parameter settings

All parameters of algorithms applied in this study are based on their references as shown in Table 2. In local topology algorithms (except TVMS-VLBPSO), a ring topology is used as the neighbourhood structure and the number of neighbours is 2. In the proposed methods, different values of w, C_1 and C_2 [16-18,40] have been tested on CEC2005 benchmark functions as shown in Table 3. The best value for these parameters are $w=1$, $C_1=C_2=2$. The values of σ_{max} and $\sigma_{\min}$ for the proposed transfer function were tested by some benchmark functions. The best values for σ_{max} and $\sigma_{\min}$ are 1 and 0.1, respectively

Table 5
The mean and standard deviation (±SD) of the best solution for the CEC 2005 benchmark functions.

Algorithms	F1	F2	F3	F4	F5
Global Topology					
TVMS-BPSO	−446.97±5.4981	−438.98±17.377	98287±1.13e+05	−441.77±12.37	−241.97±106.14
MS-BPSO	−420.73±60.906	−395.92±181.1	3.4725e+05±3.684e+05	−406.59±56.808	225.78±694.32
TV-BPSO	−446.1 ± 5.9235	−436.31±22.081	1.0534e+05±1.11e+05	−433.73±25.236	−127.65±270.05
BPSO	−432.56±16.237	−421.95±24.169	2.9776e+05±2.587e+05	−409.16±31.834	−4.5091±106.95
PSO-bin	−421.48±53.882	−429.29±23.722	1.7321e+05±1.891e+05	−417.53±44.676	−28.15±230.95
INBPSO	−388.44±111.09	−343.49±260.67	4.2293e+05±9.474e+05	−257.83±405.01	314.78±777.11
VBPSO8	−440.99±15.999	−434.54±22.582	1.5006e+05±1.417e+05	−432.74±20.972	−140.78±142.44
Local Topology					
TVMS-LBPSO	**−449.95±0.08195**	**−448.58±1.8448**	**42613±34193**	**−447.9 ± 2.9061**	**−300.46±16.03**
MS-LBPSO	−449.93±0.10321	−442.39±12.973	53740±45736	−443.93±6.1872	−240.49±86.221
TV-LBPSO	−449.88±0.11444	−446.16±4.1313	46468±47603	−445.95±6.322	−271.68±47.705
LBPSO	−420.04±18.903	−411.31±18.258	2.7076e+05±1.655e+05	−391.81±25.165	151.69±143.66
LPSO-bin	−416.56±24.474	−409.04±26.071	1.9264e+05±1.08e+05	−388.8 ± 44.287	226.64±223.99
LINBPSO	−449.39±1.2979	−435.02±20.452	1.1082e+05±1.016e+05	−437.05±12.252	−161.17±168.01
LVBPSO8	−447.98±2.8167	−440.41±9.6179	85863±62192	−436.29±11.742	−147.08±110.47

Algorithms	F6	F7	F8	F9	F10
Global Topology					
TVMS-BPSO	503.45±146.36	266.25±265.91	−120.05±0.74112	−328.55±1.3196	−325.6 ± 1.9871
MS-BPSO	10293±22647	266.66±265.95	−120.01±0.31108	−328.03±1.3623	−322.63±4.0479
TV-BPSO	1625±7752.8	266.43±266.04	−120.24±1.7162	−328.54±1.1191	−325.14±2.6135
BPSO	4926.8 ± 9628.2	269.87±267.37	−120.05±0.62672	−323.75±2.2942	−320±3.6903
PSO-bin	10056±30363	267.48±266.49	−119.91±0.042468	−326.1 ± 1.8146	−322.1 ± 3.6177
INBPSO	4.6038e+05±2.146e+06	266.75±265.92	−120.02±0.64313	−325.78±2.7139	−321.43±4.6189
VBPSO8	3338.2 ± 13255	266.54±266	−120.23±1.6075	−327.56±1.5012	−323.15±3.0142
Local Topology					
TVMS-LBPSO	**402.95±24.191**	**266.09±266**	**−120.25±1.0603**	**−329.15±0.64132**	**−326.85±1.2275**
MS-LBPSO	427.8 ± 57.128	266.11±266.01	−119.96±0.022946	−329±0.87708	−325.83±1.8089
TV-LBPSO	412.89±31.738	266.12±265.99	−119.95±0.16655	−328.56±0.79702	−326.51±1.198
LBPSO	16998±19042	272.98±268.97	−120±0.43143	−322.88±2.0375	−318.28±2.5028
LPSO-bin	24998±31607	273.56±268.62	−119.91±0.037455	−323.59±1.858	−318.07±3.2552
LINBPSO	528.09±129.12	266.18±266.01	−119.99±0.53181	−327.29±1.5843	−323.9 ± 2.3508
LVBPSO8	886.16±936.86	266.3 ± 266	−120±0.66175	−327.05±1.5297	−323.22±2.6866

Algorithms	F11	F12	F13	F14	F15
Global Topology					
TVMS-BPSO	91.167±0.65567	−238.13±222.15	−129.77±0.095197	−298.87±0.28931	379.16±154.18
MS-BPSO	91.698±0.70823	−64.804±457.07	−129.66±0.21671	−298.75±0.35812	388.99±156.18
TV-BPSO	91.293±0.67296	−224.79±277.22	−129.78±0.10462	−298.86±0.31671	406.85±157.16
BPSO	92.447±0.44508	−98.765±237.39	−129.3 ± 0.22568	−298.57±0.17735	639.77±101.61
PSO-bin	92.01±0.70265	−158.95±363.95	−129.44±0.24844	−298.66±0.18228	459.67±132.06
INBPSO	91.831±0.70665	144.94±663.46	−129.61±0.28131	−298.65±0.33267	470.43±191.29
VBPSO8	91.612±0.63273	−292.89±110.17	−129.69±0.13986	−298.78±0.24185	348.82±109.29
Local Topology					
TVMS-LBPSO	**91.055±0.33737**	**−404.88±66.753**	**−129.85±0.068733**	**−299±0.26351**	**281.87±77.521**
MS-LBPSO	91.309±0.41102	−322.43±90.273	−129.81±0.07994	−298.88±0.26877	286.21±40.789
TV-LBPSO	91.258±0.4221	−380.63±82.042	−129.83±0.076963	−298.93±0.23234	297.77±75.577
LBPSO	92.73±0.41745	−93.845±171.56	−129.23±0.21253	−298.51±0.13855	629.42±64.395
LPSO-bin	92.335±0.4855	−134.86±178	−129.34±0.2235	−298.53±0.17661	521.06±75.332
LINBPSO	91.949±0.50738	−237.62±147.33	−129.69±0.12899	−298.73±0.25447	347.63±70.853
LVBPSO8	91.871±0.36564	−294.22±105.33	−129.66±0.12615	−298.72±0.22373	369.64±56.591

Algorithms	F16	F17	F18	F19	F20
Global topology					
TVMS-BPSO	261.86±17.854	276.62±24.911	795.26±208.3	737.95±220.91	748.04±203.51
MS-BPSO	280.98±33.481	278.69±27.625	911.25±136.76	875.01±164.41	866.8 ± 155.9
TV-BPSO	267±18.718	270.22±22.168	776.61±207.36	782.17±195.41	812.13±193.14
BPSO	373.48±23.068	399.26±23.941	976.16±92.887	965.66±106.24	962.82±83.981
PSO-bin	336.12±26.462	376.83±35.905	920.38±123.51	914.27±113.77	868.68±161.03
INBPSO	291.4 ± 32.902	300.78±26.573	903.6 ± 150.76	923.5 ± 137.33	917.09±124.39
VBPSO8	281.99±24.545	295.43±29.32	862.6 ± 158.3	756.03±204.36	777.48±214.19
Local Topology					
TVMS-LBPSO	**248.25±11.188**	**269.86±13.777**	**651.28±152.91**	674.26±164.1	655.55±151.15
MS-LBPSO	257.19±13.155	273.85±18.529	792.61±120.16	789.19±151.57	774.7 ± 153.95
TV-LBPSO	255.49±11.815	277.41±15.24	662.65±151.91	**656.18±152.3**	**641.04±167.54**
LBPSO	373.03±17.933	401.36±25.773	994.02±53.698	980.67±64.691	1001.2 ± 52.476
LPSO-bin	369.64±17.129	390.6 ± 24.902	986.3 ± 89.64	973.7 ± 83.393	999.8 ± 66.473
LINBPSO	297.93±20.526	314.41±21.132	860.19±111.6	890.73±84.47	867.31±131.58
LVBPSO8	315.27±17.965	318.44±18.385	867.79±80.257	845.89±103.67	858.99±114.48

Algorithms	F21	F22	F23	F24	F25
Global Topology					
TVMS-BPSO	1183.3 ± 282.89	1138±26.57	1271.1 ± 261.39	526.23±91.107	2007.8 ± 8.0833
MS-BPSO	1324.9 ± 256.63	1184.8 ± 54.054	1354.2 ± 253.33	624.07±127.68	2010.7 ± 10.665
TV-BPSO	1124.2 ± 272.74	1151.2 ± 41.59	1276.2 ± 247.75	595.67±187.08	2012.5 ± 8.7731
BPSO	1508±149.16	1224.9 ± 32.686	1516.5 ± 143.39	951.93±143.85	2067.8 ± 13.635
PSO-bin	1301.5 ± 236.36	1209.4 ± 49.421	1373.8 ± 238.43	714.97±192.4	2046.8 ± 13.484
INBPSO	1430.3 ± 208.56	1200.2 ± 62.768	1429.9 ± 220.11	873.01±326.81	2018.1 ± 13.646
VBPSO8	1238.1 ± 259.79	1158.3 ± 52.273	1275.6 ± 269.84	649.59±210.43	2023.2 ± 8.2016
Local Topology					
TVMS-LBPSO	**960.08±146.28**	**1129.4 ± 44.566**	**1018.1 ± 164.44**	**468.31±10.731**	**2005.9 ± 5.7765**
MS-LBPSO	1137.7 ± 220.12	1134.3 ± 60.423	1091.3 ± 223.78	475.77±35.753	2009.9 ± 6.713
TV-LBPSO	1071.5 ± 176.39	1136.4 ± 47.679	1048.5 ± 205.9	478.5 ± 31.45	2010.5 ± 4.8097
LBPSO	1504.3 ± 100.85	1243.5 ± 24.371	1508.2 ± 106.95	1065.6 ± 93.717	2077.6 ± 11.085
LPSO-bin	1492.1 ± 130.63	1231.3 ± 27.627	1480.1 ± 118.58	1035.7 ± 139.9	2067±12.925
LINBPSO	1204.7 ± 231.74	1157.4 ± 71.476	1288.9 ± 222.52	533.83±78.492	2011.4 ± 6.5512
LVBPSO8	1285.1 ± 204.65	1171.3 ± 15.948	1287.2 ± 180.37	609.03±91.381	2028.5 ± 6.2044

Table 6
The results of Friedman test on CEC 2005 benchmark functions.

Algorithm	Global Topology							Local Topology						
Function	TVMS-BPSO	MS-BPSO	TV-BPSO	BPSO	PSO-bin	INBPSO	VBPSO8	TVMS-LBPSO	MS-LBPSO	TV-LBPSO	LBPSO	LPSO-bin	LINBPSO	LVBPSO8
F1	5.92	9.04	6.4	10.88	9.84	10.34	7.65	2.22	2.48	3.71	12.06	12.16	5.22	7.08
F2	5.52	8.3	6.26	9.94	8.5	10.26	6.8	2.52	5.34	4.32	11.58	11.4	7.16	7.1
F3	6.24	9.86	6.22	10.74	8.08	9.5	7.98	3.92	4.64	3.8	10.9	9.8	7.18	6.14
F4	5.28	8.94	5.9	10.44	8.42	9.44	6.72	2.88	5	4	11.78	11.72	7.04	7.44
F5	4.58	9.27	6.32	10.06	9.06	9.78	7.48	2.07	4.72	3.82	11.86	12.16	6.46	7.36
F6	5.32	7.28	5.8	11.52	9.3	8.92	7.14	2.28	4.18	3.28	12.78	12.52	6.58	8.1
F7	5.67	7.41	6.24	12.08	10.48	8.22	7.53	2.55	2.67	3.64	13.34	13.28	4.67	7.22
F8	5.16	5.84	6.78	8.64	9.54	7.62	9.66	4.4	4.14	7.96	7.8	9.34	8.78	9.34
F9	4.46	5.8	4.77	11.86	9.32	9.26	6.98	3.11	3.62	4.76	12.78	12.4	7.68	8.2
F10	4.88	8.04	5.04	10.76	8.92	8.96	7.88	3.12	4.62	3.7	12.34	12.08	7.08	7.58
F11	4.32	7.2	5	11.16	9.1	8.06	6.82	3.5	4.74	4.58	12.64	10.8	8.76	8.32
F12	6.94	8.62	6.86	9.8	8.6	10.42	6.34	3.14	5.84	4.04	10.68	9.68	7.64	6.4
F13	5.46	7.08	4.96	12	10.7	7.62	7	3.02	4.04	3.54	12.58	11.7	7.44	7.86
F14	5.4	7.9	5.74	10.44	8.64	8.92	6.5	3.54	5.3	4.62	11.26	11.22	7.58	7.94
F15	6.08	6.78	6.88	12.34	9.22	8.74	5.98	3.46	4.26	4.38	12.62	10.34	6.46	7.46
F16	4.24	6.12	5.02	12.86	10.62	7.1	6.48	2.42	3.6	3.62	12.92	12.68	7.92	9.4
F17	4.4	4.54	3.84	12.7	11.72	6.66	6.26	3.66	4.32	4.48	12.86	12.42	8.42	8.72
F18	6.4	8.96	5.8	10.76	9.06	8.96	7.68	2.96	4.9	3.12	11.4	11.14	7	6.86
F19	5.34	8.4	5.82	10.74	9.06	9.76	5.94	3.66	6.02	3.28	11.02	11.14	7.8	7.02
F20	5.18	7.84	6.6	10.48	8.26	9.26	6.52	3.24	5.44	3.36	11.8	11.84	7.84	7.34
F21	6.14	8.26	5.14	11.6	7.86	10.04	7.04	2.9	5.14	4.56	11.06	11.12	6.76	7.38
F22	3.54	7.76	4.32	11.44	10.32	9.08	6.54	3.2	4.68	4.3	12.62	12.04	7.14	8.02
F23	6.83	8.46	7.06	11.38	8.88	9.5	7.32	3.18	4.14	3.81	10.62	9.92	7.08	6.82
F24	4.88	7.54	5.92	12	9.12	10.36	7.7	2.08	2.74	3.42	12.86	12.58	6	7.8
F25	4.04	5.18	5.38	12.72	10.94	6.48	8.2	3.06	4.24	4.6	13.36	12.56	4.84	9.4
Sum	132.22	190.42	144.07	279.34	233.56	223.26	178.14	**76.09**	110.81	102.7	297.52	288.04	176.53	192.3

as shown in Table 4. In this table, *F8, F10* and *F25* have been run twice and the best results have been shown in bold. As seen in this table, the best solutions have been achieved by σ in the range [0.1, 1].

The other parameter settings are as follows: the population size (N) is set to 40 for CEC 2005 functions and 100 for 0–1 MKP benchmark instances [3]. The maximum number of iterations as the stopping criterion is set to 500 [33] for CEC 2005 functions. The results of 0–1 MKP benchmark instances are obtained for the maximum number of iterations 3000 [3] and 5000.

5.2. Results and discussion of nonlinear benchmark functions

Twenty five CEC 2005 benchmark functions are selected in this study. They are divided into four categories [43]: shifted unimodal (*F1-F5*), shifted and rotated multimodal (*F6-F12*), expanded (*F13 and F14*) and hybrid composition (*F15-F25*) functions.

The dimensions of functions *F1-F14* are set to 5 [33]. For hybrid composition functions *F15–F25*, the dimension is set to 10 [33]. In these test functions, 15 bits are considered to represent each continuous value [33] and the particle dimension is computed as follows:

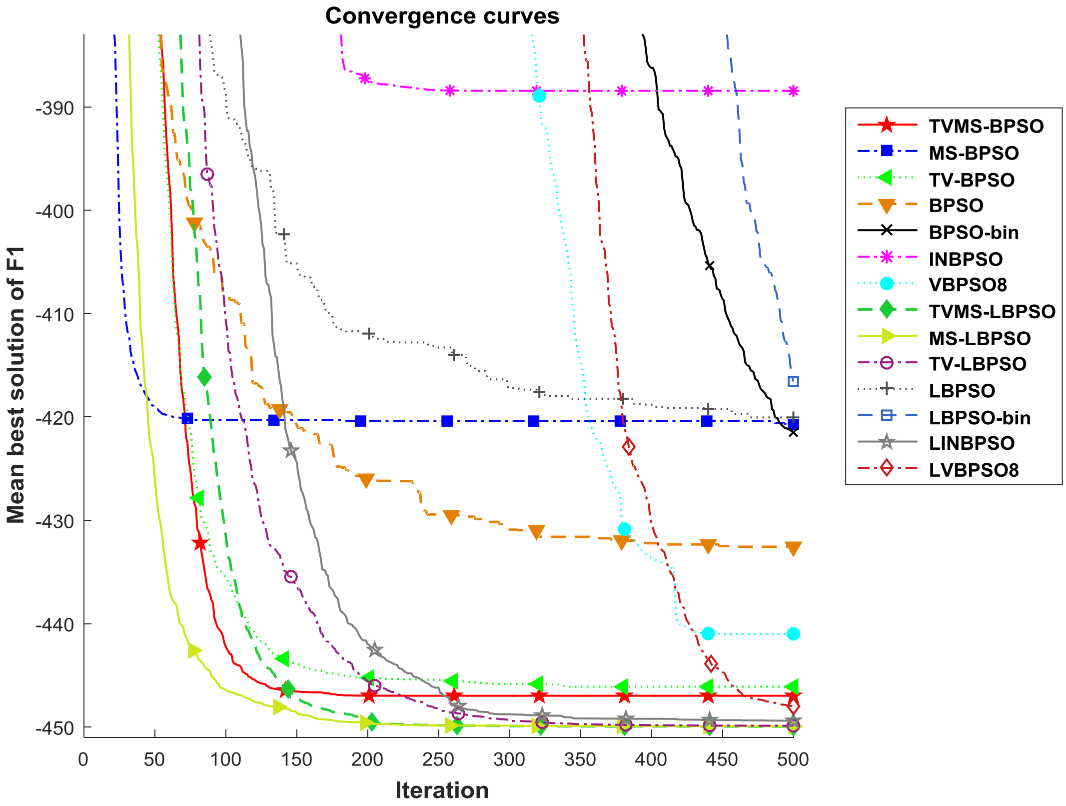
$$D = Dimension_{function} \times 15 \tag{42}$$

All binary algorithms are randomly initialized and run 50 times on minimum benchmark functions. The best results achieved by algorithms and the standard deviation (SD) of the best solution in the last iteration are shown in Table 5. In this table, algorithms are divided into two groups: global topology and local topology. Seven algorithms, TVMS-BPSO, MS-BPSO, TV-BPSO, BPSO, PSO-bin, INBPSO and VBPSO8, have been implemented, based on the global topology. The others, TVMS-LBPSO, MS-LBPSO, TV-LBPSO, LBPSO, LPSO-bin, LINBPSO and LVBPSO8, are based on the local topology. The best results in the group of local and global topologies have been separately shown in underline. The best results of all algorithms have been demonstrated in bold.

