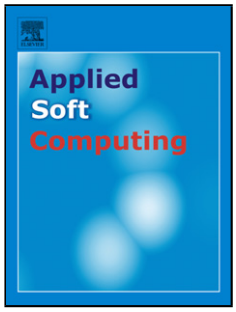


Title: A Time-Varying Transfer Function for Balancing the Exploration and Exploitation ability of a Binary PSO

Author: Md. Jakirul Islam Xiaodong Li Yi Mei

PII: S1568-4946(17)30232-6
DOI: http://dx.doi.org/doi:10.1016/j.asoc.2017.04.050
Reference: ASOC 4183

To appear in: *Applied Soft Computing*

Received date: 9-9-2016
Revised date: 28-2-2017
Accepted date: 24-4-2017

Please cite this article as: Md. Jakirul Islam, Xiaodong Li, Yi Mei, A Time-Varying Transfer Function for Balancing the Exploration and Exploitation ability of a Binary PSO, *<![CDATA[Applied Soft Computing Journal]]>* (2017), http://dx.doi.org/10.1016/j.asoc.2017.04.050

A Time-Varying Transfer Function for Balancing the Exploration and Exploitation ability of a Binary PSO

Md. Jakirul Islam[a,*], Xiaodong Li[a], Yi Mei[b]

[a] *School of Science, RMIT University, Melbourne, Australia*
[b] *School of Engineering and Computer Science, Victoria University of Wellington, New Zealand*

Abstract

Many real-world problems belong to the family of discrete optimization problems. Most of these problems are NP-hard and difficult to solve efficiently using classical linear and convex optimization methods. In addition, the computational difficulties of these optimization tasks increase rapidly with the increasing number of decision variables. A further difficulty can be also caused by the search space being intrinsically multimodal and non-convex. In such a case, it is more desirable to have an effective optimization method that can cope better with these problem characteristics. Binary Particle Swarm Optimization (BPSO) is a simple and effective discrete optimization method. The original BPSO and its variants have been used to solve a number of classic discrete optimization problems. However, it is reported that the original BPSO and its variants are unable to provide satisfactory results due to the use of inappropriate transfer functions. More specifically, these transfer functions are unable to provide BPSO a good balance between exploration and exploitation in the search space, limiting their performances. To overcome this problem, this paper proposes to employ a time-varying transfer function in the BPSO, namely TV_T-BPSO. To understand the search behaviour of the TV_T-BPSO, we provide a systematic analysis of its exploration and exploitation capability. Our experimental results demonstrate that TV_T-BPSO outperforms existing BPSO variants on both low-dimensional and high-dimensional classical 0-1 knapsack problems, as well as a 200-member

*Corresponding author
Email addresses: md.jakirul.islam@rmit.edu.au (Md. Jakirul Islam), xiaodong.li@rmit.edu.au (Xiaodong Li), yi.mei@ecs.vuw.ac.nz (Yi Mei)

truss problem, suggesting that TV_T-BPSO is able to better scale to high dimensional combinatorial problems than the existing BPSO variants and other metaheuristic algorithms.

Keywords: Discrete optimization problems, Binary particle swarm optimization, Transfer function, The 0-1 knapsack problem, The truss optimization problem

1. Introduction

Many real-world optimization problems are typically discrete problems. Classic examples include the 0-1 knapsack problem [1, 2], traveling salesman problem [3], scheduling problem [4], and optimal power flow problem [5]. It has been shown that these problems are NP-hard, i.e., the size of the solution space increases exponentially with the number of decision variables [6]. In addition, their search spaces are often intrinsically multi-modal and non-convex. Hence it is highly desirable to design optimization methods that can overcome these difficulties.

Since the past decade, many researchers have developed powerful optimization methods that are inspired by nature, often referred to as metaheuristics [7]. Binary particle swarm optimization (BPSO) is one popular metaheuristic algorithm first proposed by Kennedy and Eberhart [8]. This BPSO has been used for solving various types of discrete optimization problems [4, 9, 10, 11, 12, 13]. However, it has been observed that BPSO seems to be lacking of the exploration capability that is needed for obtaining high quality solutions [14, 15, 16]. Many BPSO variants have been developed to tackle this issue [14, 15, 16, 17, 18, 19, 20, 21, 22, 23]. However, the existing BPSO variants still have the issue of maintaining a good balance between exploration and exploitations, since too much exploration often degrades the fine-tuning ability of BPSO. On the other hand, too much exploitation gives more refining capability but it may adversely drive the search towards local optimal solutions.

In BPSO, the transfer function is considered as a key operator for controlling exploration and exploitation [14, 15]. Using an inappropriate transfer function can substantially degrade the performance of the BPSO. The sigmoid transfer function (S_T) typically employed in BPSO has several identified limitations [14, 15, 24]. Specifically, this transfer function is unable to provide BPSO a sufficient amount of diversity causing an imbalance between

exploration and exploitation.

In the past few years, two different approaches have been proposed to modify the original BPSO for improving its performance. The first approach focused on designing new rules to update the particle's velocity and position for BPSO. For example, in [25], the quantum-inspired BPSO (QBPSO) adopts a Q-bit individual for the probabilistic representation to replace the velocity update equation in BPSO. In [26], a new velocity update equation is proposed based on the observation of the personal best influence and initial velocity values of BPSO. Another modified BPSO namely Binary Accelerated Particle Swarm Algorithm (BAPSA) is proposed in [27]. In this modified BPSO, a new velocity update equation adopted which is designed based on Newtonians motion laws. Recently, Banka and Dara [28] proposed a hamming distance based binary particle swarm optimization (HDBPSO) for features election, classification and validation. In this BPSO, the hamming distance is used as a proximity measure for updating the velocity of particle(s) in binary PSO. In another improved BPSO [21], a new position update rule is used to enhance the performance of original BPSO for gene selection from microarray data. However, the above-mentioned BPSOs have several shortcomings, e.g., no strategy to tune the new parameters, computationally expensive, and relatively hard to implement [29, 30]. Therefore, they are not generally applicable to a wide range of discrete optimization problems.

In contrast, the second approach focused on replacing the sigmoid transfer function with new transfer functions in order to update each particle's position in a way to encourage a better exploration for the search space. The V-shaped and linear normalized transfer function is one such scheme proposed in [14, 15, 16]. It is reported that the V-shaped transfer function based BPSOs [15, 16] and linear normalized transfer function based BPSO [14] outperform the original BPSO [8] and some other well-known BPSOs [31, 32, 33] over a set of low-dimensional optimization problems [14, 34]. The reason is also reported in [14, 15, 16] that these transfer functions have the ability to promote more exploration compared to the conventional sigmoid transfer function.