From Table 5, it can be concluded that the proposed transfer function has superior performance in both local and global topologies. It shows that a good transfer function can considerably improve the efficiency of BPSO. For the first group, unimodal functions, *F1-F5*, TVMS-LBPSO, provides the best solution among all algorithms. Also, TVMS-BPSO shows better results in the global topology. Although INBPSO shows the worst results for the global topology in this group, LINBPSO performs better than INBPSO for the local topology.

As observed in Table 5, TVMS-LBPSO performs superior on the second group of benchmark functions (*F6-F12)*. Among the global topology algorithms, TVMS-BPSO provides the best results except for two functions *F8* and *F12*. In *F8*, TV-BPSO shows the best solution and VBPSO8 generates the best results for *F12*.

In group 3 (*F13* and *F14*), the results of all algorithms are near each other. These functions are expanded functions.The binary algorithms return good results close to the best global optimum. The best results for these functions are achieved by TVMS-LBPSO.



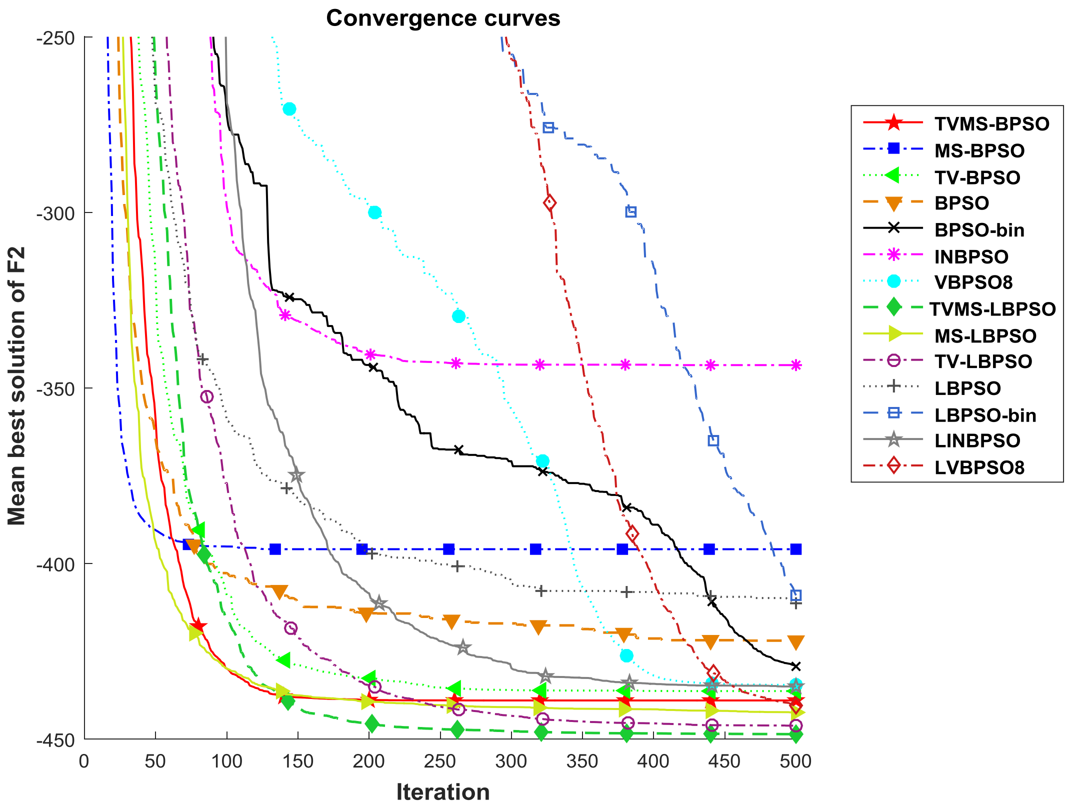
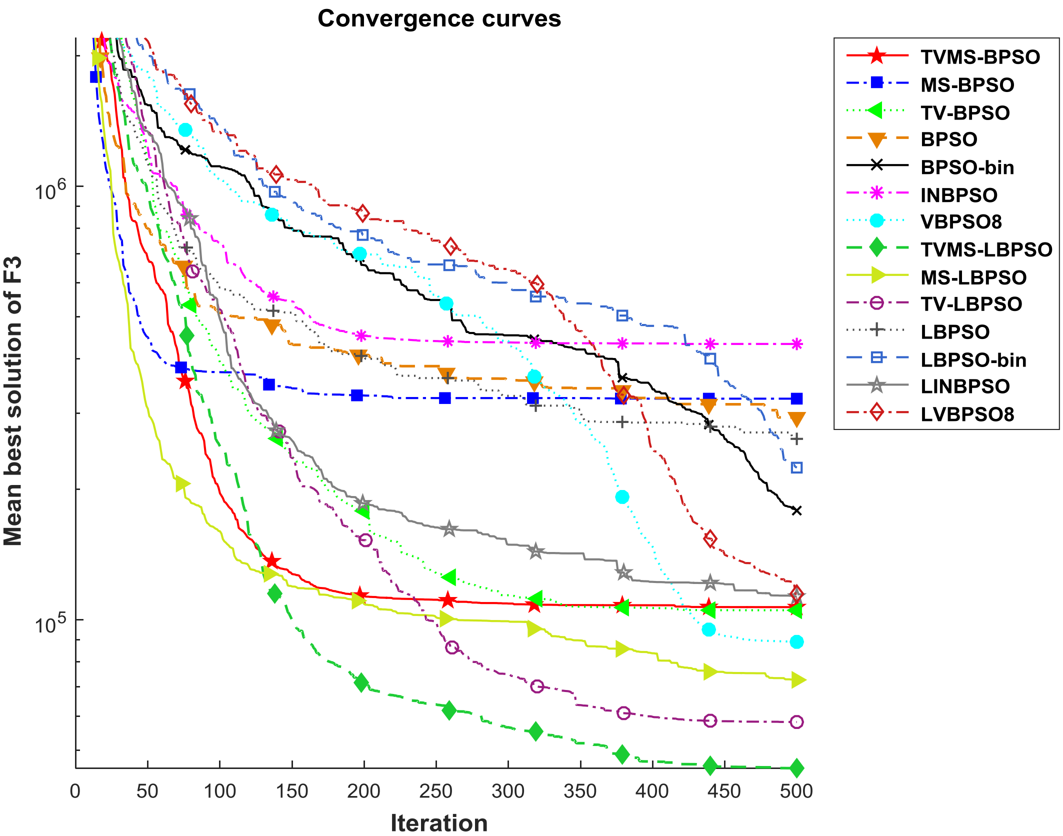


Fig. 4. The convergence curve of CEC 2005 benchmark functions.



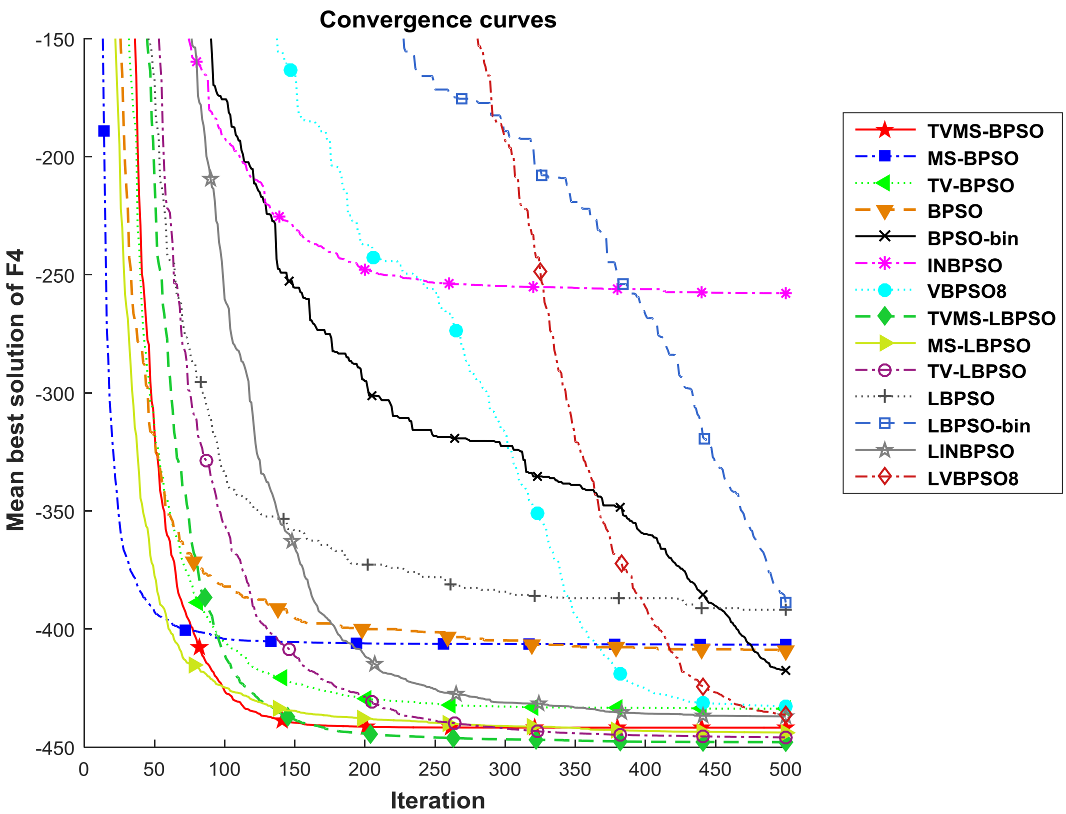
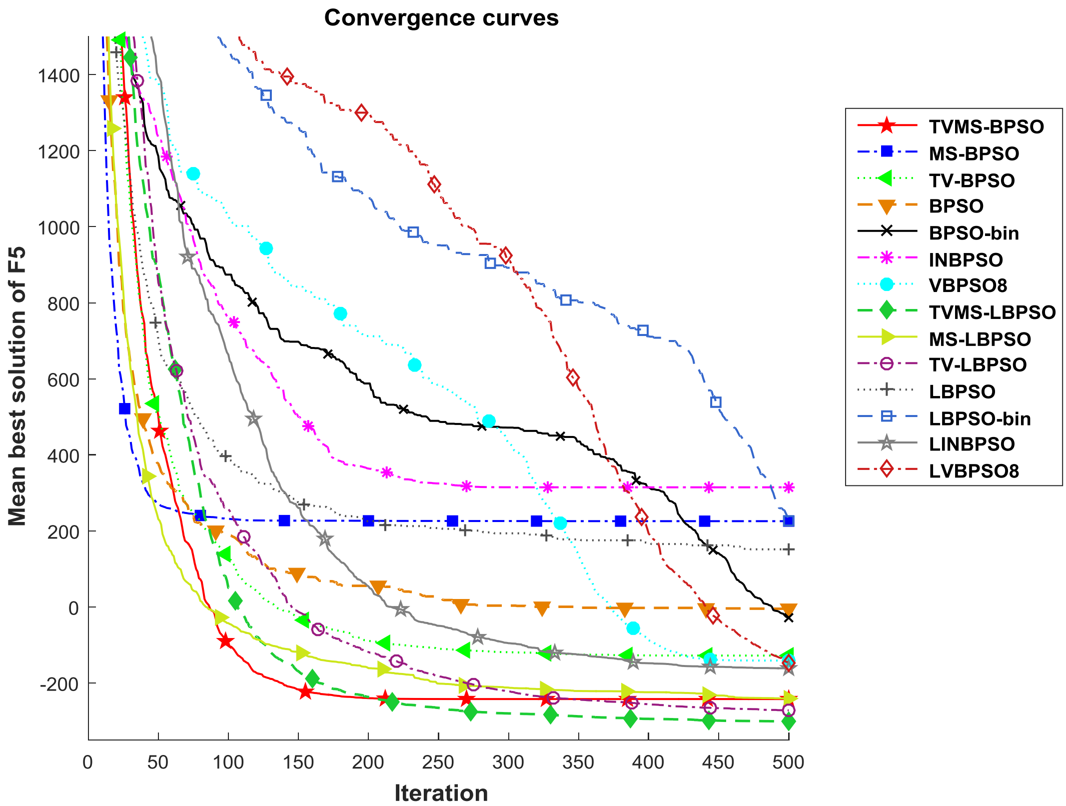


Fig. 4. Continued



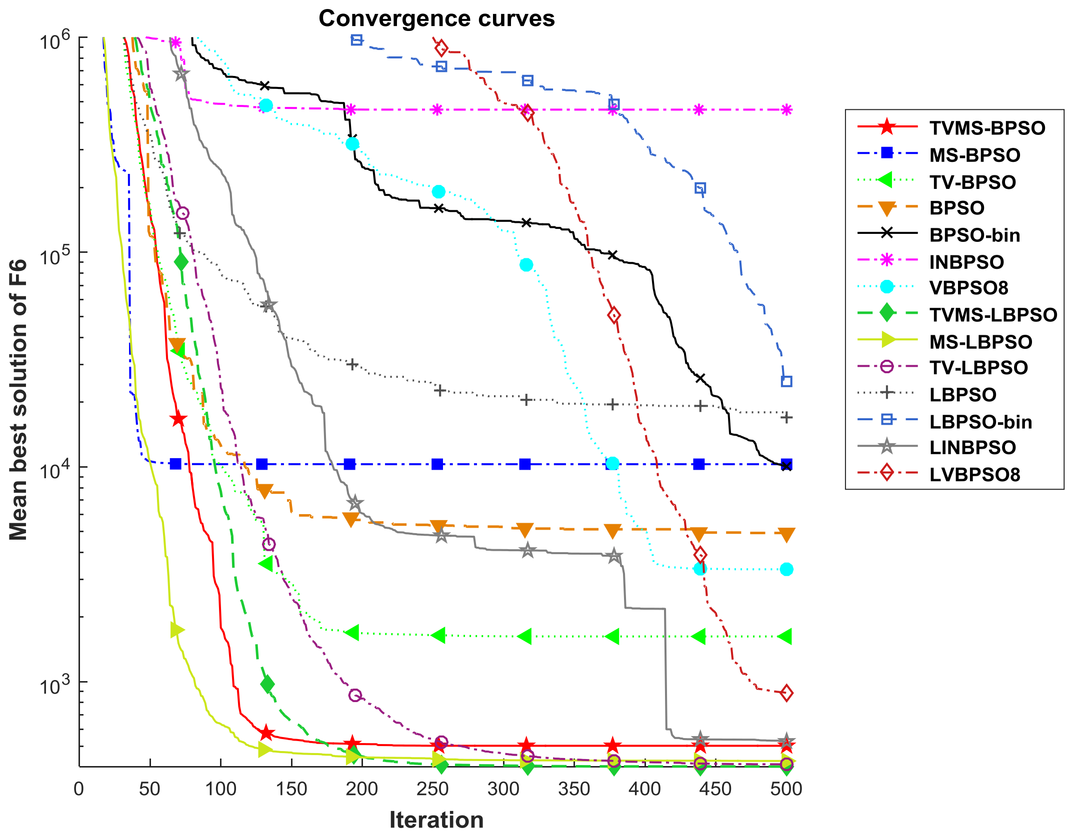
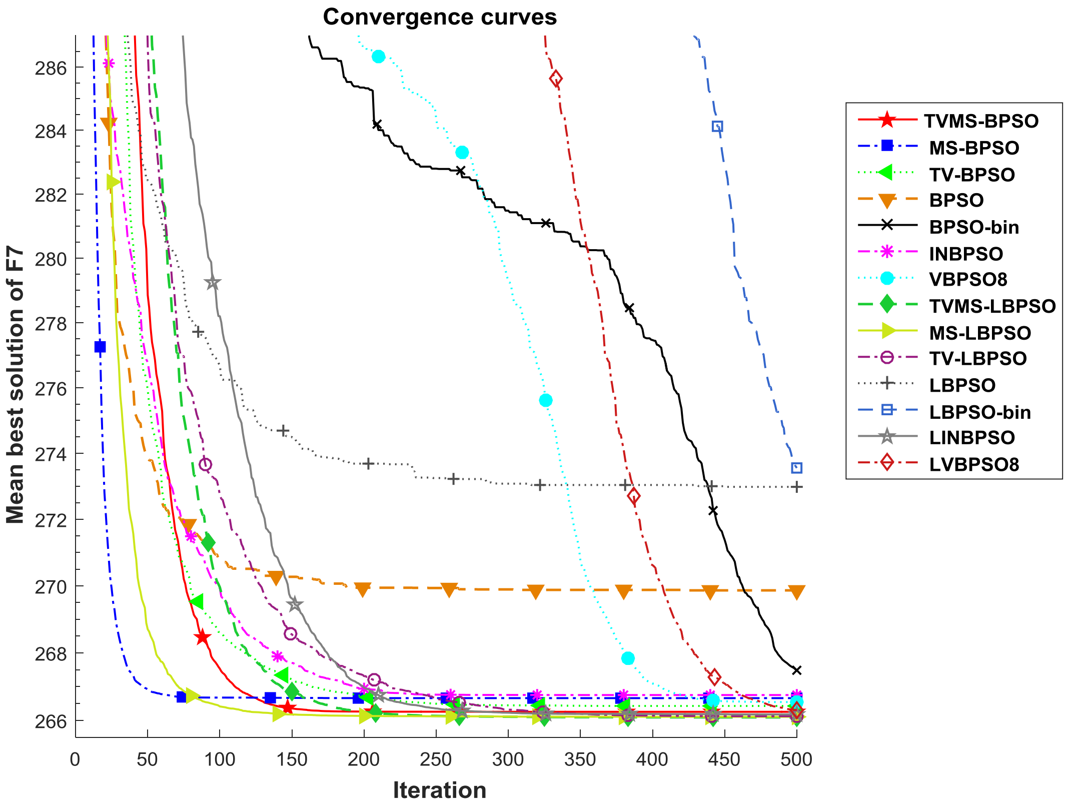


Fig. 4. Continued



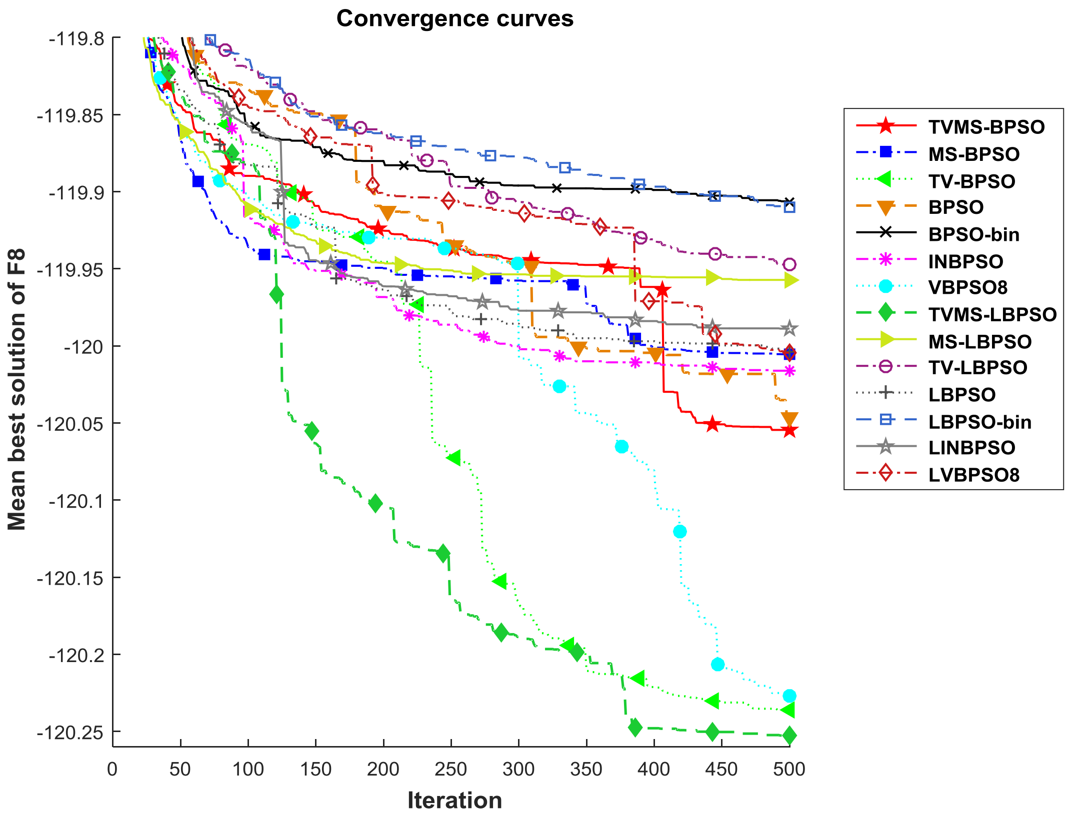
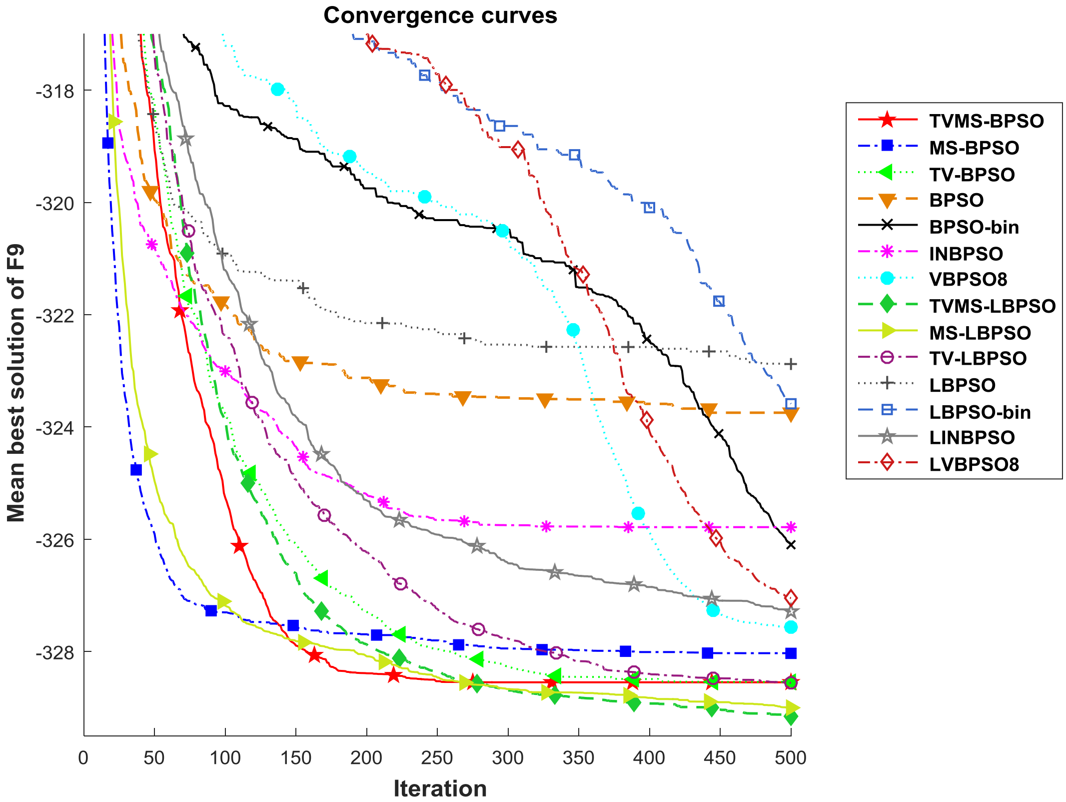


Fig. 4. Continued



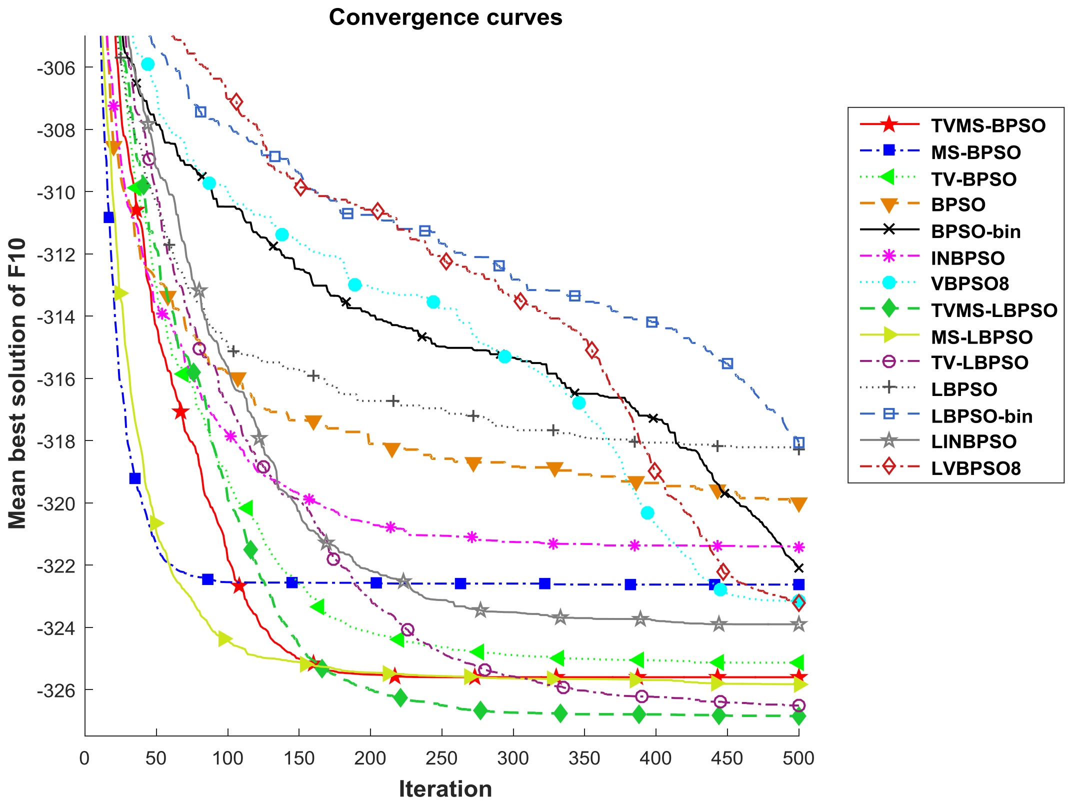
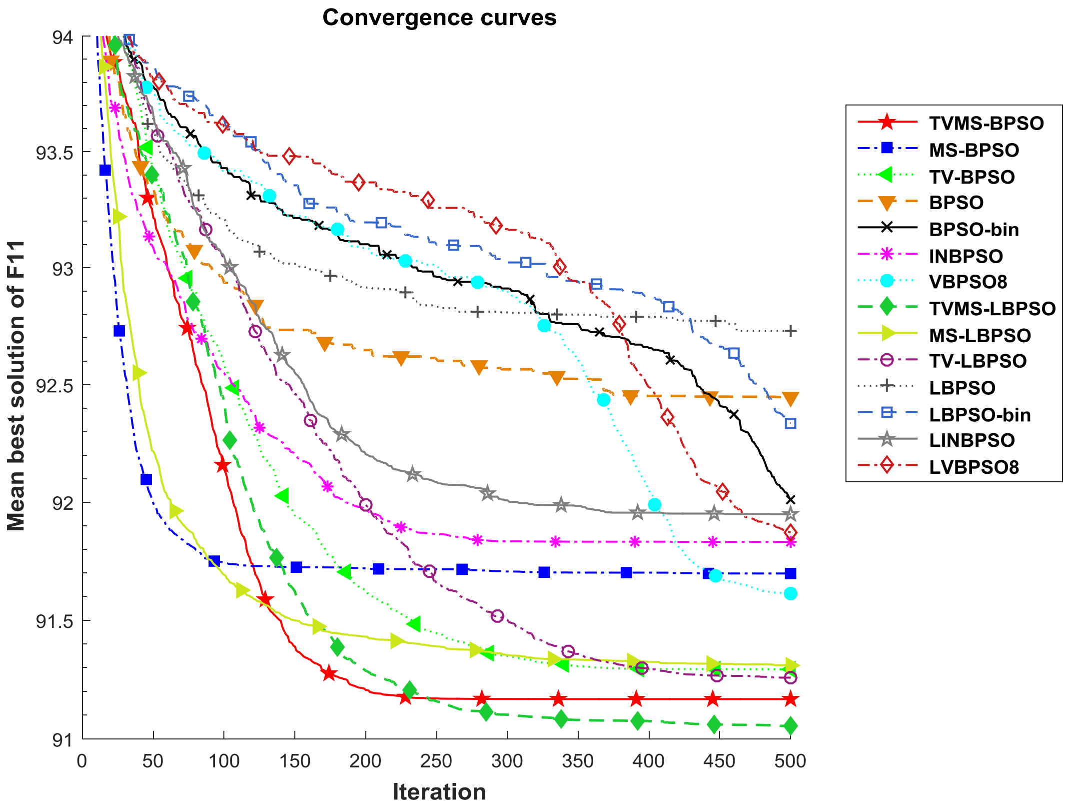


Fig. 4. Continued



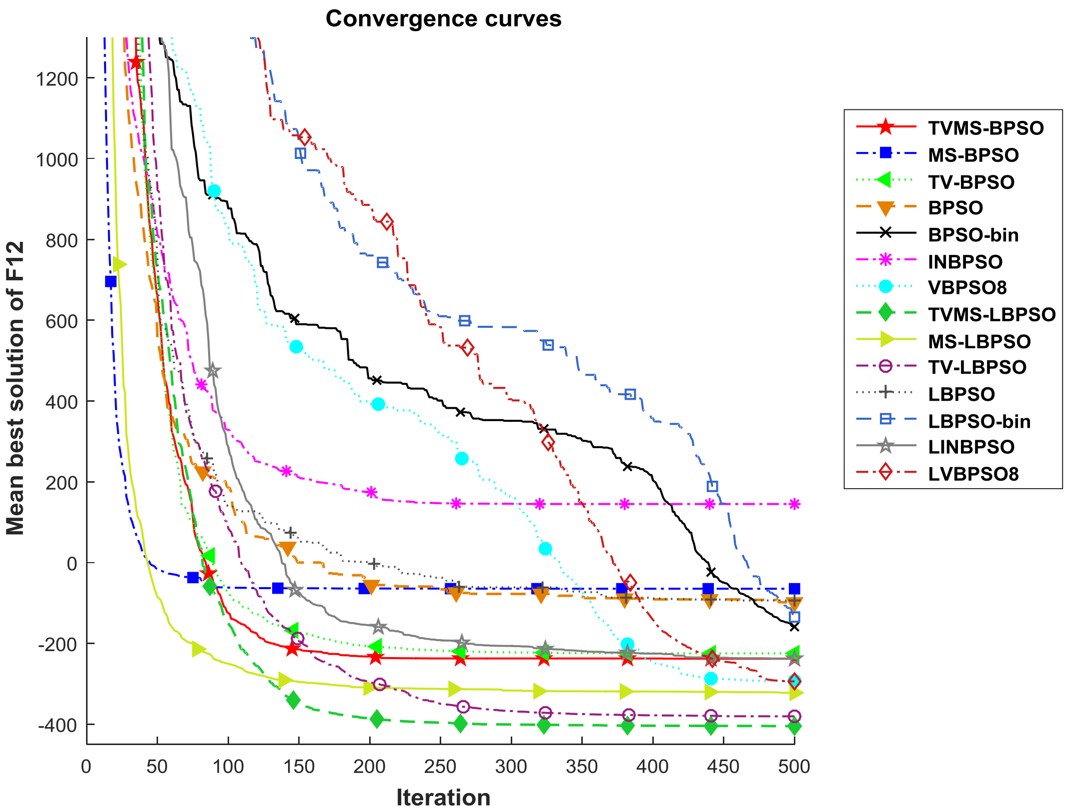
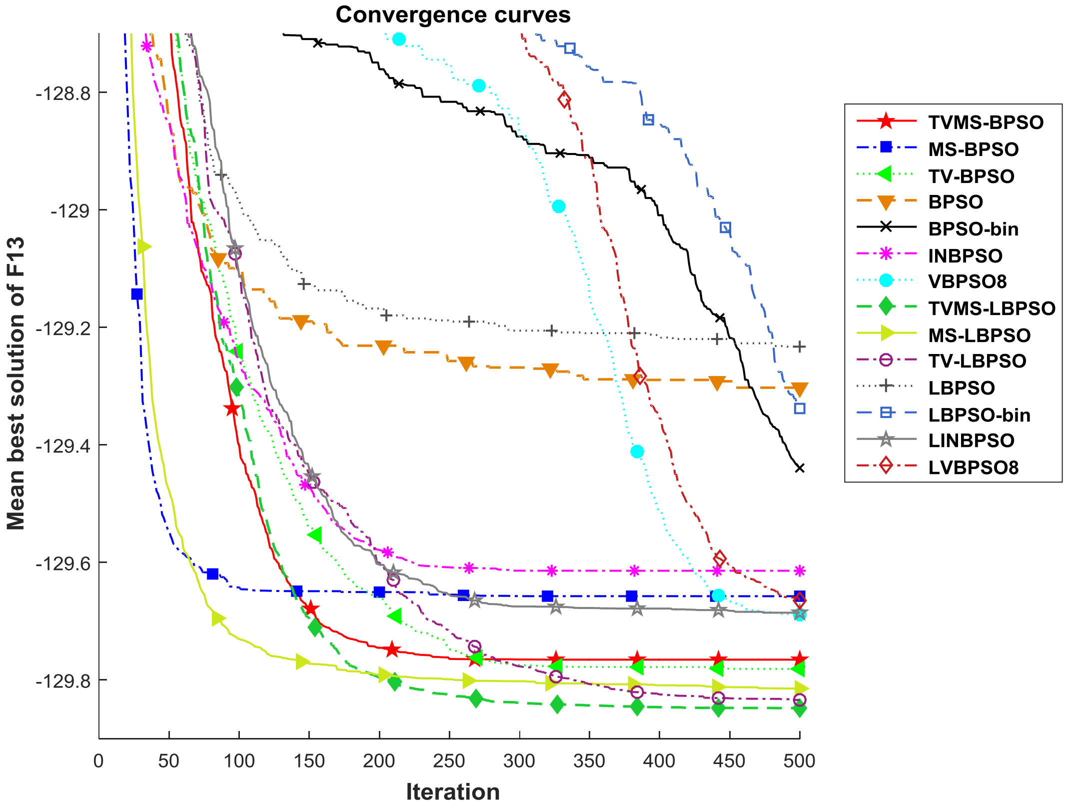


Fig. 4. Continued



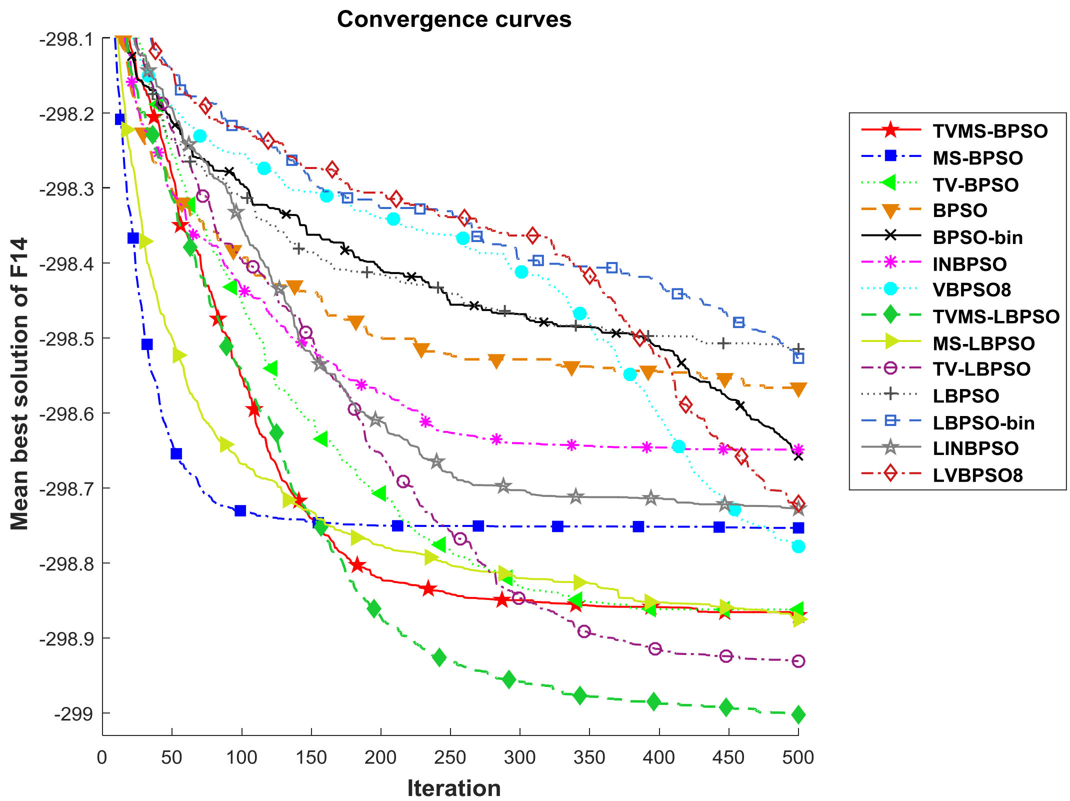
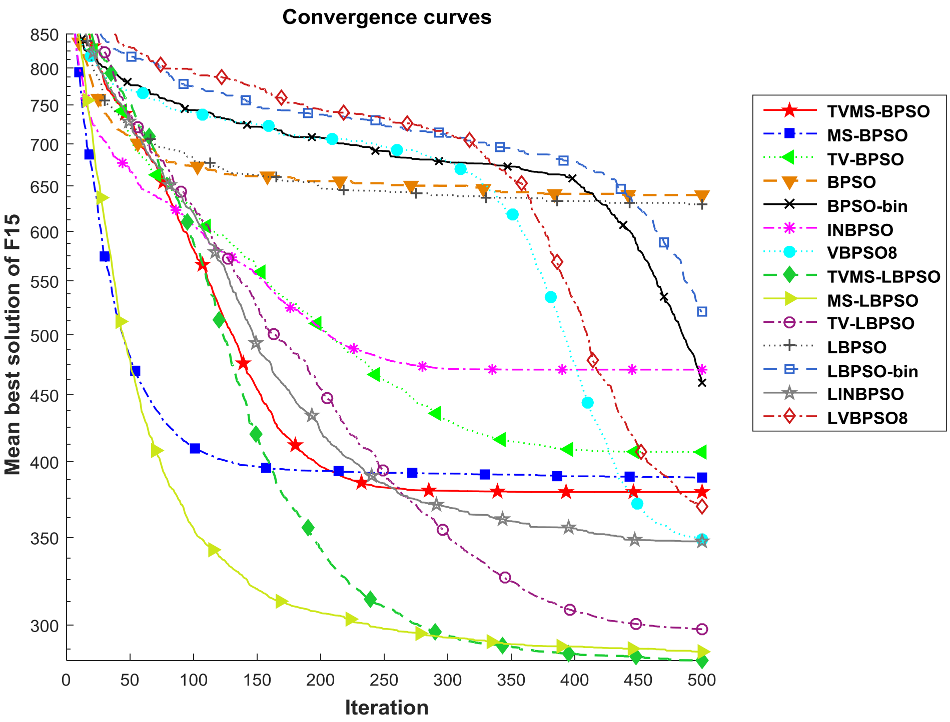