To examine the general exploration and exploitation capability of the BPSO, it is necessary to analyze the performance of the existing transfer functions[14, 15, 16] over a diverse set of both low-dimensional and high-dimensional discrete test problems to see whether they can maintain a good balance between exploration and exploitation. In this context, we aim to do the followings which form the key contributions of this paper:

- We provide an analysis on the behaviour of BPSO employing existing transfer functions and identify their shortcomings in terms of balancing between exploration and exploitation.
- We propose and design a time-varying transfer function considering the shortcomings identified. The BPSO adopting the proposed time-varying transfer function is named as TV_T-BPSO.
- We validate the superiority of TV_T-BPSO by comparing it with the original BPSO and three existing BPSO variants on a diverse set of 0-1 knapsack benchmark instances and a truss design problem.

The rest of the paper is organized as follows: Section 2 provides some background information, including continuous PSO and the binary PSO (BPSO), followed by an analysis on the behaviour of existing transfer functions. Section 3 presents the details of the TV_T-BPSO, followed by Section 4 on a simple example to demonstrate the exploration and exploitation ability of TV_T-BPSO. Section 5 first presents some statistical results that compare TV_T-BPSO with four other well-known BPSOs over 0-1 knapsack test instances and a truss design problem. Empirical investigations on the influence of the parameter settings of TV_T-BPSO are also conducted. Finally, the concluding remarks are presented in Section 6.

2. Background

This section first introduces the PSO algorithm and its discrete version, the BPSO algorithm, and then examines the behaviour of BPSO employing existing transfer functions to identify their merits and shortcomings in balancing between exploration and exploitation.

2.1. Particle swarm optimization

In [35], the particle swarm optimization (PSO) is proposed to optimize the continuous optimization problems. In PSO, the particles (individuals of a swarm) are first initialized by placing them randomly in the d-dimensional search space and the search for an optimal solution is done by updating individual over successive iterations. In the iterative process, the i-th particle keeps two values in its memory; its personal best position $\mathbf{p}_i=(p_{i1},\ldots,p_{id})$ and the global best position of the whole swarm $\mathbf{g}=(g_1,\ldots,g_d)$. At each

iteration, the velocity $\mathbf{v}_i$=($v_{i1}, \ldots v_{id}$) of i-th particle of the swarm is modified by the following equation:

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_{1d}^{k}(p_{id}^{k} - x_{id}^{k}) + c_2 r_{2d}^{k}(g_{d}^{k} - x_{id}^{k}), \tag{1}$$

where $(k+1)$ represents the current iteration, w represents the inertia weight which is used to maintain a balance between exploration and exploitation, d represents the dimension, r_{1d} and r_{2d} are two random numbers drawn from the uniformly distribution $\mathcal{U}$(0,1), and $c_1 > 0$ and $c_2 > 0$ are the cognitive and social weights, respectively.

PSO offers several benefits: it is simple to implement, it has fast convergence, and computationally efficient [36]. Due to these benefits, PSO has been applied to various problem domains including antennas design [37], civil engineering [38], and medical diagnoses [39].

In practice, PSO often uses the problem-dependent velocity clamping techniques (e.g. [40]) to prevent too large velocities. More specifically, the velocity v_{id}^{k+1} is bounded by a threshold v_{max} as follows:

$$v_{id}^{k+1} = \begin{cases} v_{max}, & \text{if } v_{id}^{k+1} > v_{max}, \\ -v_{max}, & \text{if } v_{id}^{k+1} < v_{max}. \end{cases} \tag{2}$$

Once the velocity has been clamped, the position x_{id} of i-th particle is modified according to the following equation:

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{3}$$

Up to now, PSO and most of its variants have been developed mostly for handling continuous optimization problems. These PSO models cannot be directly used for the discrete optimization problems.

2.2. Binary particle swarm optimization

Kennedy and Eberhart [8] developed the first BPSO to tackle the binary optimization problems. For clarity, we call the original BPSO as S_T-BPSO in this paper, to differentiate it from other BPSO variants. The S_T-BPSO primarily extended the basic concept of the original PSO by using the sigmoid transfer function to transform the value of velocity from the continuous space into binary space. In this BPSO, at each iteration, the velocity v_{id} is modified

according to Eq. (1) and Eq. (2) with $w = 1$ and the position x_{id} is modified according to the following:

$$x_{id}^{k+1} = \begin{cases} 0 & \text{if } rand() \geq S_T(v_{id}^{k+1}), \\ 1 & \text{otherwise,} \end{cases} \tag{4}$$

where $S_T(v_{id}^{k+1})$ is a sigmoid transfer function which denotes the probability (in the range of [0,1]) for a bit that takes the value 0 or 1:

$$S_T(v_{id}^{k+1}) = \frac{1}{1 + e^{-v_{id}^{k+1}}}. \tag{5}$$

It is reported that BPSO has the difficulties to provide a sufficient amount of exploration, because of the conventional sigmoid transfer function [17, 24, 31, 32, 41]. To overcome this, the modified BPSO with a linear normalized and two different V-Shaped transfer functions were proposed in literature [14, 15, 16].

In the followings, we will provide a brief analysis of the behavior of sigmoid function, linear normalized function, and two different V-shaped functions to demonstrate that they have the difficulties to maintain a good balance between exploration and exploitation. However, before providing this analysis, it is important to describe the basics of two key phases of a typical BPSO search process, i.e., the exploration phase and exploitation phase.

2.3. Exploration and exploitation phase of BPSO

In BPSO, the search space is considered as a hypercube, in which a particle moves from one node to another by flipping one or more bits of its position vector $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$. BPSO uses a transfer function to determine the probability of the value of each bit x_{id} (0 or 1), which depends on the corresponding velocity. If the absolute velocity is high, then the corresponding bit will have a very high probability to be 1 (positive velocity) or 0 (negative velocity). On the other hand, if the velocity is close to zero, then the value of the corresponding bit becomes uncertain. In other words, in BPSO, a large absolute velocity leads to exploitation, and a small absolute (close to zero) velocity leads to exploration.