Fig. 4. Continued



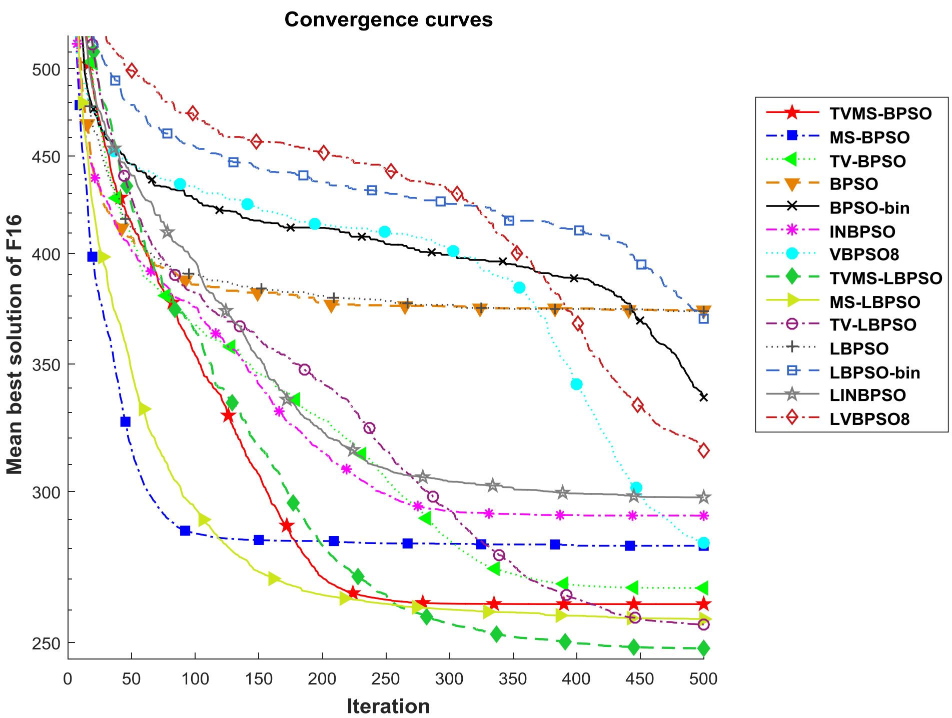
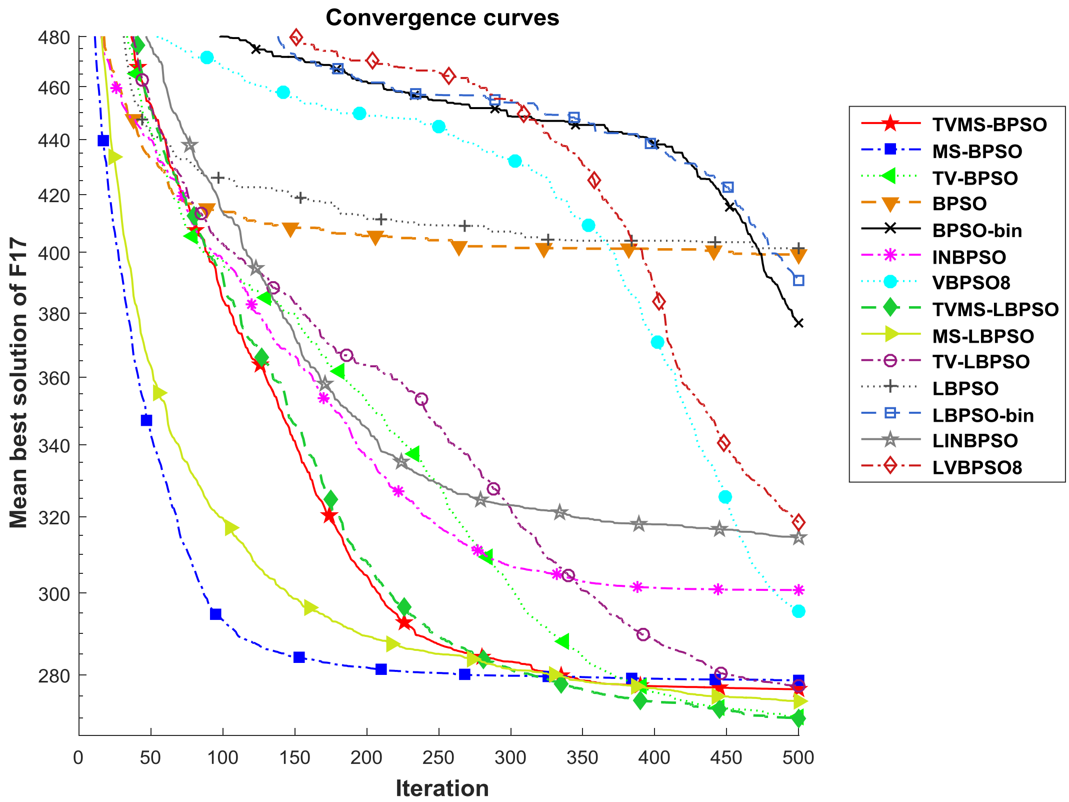


Fig. 4. Continued



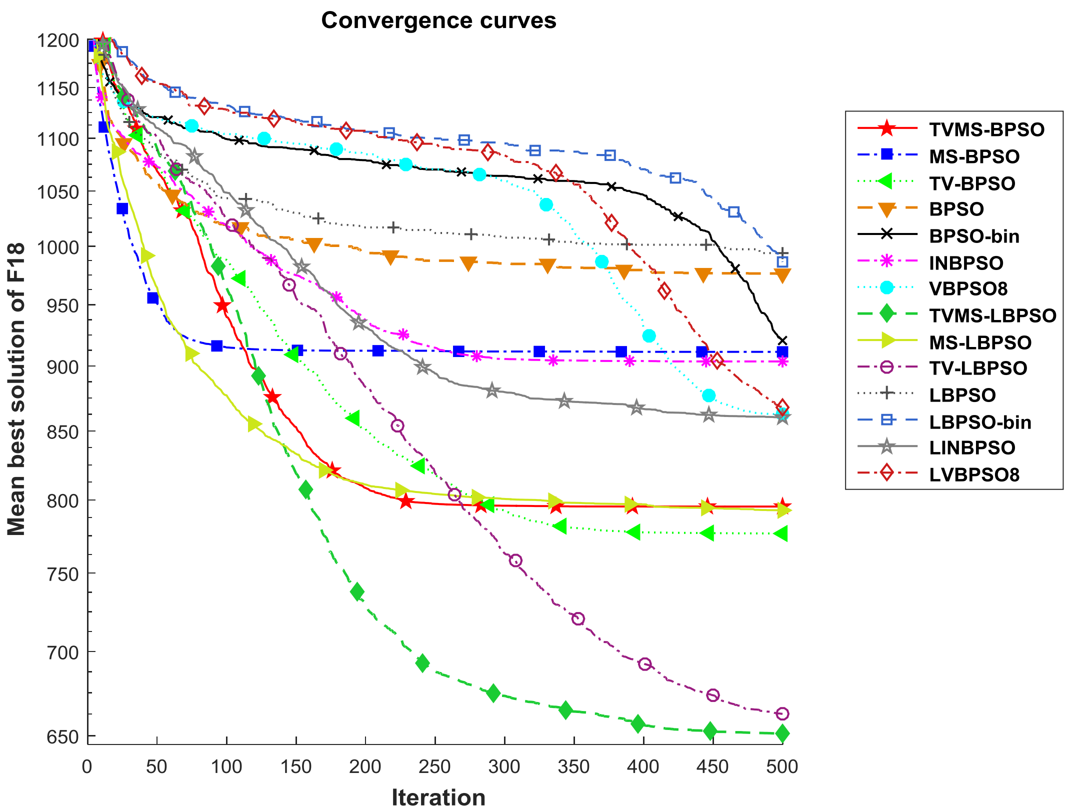
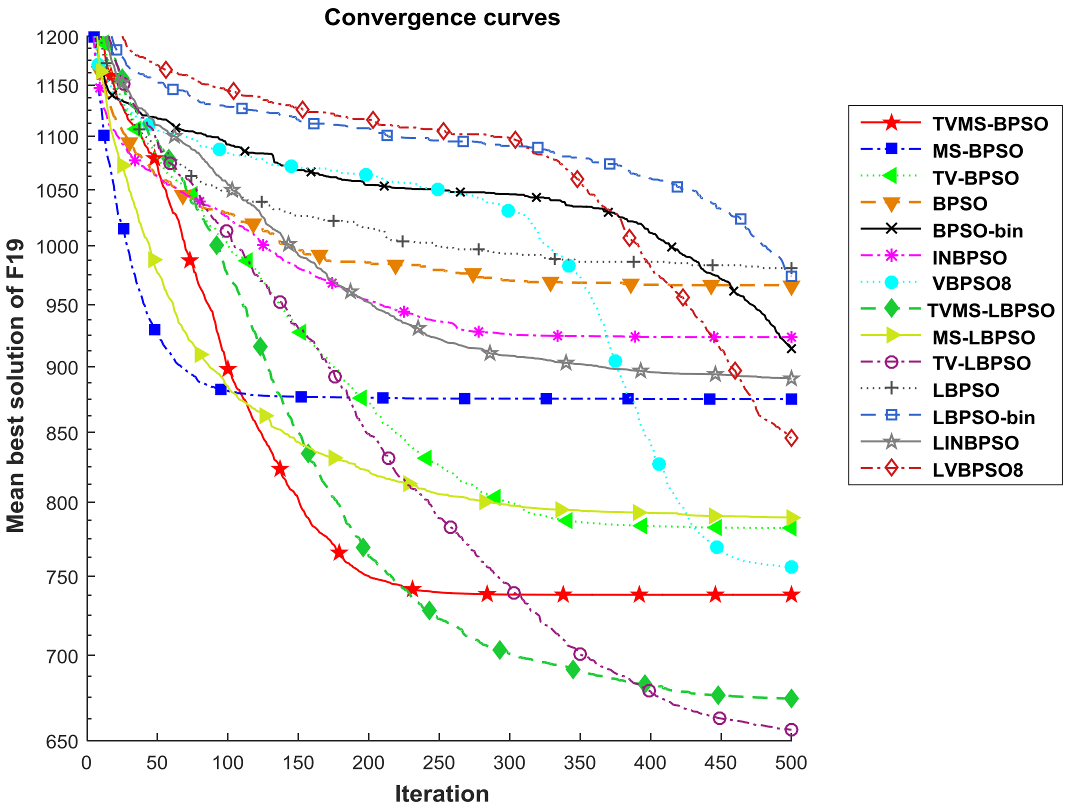


Fig. 4. Continued



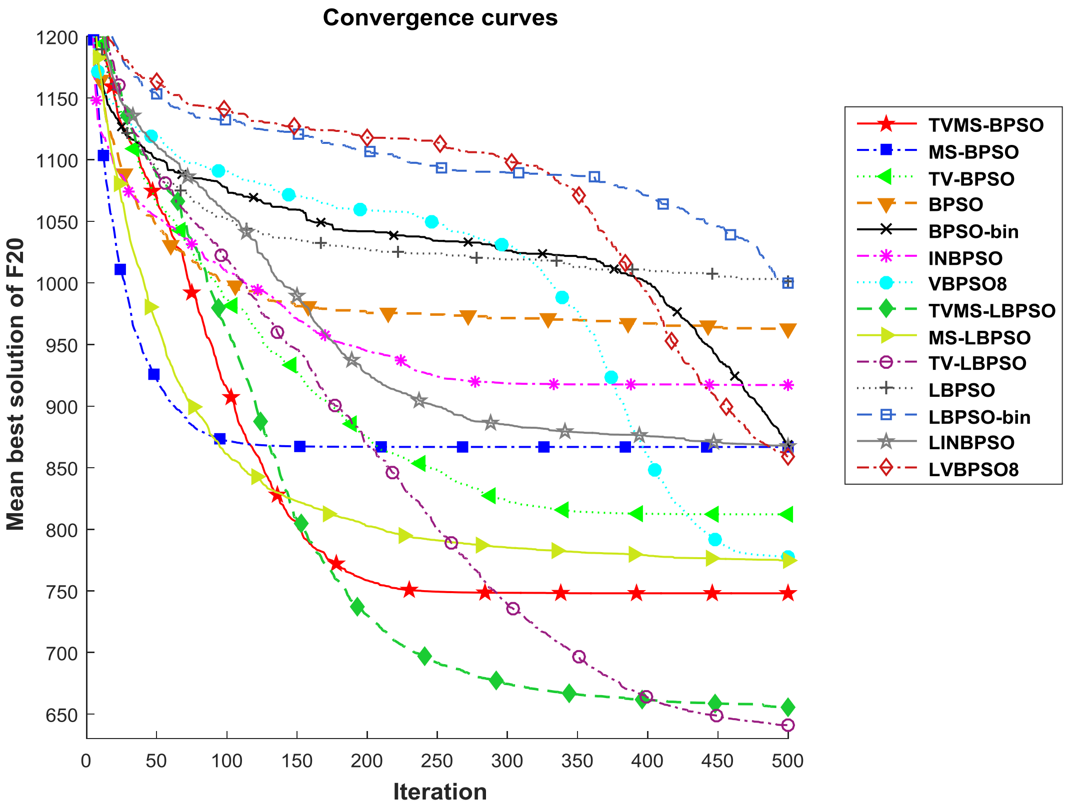
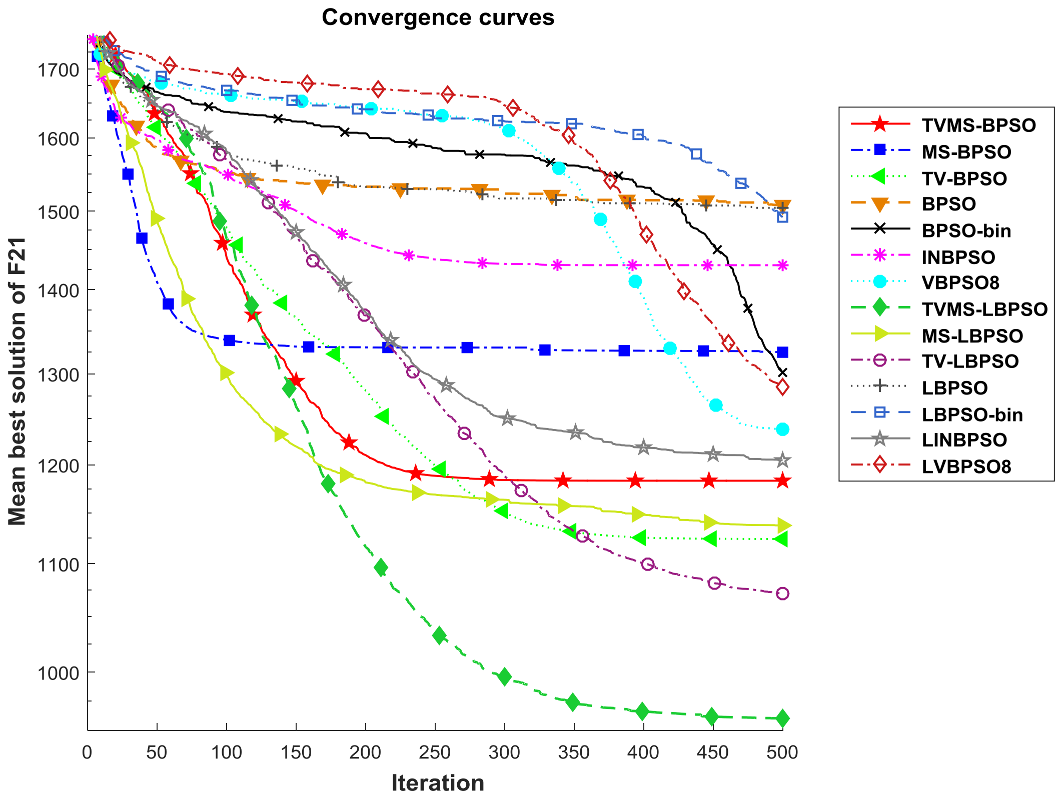


Fig. 4. Continued



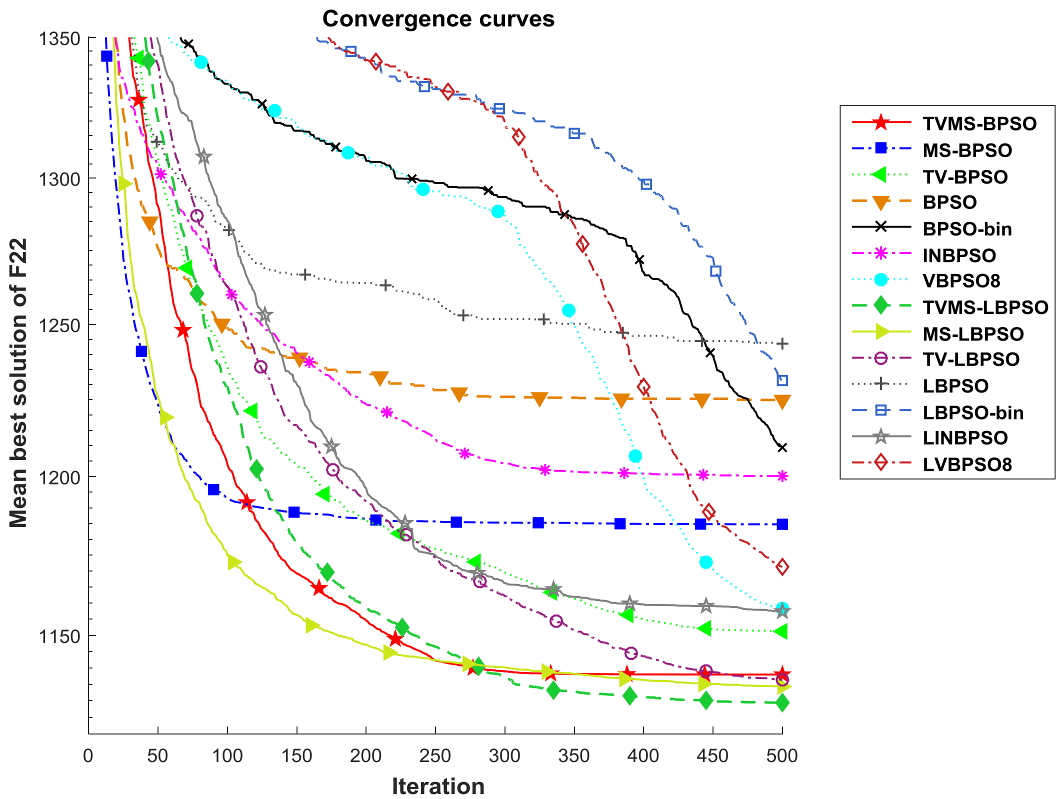
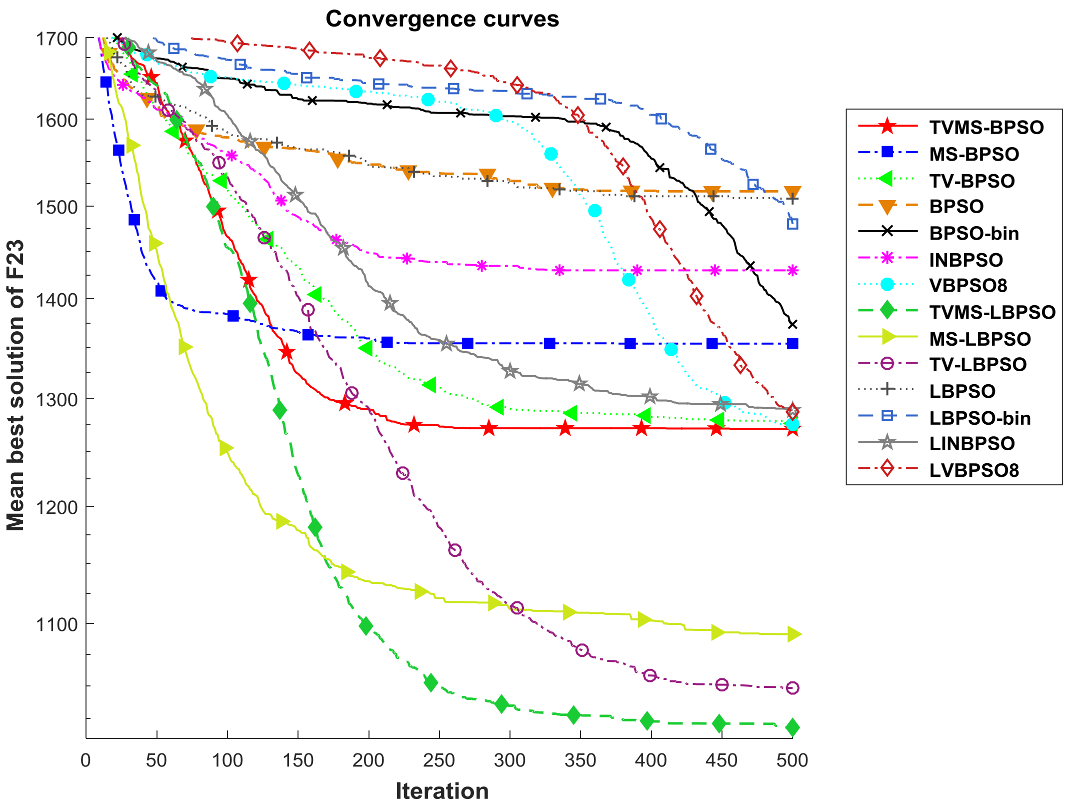


Fig. 4. Continued



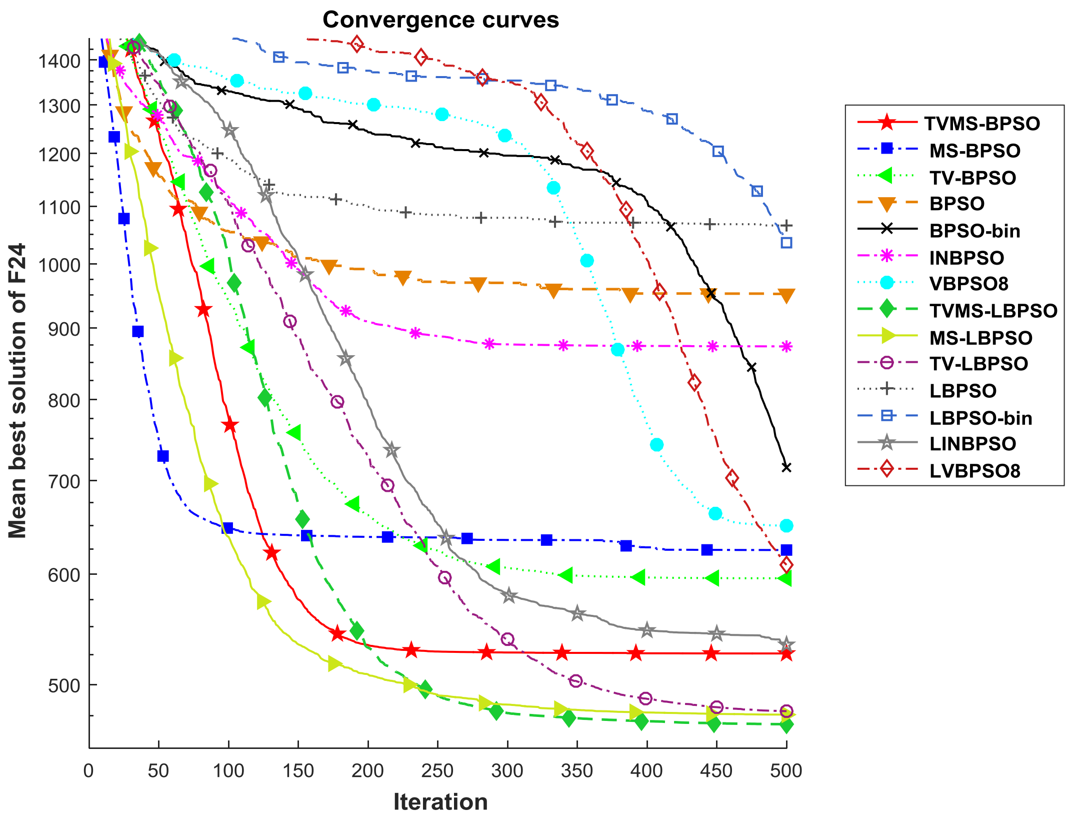


Fig. 4. Continued

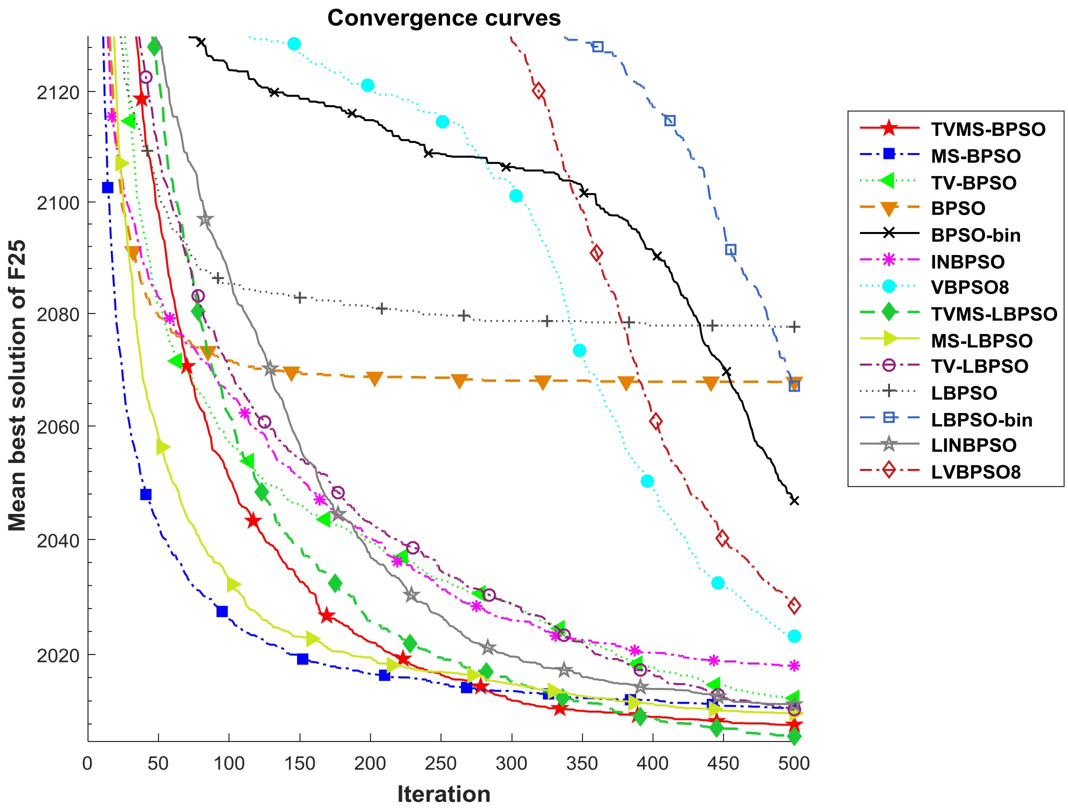


Fig. 4. Continued

The last group of functions is more difficult than the other functions to be solved because they have a very complex structure with a lot of local optima like the real world optimization problems. They are hybrid composition functions (*F15-F25*). As shown in Table 5, the best results are obtained by TVMS-LBPSO for functions *F16-F18,* and *F21-F25*. For functions *F19* and *F20*, TV-LBPSO performs better and returns the best solutions.

Furthermore, the Friedman test is carried out on the results to analyze the obtained results statistically as reported in Table 6. The minimum value is better result in this table. According to Table 6, TVMS-LBPSO achieves the best rank compared with other algorithms. TVMS-BPSO shows a good rank among global topology algorithms. It means that the time-varying transfer function creates a good balance between exploration and exploitation. As seen in the table, MS-LBPSO shows better results compared with LBPSO, LPSO-bin, LINBPSO and LVBPSO8. Besides, the results of Table 6 show that VBPSO8 has a better rank than LVBPSO8. By analyzing the results of Tables 5 and 6, it is concluded that the performance of BPSO for the local and the global topologies improves when the algorithm uses the proposed transfer function.

Fig. 4 demonstrates the convergence curve of all algorithms on CEC 2005 benchmark functions. As observed, TVMS-LBPSO shows significantly better performance than the compared algorithms on different types of problems for four groups. In addition, the proposed method performs robustly in terms of scaling dimensions, shift, rotation, hybrid and composition of difficult test functions.

The particles distributions of TVMS-BPSO in different steps have been simulated in Fig. 5. This figure shows how the proposed method achieves the best result. The tested function in this figure is *F2* as a shifted unimodal function. The dimension of *F2* is set to 2 and the maximum number of iterations is set to 100. The global optimum is [35.6267, −82.9123] in the range [−100, 100] with objective function value= −450 [43]. In the problem, the swarm is randomly initialized in the binary search space. As shown in Fig. 5, the best solution found by the swarm in the initialization phase is very far from the best solution. At the third iteration, the particles try to move towards the global optimum point until all particles are converged to the global optimum point in the 84th iteration. As seen in this figure, the proposed method outperforms to make a balance between exploration and exploitation in BPSO.

5.3. Results and discussion of 0–1 MKP benchmark instances

The 0–1 MKP is one of the most well-known binary optimization problems. n items (or objects) and m knapsacks with limited capacities are considered for the problem. Each item has a weight and profit. In the problem, the aim is to select a subset of items with maximum profit, and without exceeding the capacity of knapsacks. The problem is defined [14]

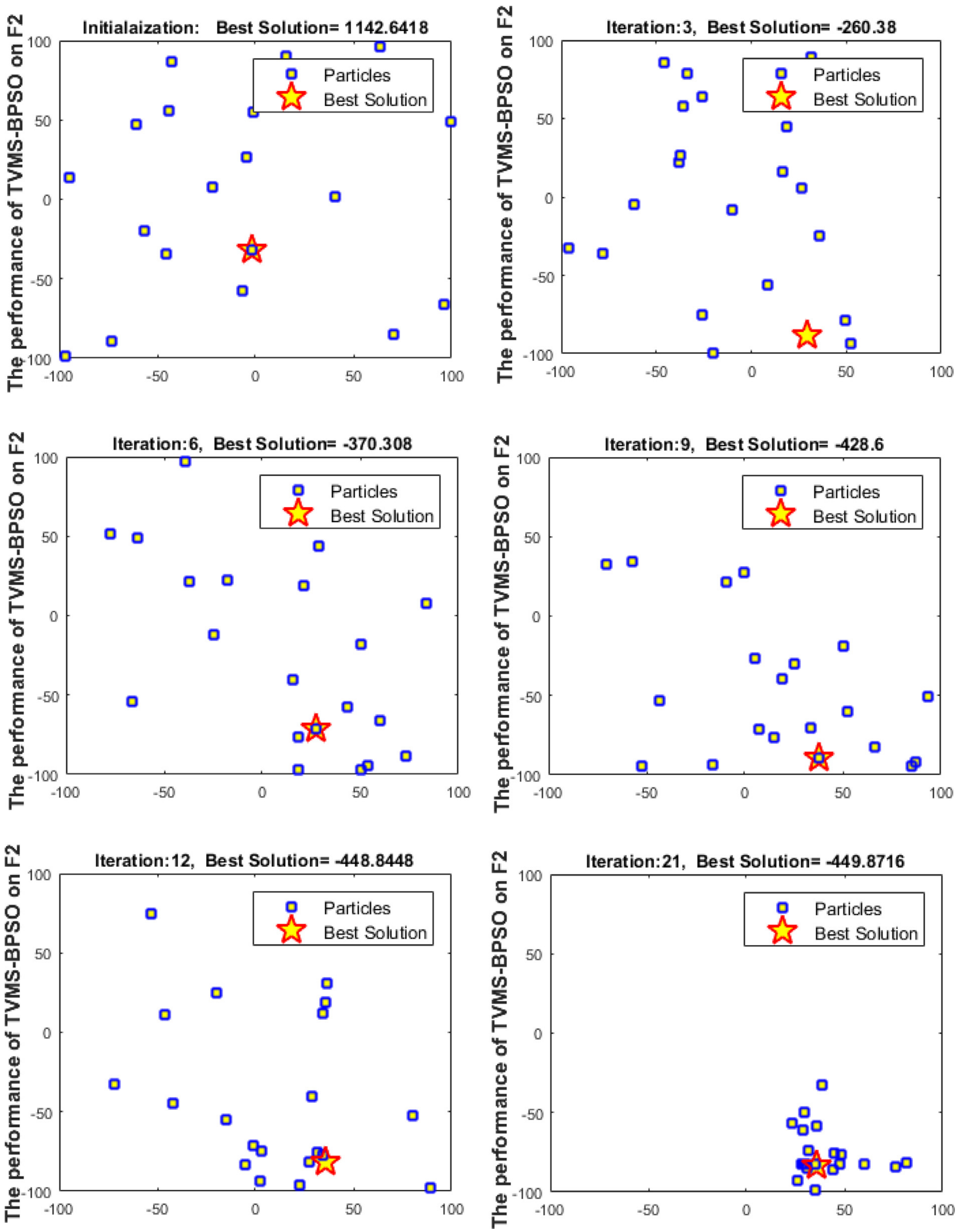


Fig. 5. The performance of TVMS-BPSO on *F2* in the different steps.