The exploration and exploitation phase can be determined for the binary search space by measuring the hamming distance [42] between the previous

position $\mathbf{x}_i^k$ and the current position $\mathbf{x}_i^{k+1}$ of the i-th particle of BPSO. The equation for measuring the hamming distance is given below:

$$dist_i = \sum_{j=1}^{d} |x_{i,j}^k - x_{i,j}^{k+1}|. \quad (6)$$

According to the Eq. (6), if the measured distance $dist_i$ is found to be sufficiently large, then it can be said that BPSO is exploring the search space, otherwise, it is exploiting the search space.

2.4. Analysis of the behaviour of existing transfer functions

In literature, three different transfer functions can be found for BPSO, namely the *sigmoid transfer function* [8], *linear normalized transfer function* [14], and *V-shaped transfer function* [16, 15]. The behaviour of these transfer functions on the performance of BPSO are examined in the following sections.

2.4.1. Sigmoid transfer function

In [8], the sigmoid transfer function (see Fig. 1) is introduced to map a real-valued velocity v_{id} to a probability value in the range of [0,1] for changing a binary position x_{id}. According to Fig. 1, the sigmoid transfer function provides a higher bit flipping probability with the smaller absolute value of

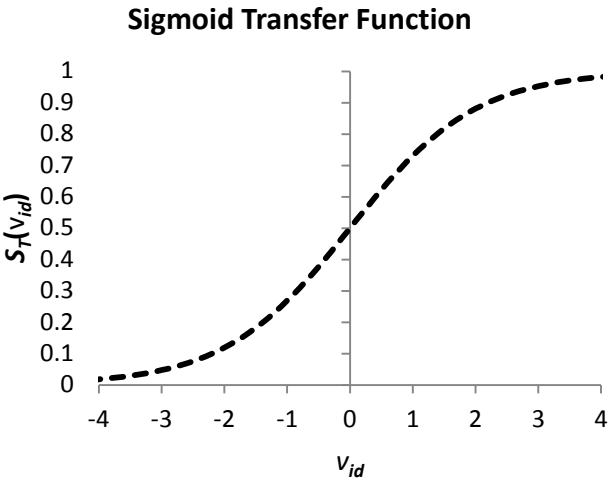


Figure 1: Illustration of the sigmoid transfer function (S_T).

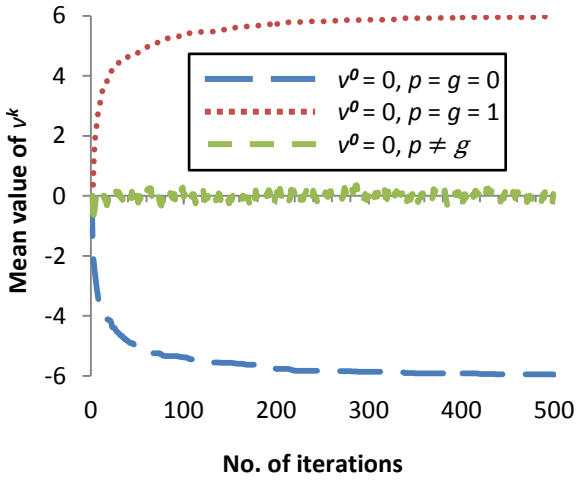
velocity and a lower bit flipping probability with a higher value of velocity. According to this, a small absolute value of velocity (close to zero) is preferable to better explore the search space in the early stages of the run. On the other hand, a large absolute value of velocity is preferable to better exploit the search space in the final stages of the run.

According to Eq. (1), the velocity v_{id}^{k+1} takes a smaller value when $p_{id}^k \neq g_d^k$, and takes a larger value when $p_{id}^k = g_d^k$. Since both p_{id} and g_d controls the velocity of i-th particle, S_T-BPSO cannot always maintain a smaller velocity in the early stages of the run and a higher velocity in the final stages of the run. Consequently, the sigmoid transfer function has the difficulties in maintaining a good balance between exploration and exploitation for S_T-BPSO.

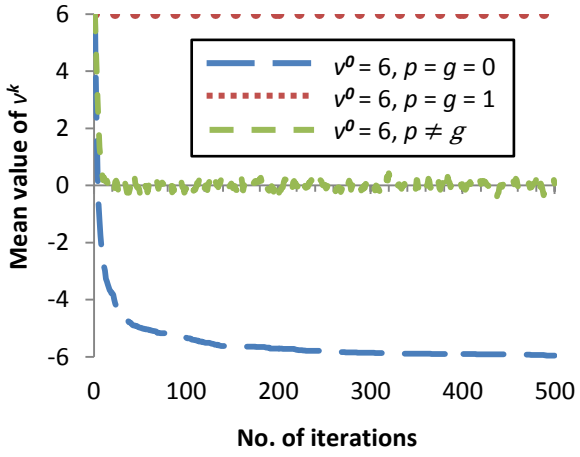
To validate the above analysis, we conduct an experiment to predict the sequences of the mean value of velocity v_{id}^{k+1} of S_T-BPSO over 30 independent runs (500 iterations per run) under different p_{id}^k and g_d^k values which are unchanged over time. For the sake of simplicity, this experiment has been conducted using a basic BPSO model where a single particle and a single dimension is considered. For this experiment, the velocity is updated using Eq. (1) and Eq. (2) where i has been set to 1 for the single particle, d has been set to 1 for the single dimension. The other parameters have been set to their standard value as $w = 1$, $c_1 = c_2 = 2$ and $v_{max} = 6$. The position of i-th particle is updated using Eq. (5) and Eq. (4), respectively.

With the above settings, we can demonstrate the sequences of the mean value of velocity in two extreme scenarios: 1) when the velocity takes a small initial value v^0=0; and 2) when it takes a large initial value v^0=v_{max}=6, as shown in Fig. 2. By considering the curves in this figure, if we assume that the search process is in the initial stages of the run, S_T-BPSO is expected to require a stronger exploration, i.e., a higher bit flipping probability to explore the search space. In this case, if $p \neq g$, then S_T-BPSO provides a small velocity value, from which the sigmoid transfer function can produce a stronger exploration for BPSO. However, if $p = g$, then the sigmoid transfer function cannot provide a stronger exploration due to the higher value of the velocity, as demonstrated in Fig. 2. As a result, most of the promising regions of the search space will remain to be unexplored; there is a higher possibility that S_T-BPSO may get trapped in local optima.