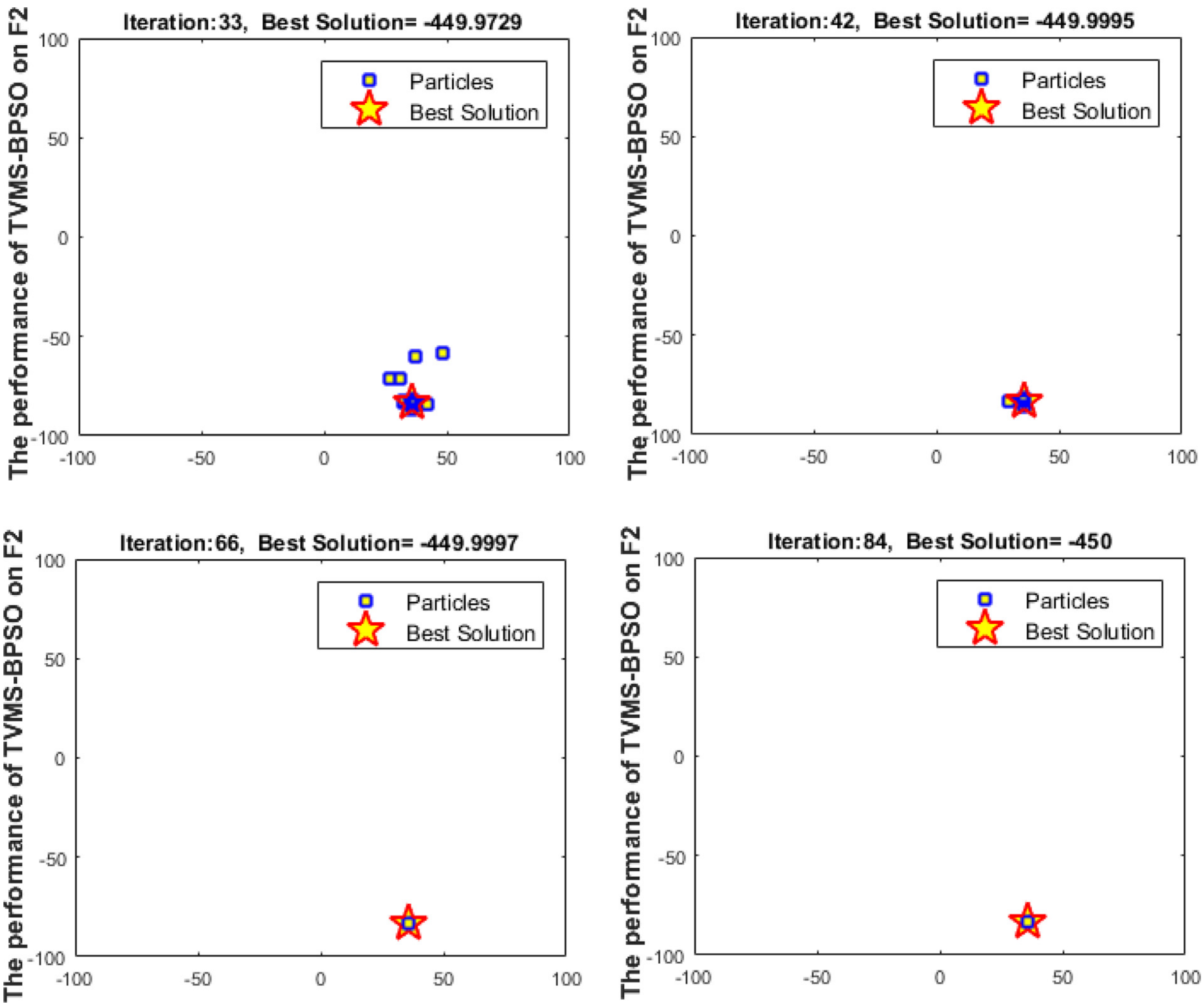
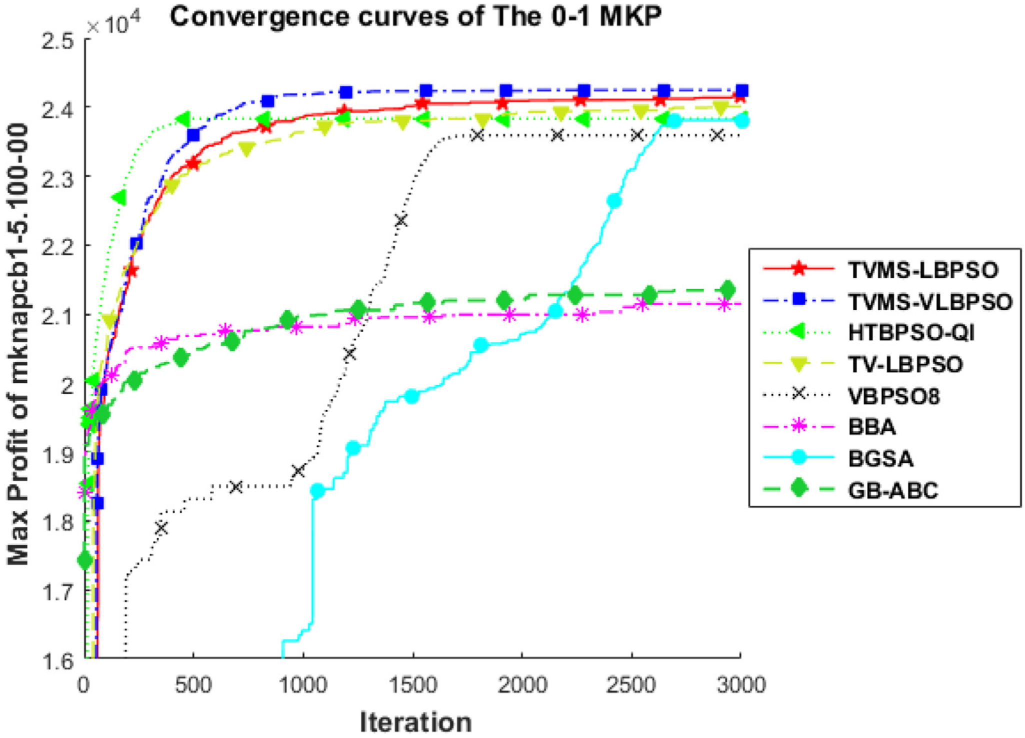


Fig. 5. Continued

Table 7
The 0–1 MKP benchmarks.

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.	mknapcb2–5.250–00	59312	250	5
4.	mknapcb2–5.250–01	61472	250	5
5.	mknapcb3–5.500–00	120130	500	5
6.	mknapcb3–5.500–01	117837	500	5
7.	mknapcb4–10.100–00	23064	100	10
8.	mknapcb4–10.100–01	22801	100	10
9.	mknapcb5–10.250–00	59187	250	10
10.	mknapcb5–10.250–01	58662	250	10
11.	mknapcb6–10.500–00	117726	500	10
12.	mknapcb6–10.500–01	119139	500	10
13.	mknapcb8–30.250–29	150038	250	30
14.	mknapcb9–30.500–29	301021	500	30



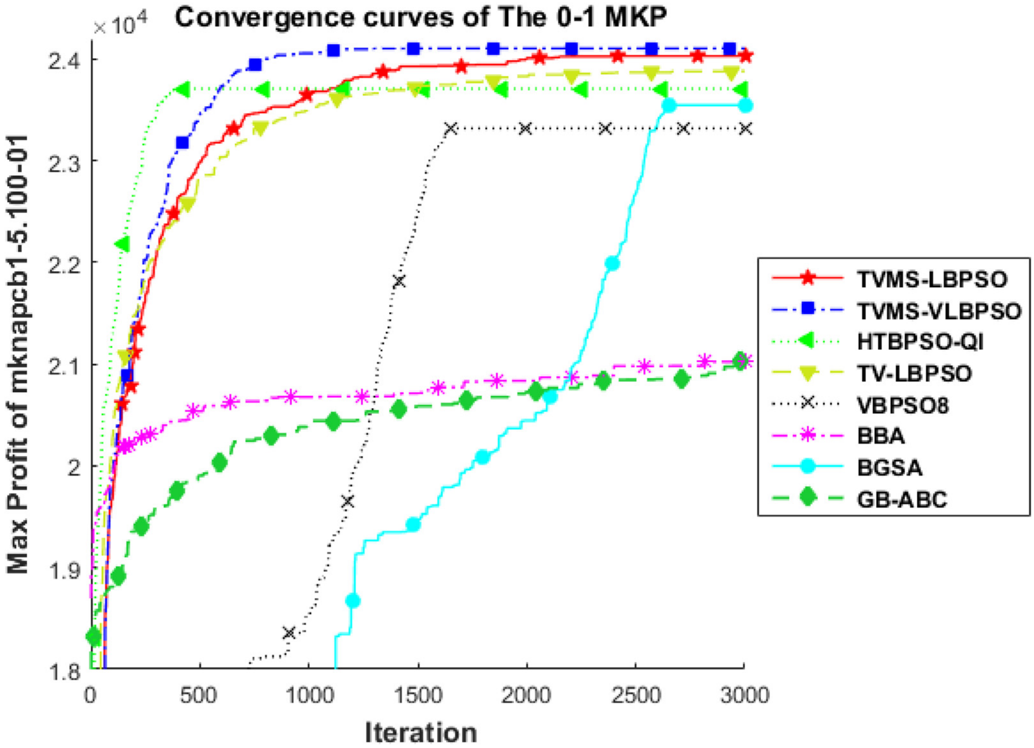
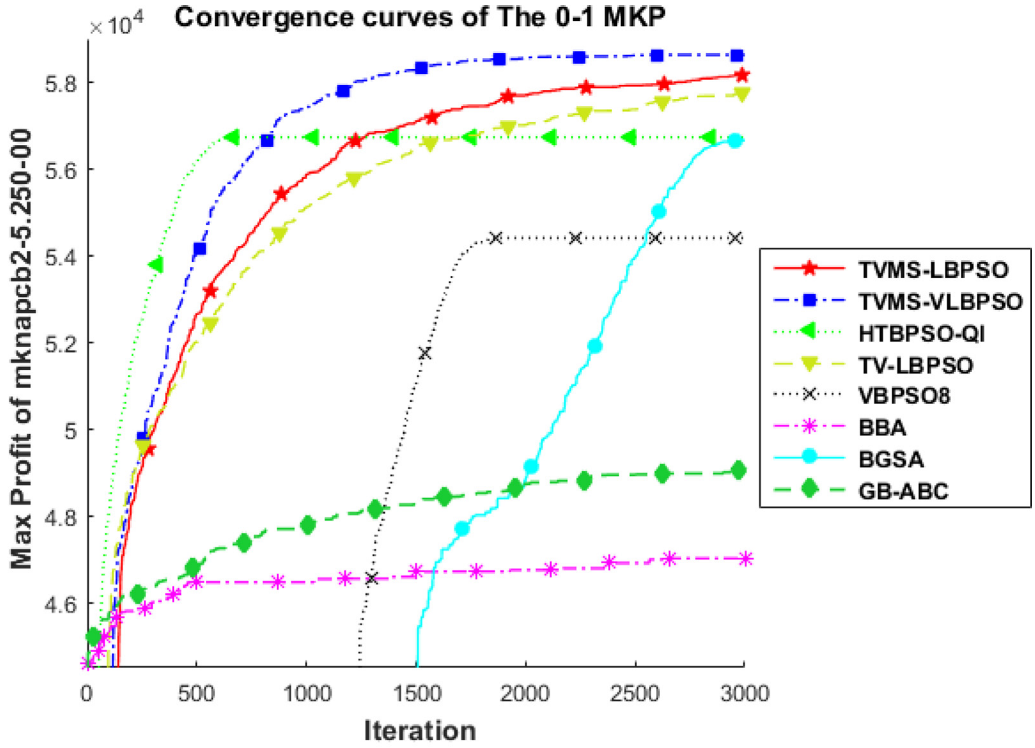


Fig. 6. The convergence curves of the 0–1 MKP (Maximum iteration=3000).



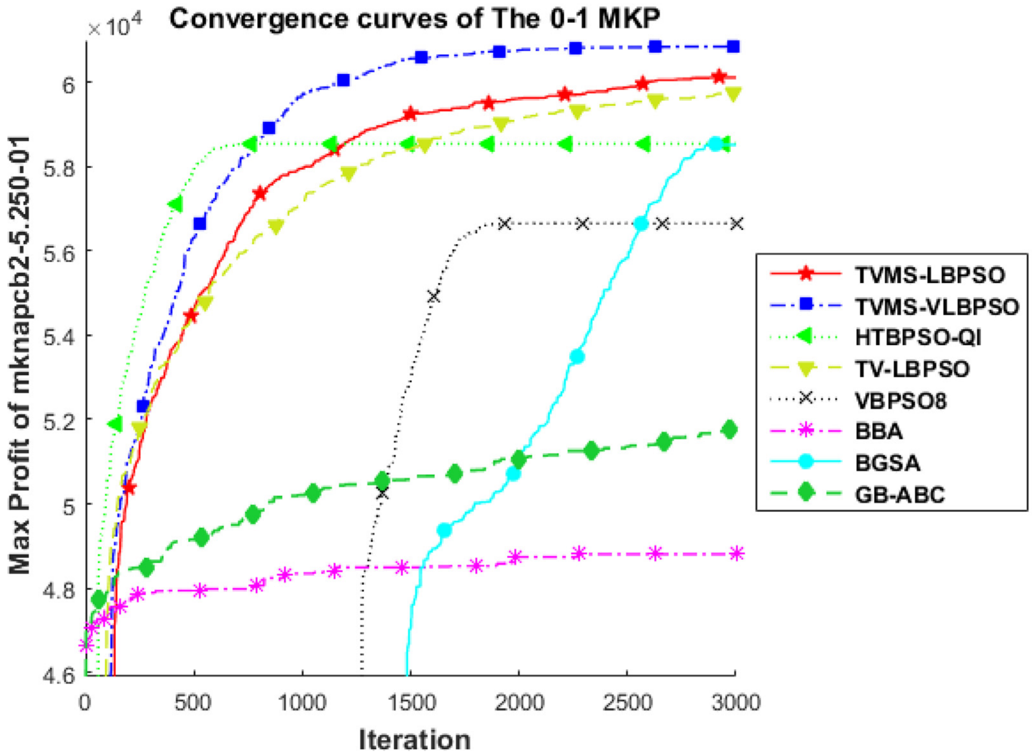
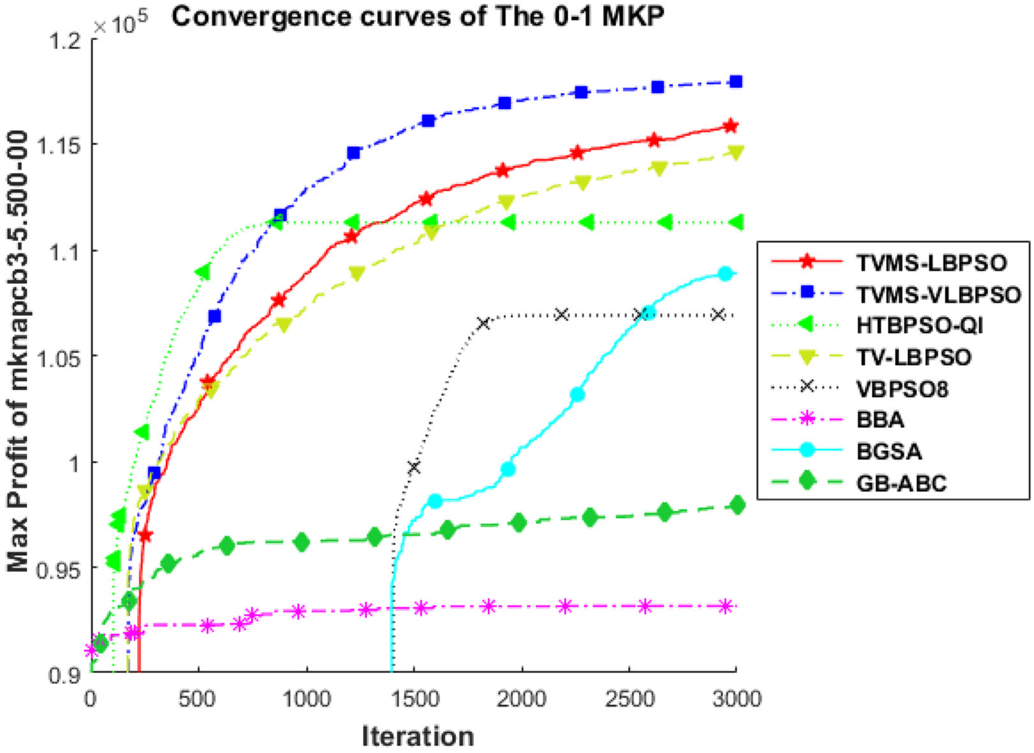