Now, if we assume that the search process is in the final stage of the run; S_T-BPSO would need a stronger exploitation, i.e., a lower bit flipping probability to improve the quality of the found solutions. According to Fig.



(a)



(b)

Figure 2: The curves of the mean value of velocity v^{k+1} of S_T-BPSO over 30 independent runs, with w=1, $c_1 = c_2 = 2$, and $v_{max} = 6$, when (a) v^0=0 and (b) v^0=v_{max}=6

2, if $p = g$, then the sigmoid transfer function will provide a low bit flipping probability (because $v^{k+1} = 6$ and $S_T(v^{k+1}) \simeq 0$) for exploiting the search space. However, if $p \neq g$, S_T will provide a high bit flipping probability (because $v^{k+1} \simeq 0$ and $S_T(v^{k+1}) \simeq 0.5$) instead of a low bit flipping probability

for S_T-BPSO. In this case, there is a higher chance that S_T-BPSO would lose the good solutions which have been found in the exploration phase.

Given the above discussion, it can be said that S_T-BPSO cannot maintain a good balance between exploration and exploitation due to the limitation of the sigmoid transfer function.

2.4.2. Linear normalized transfer function

Bansal, *et al.* [14] proposed a modified BPSO termed as L_T-BPSO where a linear normalized transfer function (L_T) is used to improve the exploration ability of S_T-BPSO. The linear normalize transfer function of L_T-BPSO is illustrated in Fig. 3 and mathematically defined by the following:

$$L_T(x_{id}^k, v_{id}^{k+1}) = \frac{x_{id}^k + v_{id}^{k+1} + v_{max}}{1 + 2v_{max}}. \tag{7}$$

In L_T-BPSO, the velocity of i-th particle is modified according to Eq. (1) and Eq. (2) with $w = 1$. This velocity is then transformed into probability using Eq. (7), of which the position of i-th particle is updated as follows:

$$x_{id}^{k+1} = \begin{cases} 0 & \text{if } rand() \geq L_T(x_{id}^k, v_{id}^{k+1}), \\ 1 & \text{otherwise.} \end{cases} \tag{8}$$

It can be observed that L_T-BPSO and S_T-BPSO uses the same velocity and position updating rules, except the transfer function for updating the particle position. Therefore, L_T-BPSO should experience the same difficulties as S_T-BPSO as described earlier in Section 2.4.1. However, the only difference between these BPSOs is that the linear normalized transfer function can provide a better exploration compared to the sigmoid transfer function. For example, if v_{id}=-3 and x_{id}=0, then according to the Eq. (5) and Fig. 4, there is a probability of 0.047 that x_{id} will be flipped from 0 to 1. Now according to the Eq. (6), the probability of flipping x_{id} from 0 to 1 is 0.111 which is greater than 0.047. With the higher bit flipping probability, there is a higher chance that the bits of x_{id} will be flipped in the next iteration.

2.4.3. V-shaped transfer functions

Two different V-shaped transfer functions were proposed in literature [16, 15]. The first V-shaped transfer function is termed as V_{T1} [16] and the second one V_{T2} [15], as shown in Fig. 5, and the corresponding modified BPSOs are V_{T1}-BPSO and V_{T2}-BPSO.

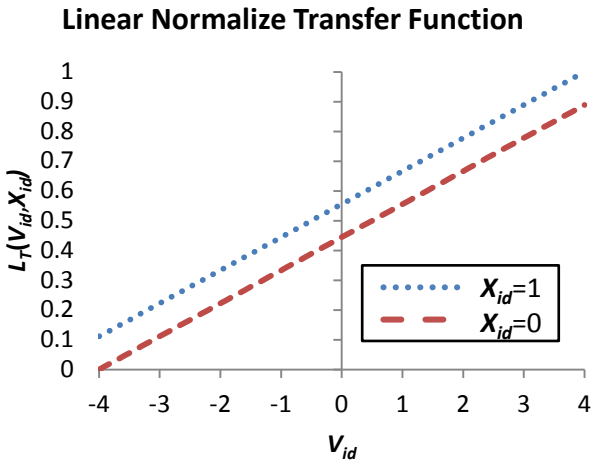


Figure 3: Illustration of the linear normalized transfer function (L_T).

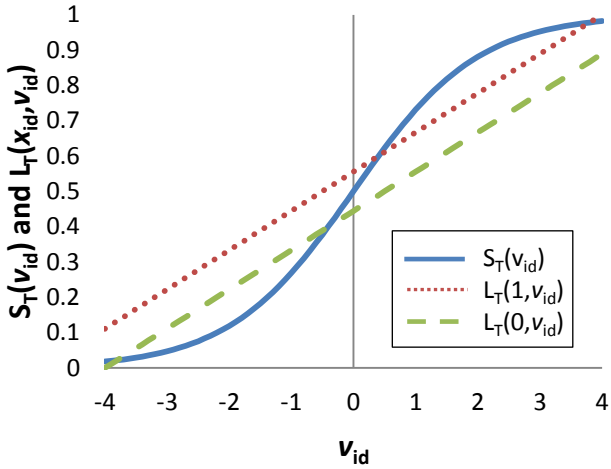


Figure 4: Comparison between the sigmoid and linear normalized transfer function.

Below we provide some explanation on why these two V-shaped transfer functions are unable to produce a good balance between exploration and exploitation.

1) V-shape transfer function V_{T1}: The first V-shaped transfer function V_{T1} (as

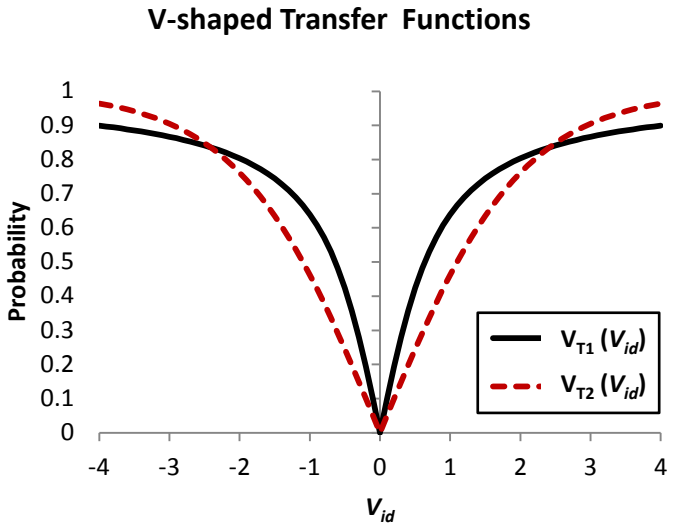


Figure 5: Illustration of two different V-shaped transfer functions V_{T1} and V_{T2}.

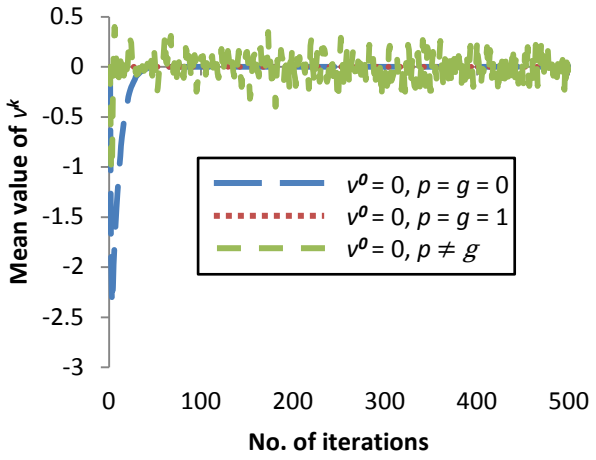
proposed for V_{T1}-BPSO) transforms a particle's velocity to a binary position using the following equation:

$$V_{T1}(v_{id}^{k+1}) = \begin{cases} 1 - \frac{2}{1+e^{-v_{id}^{k+1}}} & \text{if } v_{id}^{k+1} \leq 0, \\ \frac{2}{1+e^{-v_{id}^{k+1}}} - 1 & \text{otherwise.} \end{cases} \tag{9}$$

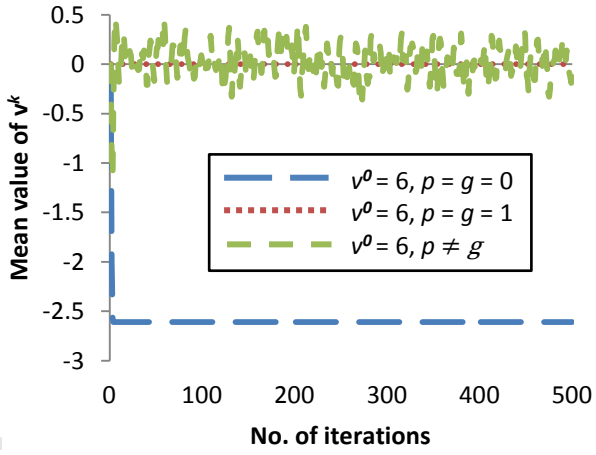
In V_{T1}-BPSO, Eq. (1) and Eq. (2) are first used to update the velocity of the i-th particle of the swarm. Eq. (9) is then used to transform the velocity into a probability value, by which the position of i-th particle is updated using the following:

$$x_{id}^{k+1} = \begin{cases} 0 & \text{if } rand() \leq V_{T1}(v_{id}^{k+1}) \text{ and } v_{id}^{k+1} \leq 0, \\ 1 & \text{if } rand() \leq V_{T1}(v_{id}^{k+1}) \text{ and } v_{id}^{k+1} > 0, \\ x_{id}^{k} & \text{if } rand() > V_{T1}(v_{id}^{k+1}). \end{cases} \tag{10}$$

Like S_T-BPSO and L_T-BPSO, V_{T1}-BPSO is also unable to make a good balance between exploration and exploitation either. This can be seen by the curves of the mean velocity values in Fig. 6. Here we ran these experiments using the similar parameter settings as in Fig. 2, except that the inertia



(a)



(b)

Figure 6: The curves of the mean value of velocity v^k of V_{T1}-BPSO over 30 independent runs under different p and g values with w_{min}=0.4, w_{max}=0.9, and c_1=c_2=2, when (a) v^0=0 and (b) v^0=6, respectively.

weight w was linearly decreased from 0.9 to 0.4, in order to obtain the best performance for V_{T1}-BPSO.

Fig. 6 shows two different situations of the mean velocity value v^k of V_{T1}-BPSO. First, if $p \neq g$, then v^k fluctuates in between -0.35 and +0.35.

With these velocity values, V_{T1} can produce a 16% bit flipping probability which is insufficient to explore the search space in the early stages of the run. In this situation, V_{T1}-BPSO can easily get stuck in a local optimum due to the lack of exploration. Second, if $p = g$, then v^k takes the value either 0 or -2.5. With these velocity values, V_{T1}-BPSO forces x^k to take the value 0, because v^k satisfies the first condition of Eq. (10). This situation also prevents V_{T1}-BPSO from exploring the search space, and the probability of getting trapped in the local optima is increased.

2) V-shape transfer function V_{T2}: This transfer function is used in V_{T2}-BPSO. Here, a tangent hyperbolic function is employed to transform a real-valued velocity into a probability value:

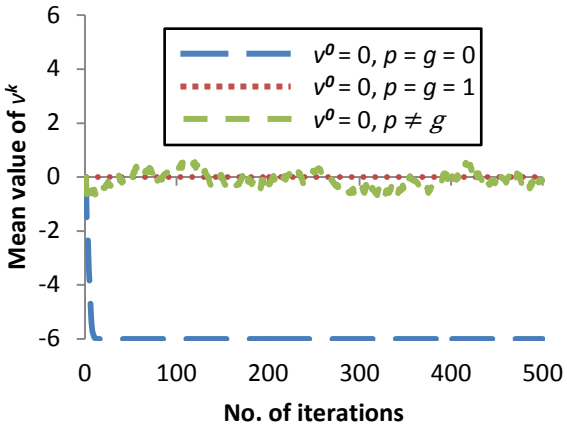
$$V_{T2}(v_{id}^{k+1}) = \left| \frac{2}{\pi} \arctan(\frac{\pi}{2} v_{id}^{k+1}) \right|. \tag{11}$$

Like V_{T1}-BPSO, V_{T2}-BPSO modifies the velocity of i-th particle according to Eq. (1) and Eq. (2). Once the velocity is updated, Eq. (11) is used to transform this velocity into probability, which in turn updates the i-th particle's position according to the following:

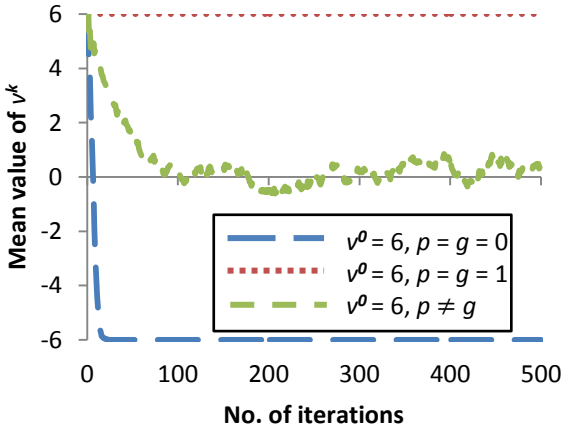
$$x_{id}^{k+1} = \begin{cases} (x_{id}^k)^{-1} & \text{if } rand() < V_{T2}(v_{id}^{k+1}), \\ x_{id}^k & \text{otherwise.} \end{cases} \tag{12}$$

To understand the search behaviour of V_{T2}-BPSO, we conduct another experiment to show the curves of the mean velocity value v^k in Fig. 7 (similar to Fig. 5). Fig. 7a shows that if p=g=1 and $p \neq g$, then v^k takes the value of either 0 or close to 0. With these values, V_{T2} produces a 0% bit flipping probability for this modified BPSO, and thus in the next iteration, x^k will not change its position. It is clear that V_{T2}-BPSO cannot provide the exploration ability as needed in the early stages of the run.

Fig. 7b also shows that v^k reaches to the maximum limit when p=g. In this case, V_{T2} produces the highest bit flipping probability according to the Eq. (11). As we know that a higher bit flipping probability provides a stronger exploration which is good for V_{T2}-BPSO to explore the search space in the early stages of the run. However, this higher bit flipping probability is not desirable for V_{T2}-BPSO, since it prevents the exploitation of the search space in the final stages of the run.



(a)



(b)

Figure 7: The curves of the mean velocity v^k of V_{T2}-BPSO over 30 independent runs under different p and g values with w_{min}=0.4, w_{max}=0.9, and c_1=c_2=2, when (a) v^0=0 and (b) v^0=6, respectively.

Considering the above discussion in this section, clearly there is a mismatch between the velocity update equation and transfer function, resulting in the above mentioned BPSOs being unable to maintain a good balance between exploration and exploitation. To overcome this problem, this paper

proposes a new transfer function employing a time-varying scheme in order to provide a better balance between the exploration and exploitation during the run of a BPSO.

3. Time-varying transfer function based BPSO (TV_T-BPSO)

3.1. Design considerations for the time-varying transfer function

Generally speaking, in the early stages of the run, an optimization algorithm is expected to focus more on exploration to avoid being trapped in local optima, but in the later stages of the run, the algorithm needs to switch to emphasizing more on exploitation to refine the solution quality. Following this intuition, we design a dynamic transfer function for the proposed TV_T-BPSO with the following considerations:

- In the early stages of the run, the transfer function should provide a high probability of flipping all the bits of x_{id} at any value of the velocity v_{id} so that the BPSO can provide a stronger exploration.
- In the intermediate stages of the run, the BPSO should start shifting from exploration to exploitation. This can be achieved by using a transfer function able to decrease the probability of flipping all the bits of x_{id} at any value of velocity v_{id} over iterations.
- In the final stages of the run, the transfer function should provide a low probability of flipping all the bits of x_{id} at any value of velocity v_{id} so that the BPSO can provide a stronger exploitation capability.

We apply the above concepts by adopting a new control parameter φ in the sigmoid transfer function as given in Eq. (5). This new transfer function is given as the following:

$$TV_T(v_{id}^{k+1}, \varphi) = \frac{1}{1 + e^{-\frac{v_{id}^{k+1}}{\varphi}}}, \tag{13}$$

where v_{id}^{k+1} is the velocity of the i-th particle at $(k+1)$-th iteration.

Eq.(13) can be used as a transfer function for the proposed TV_T-BPSO. However, instead of using a fixed value for the control parameter φ, we start with a larger value for φ and gradually decrease it as the run progresses,

in order to shift smoothly from more exploration to exploitation over time. This is achieved by the following:

$$\varphi = \varphi_{max} - Itr_{k+1} * \left(\frac{\varphi_{max} - \varphi_{min}}{Itr_{max}} \right), \tag{14}$$

where φ_{max} and φ_{min} are the bounds on the control parameter φ, Itr_{max} is the maximum number of iterations, and Itr_{k+1} is the current iteration, where $k = 0, 1, 2, ..., Itr_{max} - 1$.

We call the proposed transfer function as the time-varying transfer function (TV_T) because the shape of this transfer function changes over time depending on the different value of φ, as illustrated in Fig. 8. The curve $TV_T(v_{id}, \varphi_{max})$ represents the initial shape of the proposed transfer function obtained by setting φ=4. Similarly, the curve $TV_T(v_{id}, \varphi_{min})$ represents the final shape of the proposed transfer function obtained by setting φ=0.05. Other curves are obtained by the linearly decreasing values of φ.

As can be seen in Fig. 8, the curve $TV_T(v_{id}, \varphi_{max})$ linearly increases as the velocity value increases, which is much slower than the other curves. It

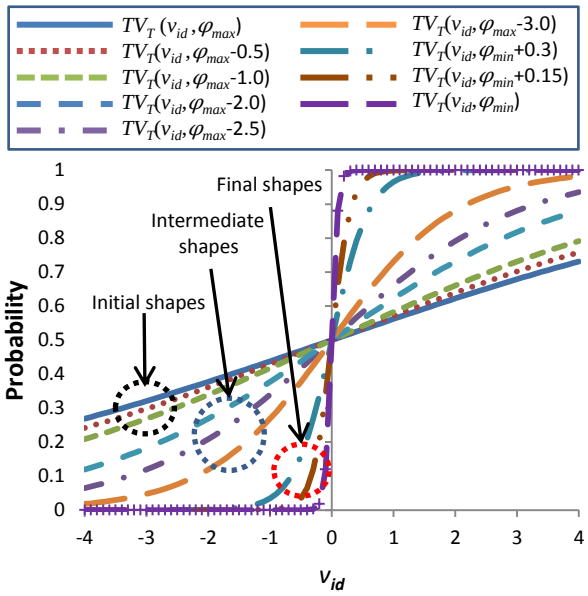


Figure 8: An illustration of different shapes of the time-varying transfer function (Eq. (13)) with different values of the control parameter φ.