Fig. 6. Continued



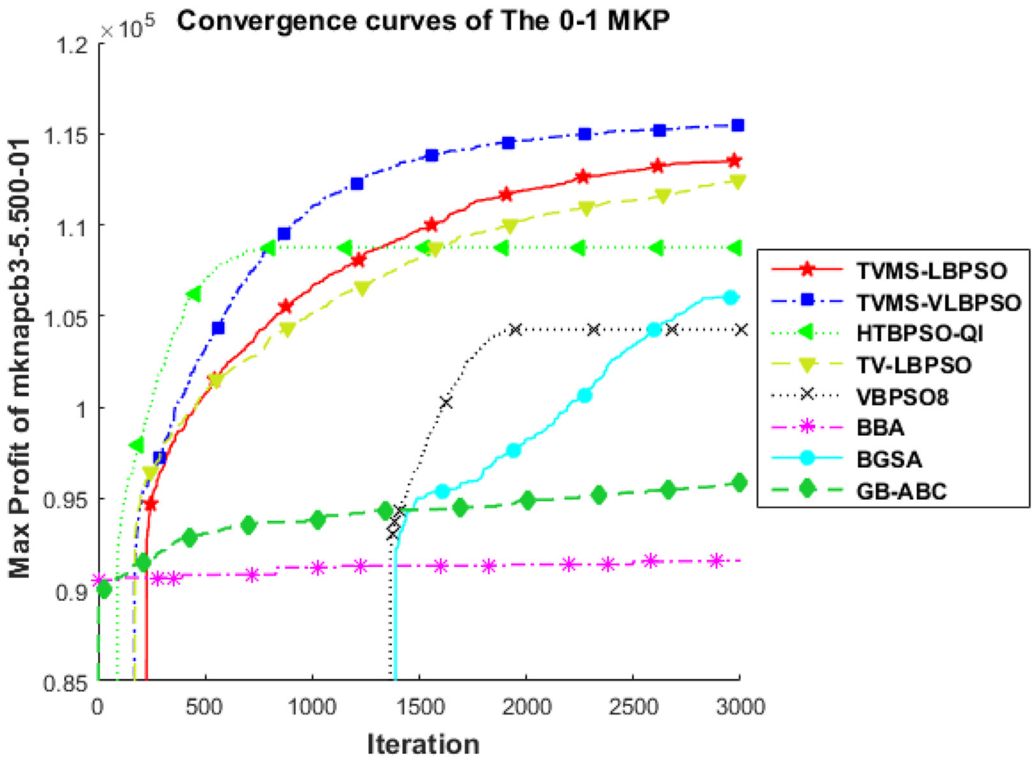


Fig. 6. Continued

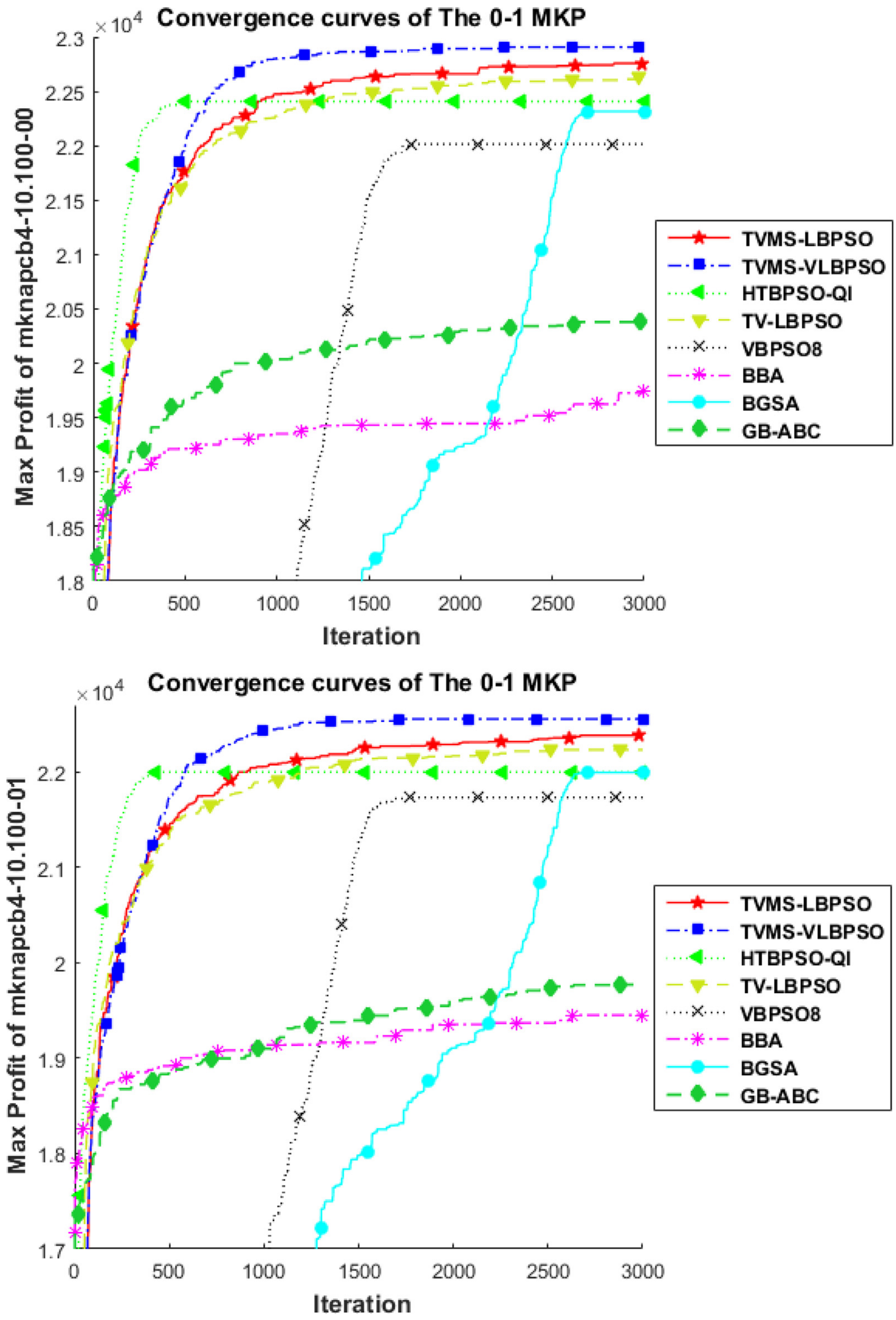
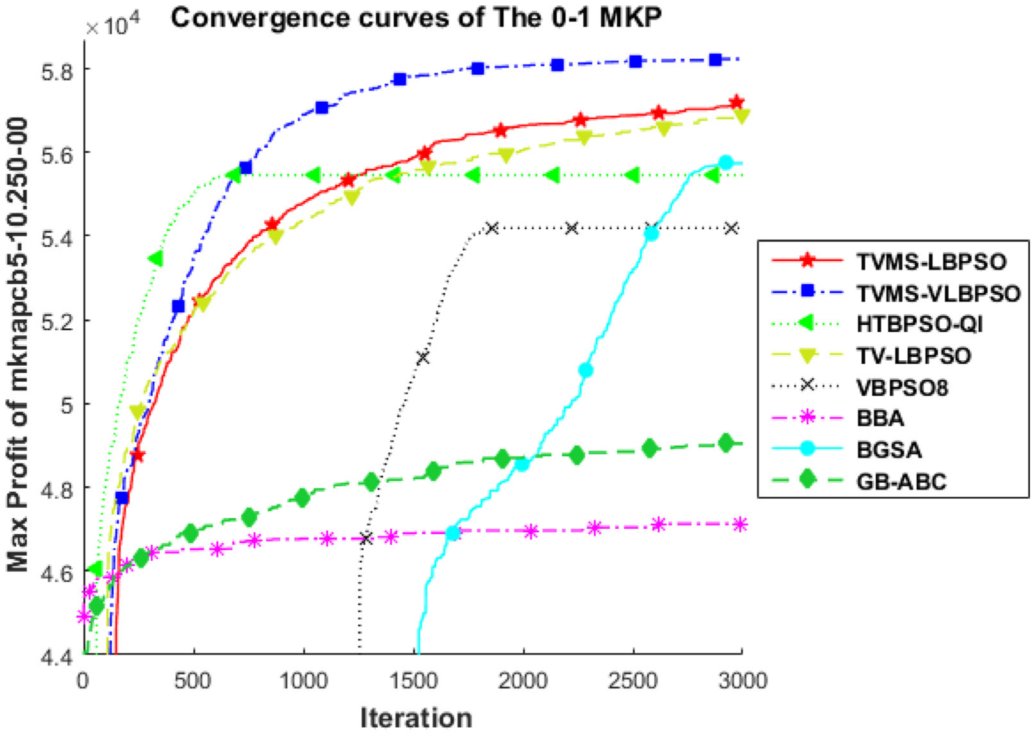


Fig. 6. Continued



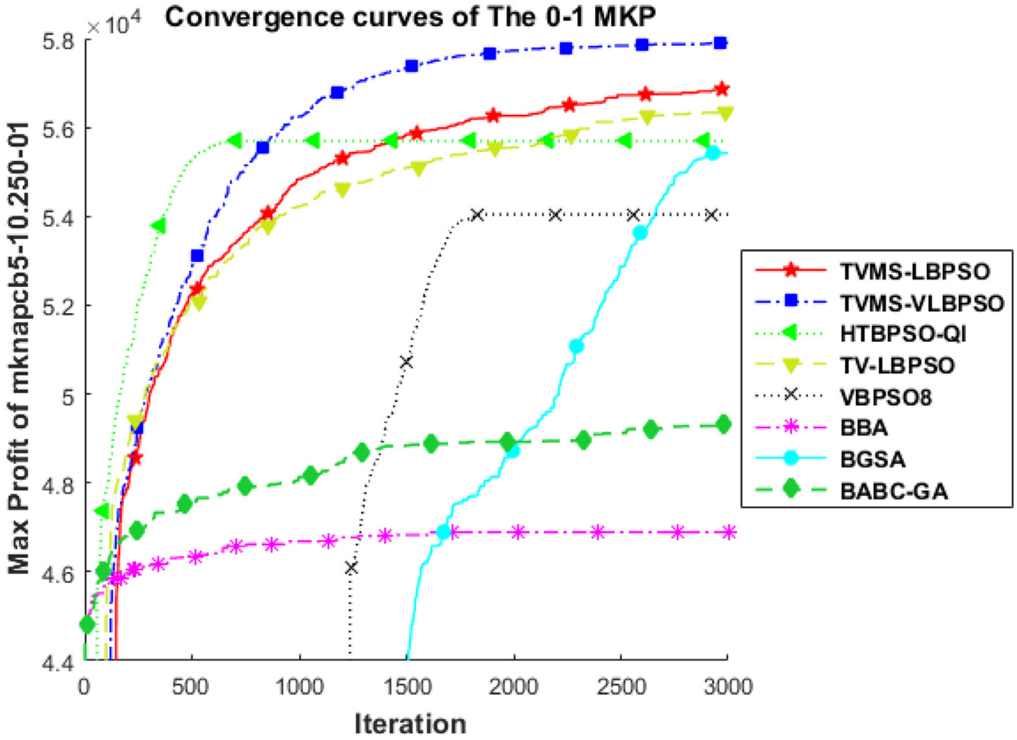
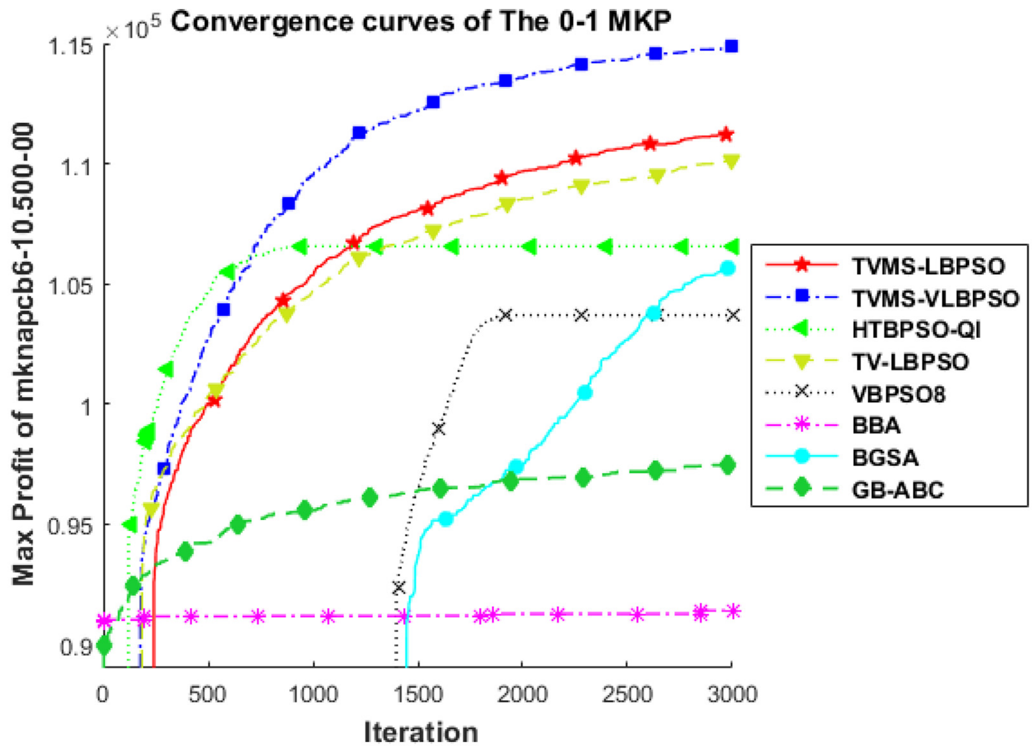


Fig. 6. Continued



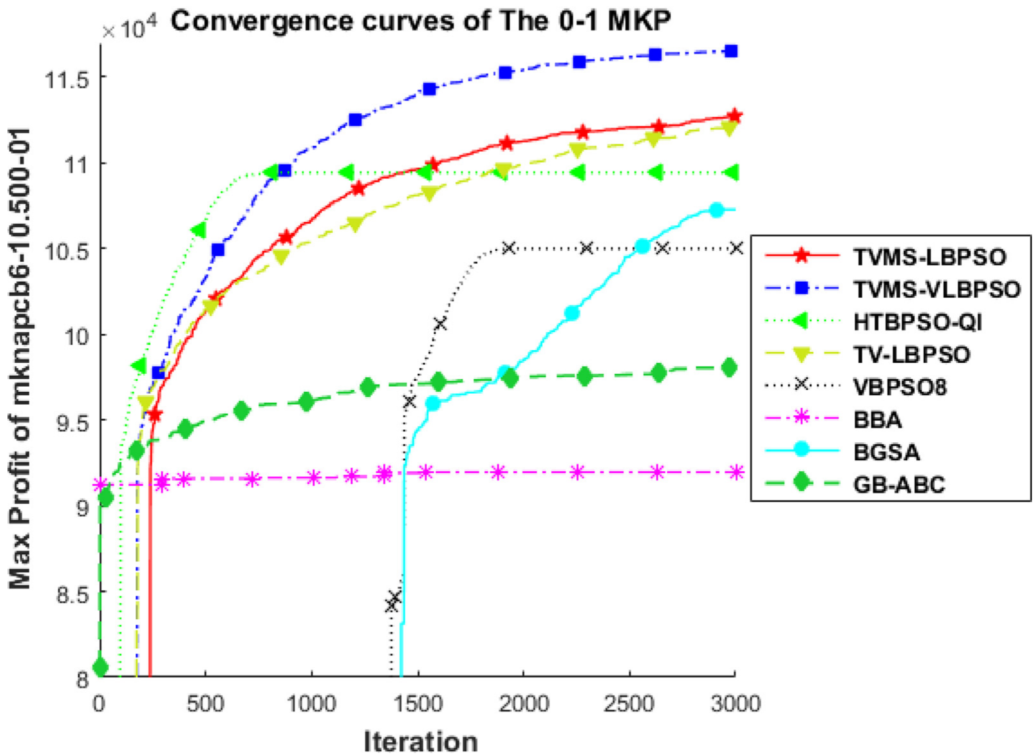


Fig. 6. Continued

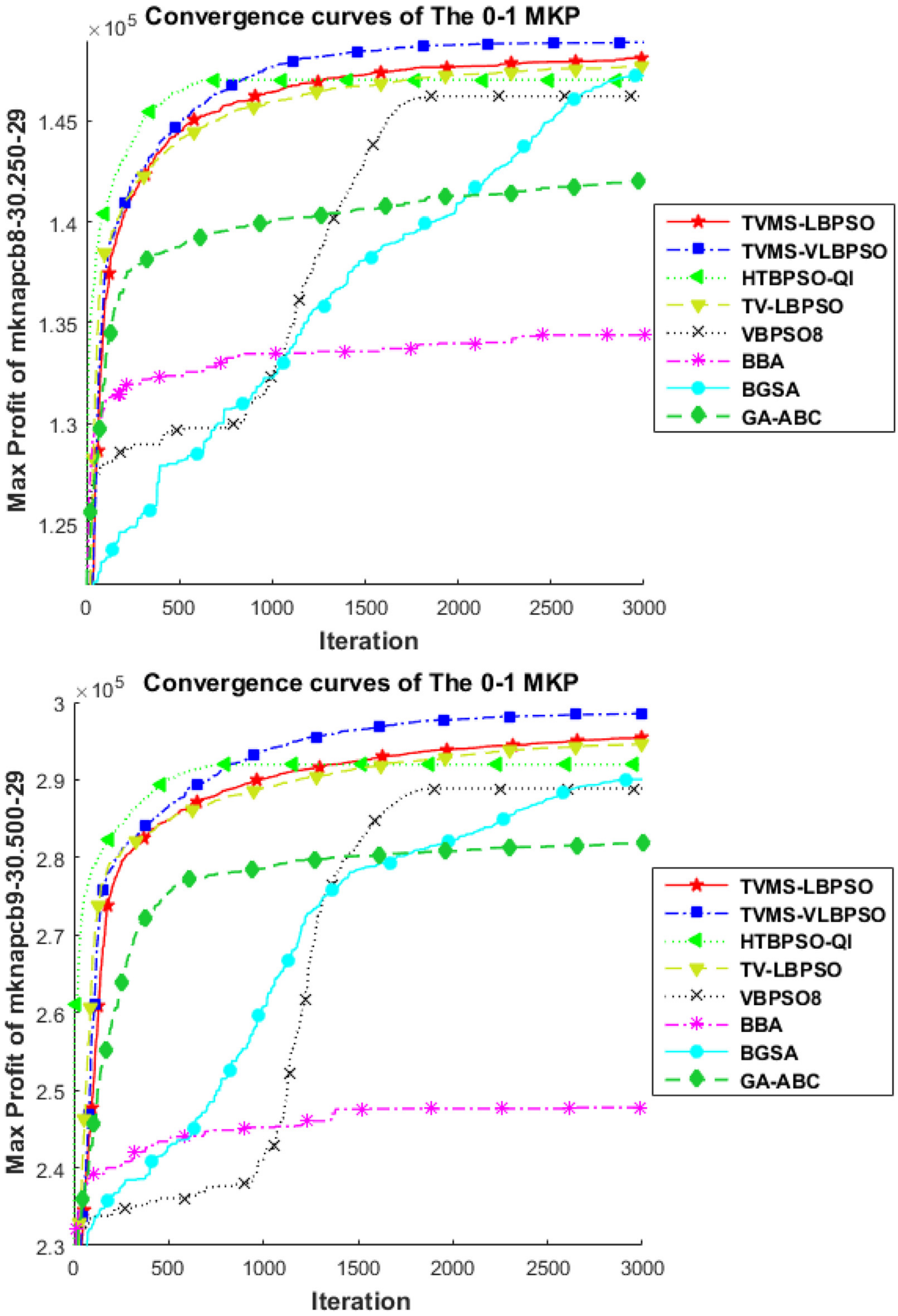


Fig. 6. Continued

Table 8
The results of algorithms on the benchmarks of Table 4, using PF technique (*Maximum iteration=3000*).

Benchmark	*Max Profit*	TVMS-LBPSO	TVMS-VLBPSO	HTBPSO-QI	TV-LBPSO	VBPSO8	BBA	BGSA	GB-ABC
mknapcb1–5.100–00	*Best*	24273	**24330**	24253	24228	24026	22092	24193	22174
	Mean	24150.2	**24245.9**	23831.8	24007.8	23591.7	21146.8	23813.1	21353.6
	Worst	24005	**24094**	23589	23760	23317	20481	23482	20527
	STD	96.5549	**80.307**	236.38	155.181	198.995	428.068	210.976	476.218
mknapcb1–5.100–01	*Best*	24216	**24258**	24077	24038	23821	21497	24054	21690
	Mean	24031.6	**24105.6**	23710.8	23883	23318.6	21024.5	23548	21023.3
	Worst	23927	**23904**	23241	23697	22864	20776	22786	20086
	STD	97.3301	**95.9493**	234.356	103.276	251.05	256.864	378.862	594.001
mknapcb2–5.250–00	*Best*	58497	**58851**	57527	58042	55131	48258	57369	51170
	Mean	58173.9	**58631.6**	56731.5	57745.3	54408.8	47019.9	56663.7	49057.7
	Worst	57856	**58271**	55030	57266	53616	45996	55852	47809
	STD	209.187	**178.053**	731.228	259.476	532.764	652.205	492.747	1064.24
mknapcb2–5.250–01	*Best*	60414	**61130**	59706	60073	57053	49982	59444	52990
	Mean	60118.9	**60865.7**	58542.2	59741.1	56654.6	48806	58521.7	51783.8
	Worst	59863	**60611**	56544	59411	55630	47310	57930	49372
	STD	167.681	**131.321**	971.68	238.506	454.979	904.028	512.883	1364.58
mknapcb3–5.500–00	*Best*	116763	**118419**	113322	115416	108388	96376	112052	99875
	Mean	115806	**117906**	111288	114653	106895	93159.3	108882	97910
	Worst	114790	**116663**	108513	113776	104417	90563	107461	94452
	STD	649.206	**546.345**	1418.4	588.321	1464.83	1591.85	1317.06	1618.58
mknapcb3–5.500–01	*Best*	114080	**116189**	111533	112974	106132	94927	107026	97371
	Mean	113547	**115495**	108770	112467	104259	91593.1	106054	95825.4
	Worst	112574	**114516**	105269	112090	101824	88177	104160	93967
	STD	492.407	**451.36**	2004.1	291.554	1220.43	2358.86	999.244	1348.57
mknapcb4–10.100–00	*Best*	23050	**23055**	22715	22763	22412	20479	22631	21303
	Mean	22757.6	**22906.6**	22407.7	22632.1	22014.8	19739.5	22316.2	20392.3
	Worst	22544	**22777**	21823	22427	21605	18867	21471	19381
	STD	129.161	**95.6268**	261.561	117.378	222.513	564.708	333.24	597.074
mknapcb4–10.100–01	*Best*	22602	**22753**	22290	22572	22097	19916	22507	20667
	Mean	22385.1	**22553.5**	21997.7	22229.6	21730.8	19449.2	21992.7	19768.9
	Worst	22177	**22470**	21356	21907	21273	18617	21636	19166
	STD	125.079	**80.6202**	272.849	179.335	315.888	432.69	274.026	565.647
mknapcb5–10.250–00	*Best*	57760	**58424**	57180	57653	55482	48231	56501	51035
	Mean	57186.1	**58231.8**	55457.4	56884	54193.6	47126.7	55738.9	49047
	Worst	56289	**57965**	52213	56071	51821	45948	54937	47315
	STD	432.731	**165.257**	1410.04	471.341	1021.35	693.514	539.56	1358.08
mknapcb5–10.250–01	*Best*	57769	**58214**	56792	56744	54731	48255	55939	50554
	Mean	56878.8	**57901.3**	55690.9	56351.8	54044.6	46892.6	55416.9	49318.4
	Worst	56196	**57620**	55054	56054	53112	45479	55039	47151
	STD	485.817	**177.318**	630.333	207.935	489.824	900.393	290.37	1021.32
mknapcb6–10.500–00	*Best*	111927	**115343**	111576	111257	105826	93471	107144	98930
	Mean	111228	**114866**	106556	110135	103684	91387.4	105609	97451.2
	Worst	109697	**113818**	100739	109273	101834	89532	104263	95929
	STD	855.045	**495.481**	3389.74	712.441	1465.28	1193.36	868.752	999.849
mknapcb6–10.500–01	*Best*	114060	**117793**	111838	113348	106725	95397	109168	100049
	Mean	112741	**116576**	109444	112081	105002	91964.2	107285	98103.2
	Worst	110936	**115637**	107328	111085	103468	90149	105505	95874
	STD	1018.8	**746.423**	1575.89	746.251	1230.73	1719.7	1183.74	1316.05
mknapcb8–30.250–29	*Best*	148509	**149139**	148222	148202	147238	135903	148018	143249
	Mean	148139	**148898**	147039	147753	146228	134389	147243	142009
	Worst	147293	**148455**	145804	147287	145353	130768	146771	140264
	STD	333.064	**185.733**	730.989	229.276	556.81	1395.87	383.589	946.789
mknapcb9–30.500–29	*Best*	296459	**299170**	294265	296148	291343	252402	291481	284252
	Mean	295532	**298557**	291962	294596	288869	247772	290066	281874
	Worst	294190	**298042**	289155	293468	286071	242431	288837	278128
	STD	807.211	**384.621**	1570.87	912.92	1609.32	3072.76	776.809	1723.29
Avg. error of Best profit (%)		1.88%	**0.79%**	3.10%	2.53%	5.84%	15.97%	4.17%	12.01%
Avg. error of Mean profit (%)		2.63%	**1.29%**	4.94%	3.28%	7.27%	18.25%	5.56%	14.33%

as follows:

$$Maximize \sum_{i=1}^{n} p_i x_i$$
$$Subject\ to \sum_{i=1}^{n} w_{ij} x_i \le W_j$$
$$x_i \in \{0, 1\}, 1 \le i \le n, 1 \le j \le m, \quad (43)$$

Table 9
The results of algorithms on the benchmarks of Table 4, using PF technique (*Maximum iteration=5000*).