should be noted that the probability of flipping the bits of position x_{id} increases as the slope of the curve becomes steeper. For each given velocity value, the curve $TV_T(v_{id}, \varphi_{max})$ provides the highest amount of bit flipping probability, because the curve is the closest to the probability value of 0.5 than any other curves. On the other hand, $TV_T(v_{id}, \varphi_{min})$ provides the lowest amount of bit flipping probability for changing the position x_{id} of i-th particle. Based on this observation, we propose TV_T-BPSO to adopt curves $TV_T(v_{id}, \varphi_{max})$ to $TV_T(v_{id}, \varphi_{max}\text{-}1.0)$ at the start of a run in order to provide a stronger exploration; $TV_T(v_{id}, \varphi_{max}\text{-}2.0)$ to $TV_T(v_{id}, \varphi_{max}\text{-}3.0)$ in the intermediate stage of the run to provide a moderate level of exploration; and towards the final stage of the run $TV_T(v_{id}, \varphi_{min}+0.3)$ to $TV_T(v_{id}, \varphi_{min})$ to have a stronger exploitation.

3.2. The proposed TV_T-BPSO

Like S_T-BPSO, TV_T-BPSO uses Eq. (1) for velocity update of each particle. However, for simplicity, we set the value of w=1. TV_T-BPSO uses the new time-varying transfer function Eq. (13) instead of the sigmoid transfer function in Eq. (5) for updating the position of i-th particle:

$$x_{id}^{k+1} = \begin{cases} 1 & \text{if } rand() < TV_T(v_{id}^{k+1}, \varphi), \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

The pseudo-code of the proposed TV_T-BPSO is provided in Alg. 1.

4. Behaviour analysis of TV_T-BPSO

This section provides an example to illustrate the differences between TV_T-BPSO and two well-known BPSOs, by considering their exploration and exploitation behaviours using a decision variable vector of 4-binary bits.

4.1. Exploration

Let us consider an example such that at k-th iteration of the early stages of run, the i-th particle $\mathbf{x}_i^k$ and $\mathbf{p}_i^k$ with 4-binary bits is (0010), the current $\mathbf{g}^k$ is (0011), and at the $(k+1)$-th iteration, the velocity corresponding to $\mathbf{x}_i^k$ is $\mathbf{v}_i^{k+1}$={-3.5, -3.8, 3.2, -0.1}. Note that at the $(k+1)$-th iteration, if $p_{id}^k = g_d^k$, then BPSO provides a higher value for v_{id}^{k+1}, otherwise, it provides a lower value for v_{id}^{k+1} (according to Fig. 2). For example, three higher values could have been chosen for the first, second, and third bit of $\mathbf{x}_i^k$ and one lower

Algorithm 1 Pseudocode of the proposed TV_T-BPSO algorithm.

Require: v_{max} ▷ the upper bound of velocity
Require: $iter_{max}$ ▷ the maximum number of iterations
Require: $popSize$ ▷ the population size

```
//initialization
iter = 0;
//randomly initialize i-th particle velocity, position, and personal best
for i=1 to popSize do
    x_i ∈ {-v_max, v_max}
    x_i ∈ {0, 1};
    p_i ← x_i;
end for
//main loop
repeat
    for i=1 to popSize do
        evaluate fitness f(x_i);
        if f(x_i) > f(p_i) then
            p_i ← x_i;
        end if
        if f(x_i) > f(g) then
            g ← x_i; f(g) ← f(x_i);
        end if
        update v_i using Eq. (1) and Eq. (2);
        calculate φ using Eq. (14);
        calculate TV_T(v_i, φ) using Eq. (13);
        update x_i using Eq. (15);
    end for
    iter = iter+1;
until iter < iter_max;
```

value for the fourth bit, giving us $\mathbf{v}_i^{k+1}$={-3.5, -3.8, 3.2, -0.1}. Note that the maximum velocity bound v_{max}=4 is considered here for BPSO. Based on this $\mathbf{v}_i^{k+1}$, we can compute the new position $\mathbf{x}_i^{k+1}$ using Eq. (5) and Eq. (4) for S_T-BPSO and using Eq. (13) and Eq. (15) for TV_T-BPSO, respectively.

For S_T-BPSO, we can compute the probabilities of flipping the first, second, third, and fourth bit of $\mathbf{x}_i^k$ using Eq. (5), which are 0.029, 0.021, 0.039, and 0.475, respectively. It is obvious that with these probabilities, only the fourth bit of $\mathbf{x}_i^k$ is highly likely to be flipped in the $(k+1)$-th iteration of the run, resulting in S_T-BPSO updating $\mathbf{x}_i^k$ from (0010) to (0011) using Eq. (4).

For TV_T-BPSO, let us assume that the curve $TV_T(v_{id}, \varphi_{max}\text{-}0.5)$ of Fig. 8 is used for determining the bit flipping probabilities of $\mathbf{x}_i^k$. From this curve, we can compute the bit flipping probabilities of the first, second, third, and fourth bit of $\mathbf{x}_i^k$, which are approximately 0.331, 0.318, 0.345 and 0.495, respectively. It can be noted that with the same $\mathbf{v}_i^{k+1}$, the bit flipping probabilities obtained by TV_T-BPSO are much higher than the bit flipping probabilities obtained by S_T-BPSO. In this case, there is a much higher chance that all of the bits of $\mathbf{x}_i^k$ of TV_T-BPSO will be flipped at $(k+1)$-th iteration, leading to the value of $\mathbf{x}_i^k$ being updated from (0010) to (1101).

The hamming distances between $\mathbf{x}_i^k$ and $\mathbf{x}_i^{k+1}$ are 1 and 4 respectively for the above two BPSOs according to Eq. (6). In this case, TV_T-BPSO resulting in a larger hamming distance suggesting that it has a stronger exploration capability than S_T-BPSO which only results in a smaller hamming distance.

4.2. Exploitation

Let us consider another example such that at the k-th iteration of the final stages of run, the i-th particle $\mathbf{x}_i^k$ and $\mathbf{p}_i^k$ with 4-binary bits is (0101), the current $\mathbf{g}^k$ is (0111), and at the $(k+1)$-th iteration, the velocity $\mathbf{v}_i^{k+1}$={-4.0, 4.0, -0.5, 4.0} has been chosen the same way as in the previous section. We can compute x_{id}^{k+1} using Eq. (11) and Eq. (12) for V_{T2}-BPSO and using Eq. (13) and Eq. (15) for TV_T-BPSO, respectively.