Benchmark	Max Profit	TVMS-LBPSO	TVMS-VLBPSO	HTBPSO-QI	TV-LBPSO	VBPSO8	BBA	BGSA	GB-ABC
mknapcb1–5.100–01	*Best*	**24274**	**24274**	23995	**24274**	23654	21676	23908	22638
	Mean	24094.2	**24152.7**	23736.3	24081.8	23392.2	21066.1	23608.8	21294.6
	Worst	23958	**24006**	23482	23936	22879	20249	23400	20051
	STD	95.3494	100	141.401	88.8717	246	389.064	163.682	756.281
mknapcb2–5.250–01	*Best*	60909	**61206**	59320	60539	58730	50219	59983	54021
	Mean	60655.1	**60999.4**	58287.6	60334.5	57360.6	49243.6	59005.7	52039.5
	Worst	60161	**60768**	57324	59968	56506	47604	58094	49877
	STD	209.342	**145.785**	785.449	189.925	666.892	854.219	670.519	1208.34
mknapcb3–5.500–01	*Best*	115575	**116821**	110053	115112	108056	94297	110796	99374
	Mean	115187	**116204**	106902	114496	105738	91818.5	110049	97627.8
	Worst	114568	**115645**	101456	113686	103008	89095	108078	95641
	STD	384.629	**375.255**	3425.23	412.261	1524.63	1629.6	865.162	1125.17
mknapcb4–10.100–01	*Best*	22601	**22680**	22241	22533	22363	20040	22431	21021
	Mean	22447.1	**22513.4**	21948.9	22388	21863	19561.8	22146	20192.7
	Worst	**22308**	22187	21630	22216	21541	19007	21810	18600
	STD	92.1695	148.366	186.17	95.4719	266.115	374.616	167.933	704.175
mknapcb5–10.250–01	*Best*	57763	**58143**	56554	57227	55544	48634	57226	51640
	Mean	57371.7	**57888.5**	55464.2	56840.4	54440	47488.8	56063.8	50285.3
	Worst	57050	**57616**	54225	56174	53449	46415	55396	48181
	STD	241.612	**153.42**	853.253	357.512	676.017	759.774	512.103	1068.87
mknapcb6–10.500–01	*Best*	115662	**117864**	111216	114793	107813	95471	111237	101223
	Mean	114895	**117128**	107540	113697	105537	92839.5	110271	99225.3
	Worst	114147	**116357**	102090	112514	102469	89508	108898	96574
	STD	465.911	**487.53**	2906.69	728.597	1497.92	1801.36	793.334	1686.4
mknapcb8–30.250–29	*Best*	148799	**149202**	147810	148686	147264	135977	148343	143403
	Mean	148364	**148964**	147001	148152	146363	134394	147843	142554
	Worst	148064	**148725**	145366	147469	145449	133036	147401	141848
	STD	238.646	**148.629**	677.949	432.738	651.346	1031.47	295.793	609.961
mknapcb9–30.500–29	*Best*	297916	**299270**	294223	296969	291779	251731	296625	284595
	Mean	296742	**298763**	292311	296296	289730	246956	294508	282328
	Worst	295388	**297580**	289601	295718	286855	243066	293749	280501
	STD	878.984	491.912	1403.31	395.098	1442.38	2730.51	840.627	1717.81

where the maximum capacity of each knapsack is W_j. n and m are the numbers of objects and knapsacks, respectively. w_i and p_i are the weight and the profit of the i^{th} item. x_i is one or zero and it shows whether the i^{th} item has been selected or not.

Meta-heuristic algorithms can be applied to solve the 0–1 MKP. In these algorithms, the population is randomly initialized by '0′ and '1′ values. Some solutions in the population are infeasible because their total weights are more than the knapsacks capacities. Hence, several methods have been proposed to improve the performance of these algorithms. The penalty function (PF) technique is one of them. For each infeasible solution, a penalty is computed to decrease the probability of choosing infeasible solutions. The following PF is applied to test the efficiency of proposed algorithm in the binary search space [9].

$$Penalty = \frac{\sum_{i=1}^{n} p_i x_i}{Q + Max_{j=1..m}\left(\sum_{i=1}^{n} w_{ji} x_i - W_j\right)}, \tag{44}$$

where Q is a positive constant, n and m are the numbers of items and knapsacks, respectively.

As shown in Table 7, fourteen 0–1 MKP benchmark instances have been selected from OR-Library [2] to evaluate the performance of proposed method. In this table, the best known is the maximum profit; n and m are the numbers of objects and knapsacks, respectively. The selected benchmarks are very difficult to be optimized.

All algorithms are run on 0–1 MKP benchmark instances and their results are illustrated in Table 8. The best, the worst, the mean and standard deviation of profits that are obtained by the algorithms are reported in this table. The average errors obtained by algorithms have been reported at the end of the table. The average error is calculated as follows [3]:

$$Aerage\ Error = \frac{1}{NS}\sum_{i=1}^{NS}\frac{t_i - y_i}{t_i} \times 100, \tag{45}$$

where NS is the number of benchmarks, and y_i is the best profit obtained by each algorithm on the i^{th} benchmark. t_i is the maximum profit of the i^{th} benchmark.

As shown in Table 8, TVMS-VLBPSO provides the best results and obtained the minimum error among the other algorithms. The second rank belongs to TVMS-LBPSO. These results show that the proposed transfer function has the best performance among the other transfer functions for BPSO. Furthermore, the proposed transfer function has considerably increased the performance of BPSO compared with other binary algorithms.

BBA shows the worst results compared to others. BBA applies the same transfer function of VBPSO8 but the results of tested benchmarks are much weaker than VBPSO8. This indicates that the weaknesses of algorithms in the continuous search space have a direct effect on the efficiency of algorithms in the binary search space.

The best results of TVMS-VLBPSO are concluded from Fig. 6. As seen in this figure, TVMS-VLBPSO finds the maximum profit faster than the other algorithms. It is noticeable that the complexity of TVMS-VLBPSO is similar to LBPSO. However, the proposed method has faster convergence rate and higher solution accuracy than the compared algorithms as shown in Fig. 6.

In the next experiment, the maximum number of iterations is increased to determine the effect of time on the quality of results. Some benchmarks from Table 7 have been chosen in the experiment. From each category, more complex benchmarks are selected and their results are demonstrated in Table 9. As shown in this table, the TVMS-VLBPSO is more powerful and more robust than the others in achieving the best results; even when the number of iterations is increased. In all cases, the mean profit obtained by TVMS-VLBPSO is better than the others. The second rank belongs to TVMS-LBPSO and BBA shows poor results in the experiment.

6. Conclusion

This study proposes a time-varying mirrored S-shaped (TVMS) transfer function to convert the continuous search space to the binary one in BPSO. The transfer function creates a good balance between exploration and exploitation in the binary search space. The role of transfer function is very important in enhancing the performance of binary algorithms. The proposed method applies two time-varying sigmoid functions that are mirrored for the positive and negative directions. This kind of transfer function enhances the exploration of algorithm in the first steps; therefore, the algorithm has a good search in the space. In the last steps, the algorithm switches from exploration to exploitation to search around better solutions. The proposed transfer function is easy to be implemented in all binary versions of BPSO without increasing the complexity of the algorithm. Also, it is not sensitive to the dimension of problem.

To evaluate the performance of TVMS_BPSO, the results of some well-known BPSO algorithms and binary swarm intelligence algorithms have been compared with the proposed method on CEC 2005 benchmark functions and 0–1 MKP benchmark instances. The experimental results show that the suggested transfer function is more efficient than the S-Shaped and V-shaped transfer functions in generating better quality solutions. As mentioned, the proposed transfer function can be applied in BPSO algorithms to enhance their performance in the binary search spaces and this claim was tested in this study. The transfer function has been employed for the local topologies of BPSO. The results indicate that the efficiency of LBPSO has been considerably improved by the transfer function. For further studies, the proposed transfer function can be applied in other meta-heuristic algorithms to evaluate the performance and to solve various discrete optimization problems.

Declaration of Competing Interest

None.

References

[1] J.C. Bansal, K. Deep, A modified binary particle swarm optimization for knapsack problems, Appl. Math. Comput. 218 (2012) 11042–11061.
[2] J.E. Beasley, OR-library: distributing test problems by electronic mail, J. Oper. Res. Soc. 41 (1990) 1069–1072, doi:10.1057/jors.1990.166.
[3] Z. Beheshti, S.M. Shamsuddin, S. Hasan, Memetic binary particle swarm optimization for discrete optimization problems, Inf. Sci. (Ny) 299 (2015) 58–84.
[4] Z. Beheshti, BMNABC: binary multi-neighborhood artificial bee colony for high-dimensional discrete optimization problems, Cybern. Syst. 49 (2018) 452–474, doi:10.1080/01969722.2018.1541597.
[5] Z. Beheshti, S.M. Shamsuddin, Non-parametric particle swarm optimization for global optimization, Appl. Soft Comput. 28 (2015) 345–359, doi:10.1016/j.asoc.2014.12.015.
[6] Z. Beheshti, S.M.H. Shamsuddin, CAPSO: centripetal accelerated particle swarm optimization, Inf. Sci. (Ny) 258 (2014) 54–79, doi:10.1016/j.ins.2013.08.015.
[7] Z. Beheshti, S.M. Shamsuddin, S. Sulaiman, Fusion global-local-topology particle swarm optimization for global optimization problems, Math. Probl. Eng. 2014 (2014) 1–19.
[8] Z. Beheshti, S.M. Shamsuddin, S. Hasan, N.E. Wong, Improved centripetal accelerated particle swarm optimization, Int. J. Adv. Soft Comput. Its Appl. 8 (2016) 1–26.
[9] Z. Beheshti, S.M. Shamsuddin, S.S. Yuhaniz, Binary accelerated particle swarm algorithm (BAPSA) for discrete optimization problems, J. Glob. Optim. 57 (2013) 549–573, doi:10.1007/s10898-012-0006-1.
[10] M.R. Bonyadi, Z. Michalewicz, Particle swarm optimization for single objective continuous space problems: a review, 25 (2017) 1–54.
[11] L. Cao, L. Xu, E.D. Goodman, A neighbor-based learning particle swarm optimizer with short-term and long-term memory for dynamic optimization problems, Inf. Sci. (Ny) 453 (2018) 463–485, doi:10.1016/j.ins.2018.04.056.
[12] R. Cheng, M. Yao, Particle swarm optimizer with time-varying parameters based on a novel operator, Appl. Math. Inf. Sci. 5 (2011) 33–38.
[13] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inf. Sci. (Ny) 291 (2015) 43–60, doi:10.1016/j.ins.2014.08.039.
[14] P.C. Chu, J.E. Beasley, A genetic algorithm for the multidimensional knapsack problem, J. Heuristics 4 (1998) 63–86, doi:10.1023/A:1009642405419.
[15] L.-Y. Chuang, S.-W. Tsai, C.-H. Yang, Improved binary particle swarm optimization using catfish effect for feature selection, Expert Syst. Appl. 38 (2011) 12699–12707, doi:10.1016/j.eswa.2011.04.057.
[16] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (2002) 58–73, doi:10.1109/4235.985692.
[17] R.C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, Evol. Comput. (2001) 81–86 Proc. 2001 Congr. 1 (2001), doi:10.1109/CEC.2001.934374.

[18] M.J. Islam, X. Li, Y. Mei, A time-varying transfer function for balancing the exploration and exploitation ability of a binary PSO, Appl. Soft Comput. 59 (2017) 182–196, doi:10.1016/j.asoc.2017.04.050.
[19] R. Jensi, G.W. Jiji, An enhanced particle swarm optimization with levy flight for global optimization, Appl. Soft Comput. 43 (2016) 248–261, doi:10.1016/j.asoc.2016.02.018.
[20] F. Jiang, H. Xia, Q.A. Tran, Q.M. Ha, N.Q. Tran, J. Hu, A new binary hybrid particle swarm optimization with wavelet mutation, Knowledge-Based Syst. 130 (2017) 90–101, doi:10.1016/j.knosys.2017.03.032.
[21] A.R. Jordehi, Binary particle swarm optimisation with quadratic transfer function: a new binary optimisation algorithm for optimal scheduling of appliances in smart homes, Appl. Soft Comput. 78 (2019) 465–480, doi:10.1016/j.asoc.2019.03.002.
[22] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proc. 1995 IEEE Int. Conf. Neural Networks, Perth, Australia, Piscataway, NJ, IEEE Service Center, 1995, pp. 1942–1948.
[23] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proc. IEEE Int. Conf. Syst. Man, Cybern., Washington, DC, USA, IEEE Computer Society, 1997, pp. 4104–4108.
[24] J. Kennedy, R. Mendes, Population structure and particle swarm performance, Evol. Comput. (2002) 1671–1676 CEC'02. Proc. 2002 Congr., 2002.
[25] M.S. Kiran, The continuous artificial bee colony algorithm for binary optimization, Appl. Soft Comput. 33 (2015) 15–23.
[26] S. Lee, S. Soak, S. Oh, W. Pedrycz, M. Jeon, Modified binary particle swarm optimization, Prog. Nat. Sci. 18 (2008) 1161–1166, doi:10.1016/j.pnsc.2008.03.018.
[27] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proc. 2005 IEEE Swarm Intell. Symp., SIS, 2005, pp. 124–129, doi:10.1109/SIS.2005.1501611. 2005., 2005.
[28] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (2006) 281–295, doi:10.1109/TEVC.2005.857610.
[29] G. Lin, J. Guan, A hybrid binary particle swarm optimization for the obnoxious p-median problem, Inf. Sci. (Ny) 425 (2018) 1–17, doi:10.1016/j.ins.2017.10.020.
[30] N. Lynn, P.N. Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, Swarm Evol. Comput. 24 (2015) 11–24, doi:10.1016/j.swevo.2015.05.002.
[31] Y. Marinakis, A. Migdalas, A. Sifaleras, A hybrid particle swarm optimization – variable neighborhood search algorithm for constrained shortest path problems, Eur. J. Oper. Res. 261 (2017) 819–834, doi:10.1016/j.ejor.2017.03.031.
[32] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (2004) 204–210.
[33] S. Mirjalili, A. Lewis, S-shaped versus V-shaped transfer functions for binary particle swarm optimization, Swarm Evol. Comput. 9 (2013) 1–14, doi:10.1016/j.swevo.2012.09.002.
[34] S. Mirjalili, S.M. Mirjalili, X.-S. Yang, Binary bat algorithm, Neural Comput. Appl. 25 (2014) 663–681, doi:10.1007/s00521-013-1525-5.
[35] H. Nezamabadi-pour, M. Maghfoori-Farsangi, Binary particle swarm optimization: challenges and new solutions, J. Comput. Soc. Iran Comput. Sci. Eng. 6 (2008) 21–32.
[36] C. Ozturk, E. Hancer, D. Karaboga, A novel binary artificial bee colony algorithm based on genetic operators, Inf. Sci. (Ny) 297 (2015) 154–170.
[37] K.E. Parsopoulos, UPSO: a unified particle swarm optimization scheme, Lect. Ser. Comput. Comput. Sci. 1 (2004) 868–873.
[38] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (2007) 33–57, doi:10.1007/s11721-007-0002-0.
[39] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, BGSA: binary gravitational search algorithm, Nat. Comput. 9 (2010) 727–745.
[40] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (2004) 240–255, doi:10.1109/TEVC.2004.826071.
[41] 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, Eur. J. Pharm. Sci. 22 (2004) 145–152.
[42] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proc. IEEE Congr. Evol. Comput., 1998, pp. 69–73.
[43] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, KanGAL Rep. 2005005 (2005) 2005.
[44] M. Taherkhani, R. Safabakhsh, A novel stability-based adaptive inertia weight for particle swarm optimization, Appl. Soft Comput. J. 38 (2016) 281–295, doi:10.1016/j.asoc.2015.10.004.
[45] M.R. Tanweer, S. Suresh, N. Sundararajan, Self regulating particle swarm optimization algorithm, Inf. Sci. (Ny). 294 (2015) 182–202, doi:10.1016/j.ins.2014.09.053.
[46] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft Comput. 22 (2018) 387–408, doi:10.1007/s00500-016-2474-6.
[47] L. Wang, X. Wang, J. Fu, L. Zhen, A novel probability binary particle swarm optimization algorithm and its application, J. Softw. 3 (2008) 28–35.
[48] F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, X.-L. Shen, A hybrid particle swarm optimization algorithm using adaptive learning strategy, Inf. Sci. (Ny). 436–437 (2018) 162–177, doi:10.1016/j.ins.2018.01.027.
[49] K. Zhang, Q. Huang, Y. Zhang, Enhancing comprehensive learning particle swarm optimization with local optima topology, Inf. Sci. (Ny) 471 (2019) 1–18, doi:10.1016/j.ins.2018.08.049.
[50] X. Zhang, X. Wang, Q. Kang, J. Cheng, Differential mutation and novel social learning particle swarm optimization algorithm, Inf. Sci. (Ny). 480 (2019) 109–129, doi:10.1016/j.ins.2018.12.030.