In case of V_{T2}-BPSO, we can compute the probability of flipping the first, second, third and, fourth bit of $\mathbf{x}_i^k$ using Eq. (11), which are 0.899, 0.899, 0.359, and 0.899. With these higher probabilities, there is a higher chance that the first, second, and fourth bit of $\mathbf{x}_i^k$ be flipped at $(k+1)$-th iteration, resulting in V_{T2}-BPSO updating $\mathbf{x}_i^k$ from (0101) to (1000) using Eq. (12).

In case of TV_T-BPSO, let us assume that the curve $TV_T(v_{id}, \varphi_{min})$ of Fig. 8 is used to determine the bit flipping probabilities of $\mathbf{x}_i^k$. From this curve, we can compute the bit flipping probability of four different bits of $\mathbf{x}_i^k$, which in this case are all zeros. With the zero probabilities, $\mathbf{x}_i$ will remain unchanged at the $(k+1)$-th iteration, resulting in $\mathbf{x}_i^{k+1}$ being (0101).

The hamming distances between $\mathbf{x}_i^k$ and $\mathbf{x}_i^{k+1}$ are 3 and 0 respectively for the above two BPSOs according to Eq. (6). In this case, TV_T-BPSO resulting in a smaller hamming distance suggests that it has a stronger exploitation capability than V_{T2}-BPSO.

5. Experimental studies

At the beginning, we perform there different experiments on the deterministic and non-deterministic 0-1 knapsack benchmark instances, which are commonly used benchmark problems for testing binary optimization algorithms. The purpose of the first experiment is to identify the best values for the three parameters of TV_T-BPSO, namely φ, m, and the velocity bound v_{max}. The second experiment is used to compare TV_T-BPSO with S_T-BPSO [8], L_T-BPSO [14], V_{T1}-BPSO [16], and V_{T2}-BPSO [15] over the low-dimensional ($4 \leq \mathrm{D} \leq 500$) 0-1 knapsack instances. The third experiment is carried out to further investigate the performance of these five BPSOs over the high-dimensional ($1000 \leq \mathrm{D} \leq 5000$) 0-1 knapsack instances. At the end, another experimental study is carried out to show the performance of TV_T-BPSO in solving high-dimensional truss optimization problem.

5.1. The 0-1 knapsack problem

The 0-1 knapsack problem is selected to test the five different BPSOs which can be described using the following example. Suppose we have n items and each item i has a weight w_i and a profit p_i, where $i = 1, 2, \ldots, n$. The goal is to select a subset of items in order to maximize the total profit such that the total weight of the selected items does not exceed the knapsack capacity C. To model this problem, x_i is introduced as a decision variable for each item. If the i-th item is selected then the value of x_i is set to 1, otherwise its value is set to 0. Formally, the 0-1 knapsack problem can be expressed as:

$$\text{maximize} \qquad \sum_{i=1}^{n} p_i x_i \tag{16}$$

$$\text{subject to} \qquad \sum_{i=1}^{n} w_i x_i \leq C, \tag{17}$$

$$\textit{where} \qquad x_i \in \{0, 1\},\ i = 1, ..., n. \tag{18}$$

In general, there are two groups of instances for evaluating the performance of the discrete metaheuristics algorithms, *deterministic instances* [1, 14] and *non-deterministic instances* [2]. The instances in the first group

are the low-dimensional instances ($4 \geq D \leq 24$) and their optimal solutions are available in literature [1, 14]. In contrast, the instances of the second group are the variable dimensional instances [2]. Since these instances are generated randomly, the optimal solutions of these instances are unknown. For this *non-deterministic instances* group, mainly four types of test instances have been identified, by taking into account the correlation between the profits and weights of the items [2]: *uncorrelated instances*, *weakly correlated instances*, *strongly correlated instances*, and *inversely strongly correlated instances*. According to [2], the generation procedure of these four types is provided below by considering a variable $R \in \mathbb{N}$: R is a positive number.

- *Uncorrelated instance (UCI)*: For this type, the weight and profit of an item i are selected as: $w_i \in \mathcal{U}[1, R]$ and $p_i \in \mathcal{U}[1, R]$, respectively.
- *Weakly correlated instance (WCI)*: For this type, the weight of the i-th item is specified as: $w_i \in \mathcal{U}[1, R]$ and the profit of that item is specified as: $p_i \in \mathcal{U}[w_i - R/10, w_i + R/10]$ subject to $p_i \geq 1$.
- *Strongly correlated instance (SCI)*: The weight and profit of the i-th item of the SCI instances are selected as: $w_i \in \mathcal{U}[1, R]$ and $p_i = w_i + R/10$, respectively.
- *Inversely strongly correlated instance (ISCI)*: For this type, first, the profit of an item i is selected as: $p_i \in \mathcal{U}[1, R]$ and then the weight of the same item is selected as: $w_i = p_i + R/10$.

For each instance type, the knapsack capacity C is set to a certain percentage (S) of the sum of weights as:

$$C = S \times \sum_{i=1}^{n} w_i, \tag{19}$$

5.2. Selecting the best values for m, v_{max}, and φ of TV_T-BPSO

In this section, an experiment is conducted to find the suitable values for the three parameters m, v_{max}, and φ. Three representative deterministic 0-1 knapsack instances Ks_8a, Ks_16b, and Ks_24d have been selected from [14]. And three non-deterministic 0-1 knapsack instances WCI_{100}, UCI_{500}, and $ISCI_{1000}$ have been selected, from those generated using the procedure described in Section 5.1 where the value for R is set to 1000, S is set to 0.5,

and the value for n is set to 100, 500, and 1000, respectively. This experiment has been conducted with the values of m=20 to 50, v_{max}=1 to 16, φ=1 to 5, and c_1=c_2=2. The maximum number of iterations is set to 1000 for the subsequent experiments, which is sufficient to reach competitive results for the low-dimensional as well as the high-dimensional 0-1 knapsack instances of this paper.

Table 1 shows the results on TV_T-BPSO, which are means over 30 independent runs. From this table, the best combinations of m, v_{max}, and φ have been presented in Table 2.

Table 2 shows that TV_T-BPSO obtained the best results when the swarm size m has been set to 30, 40, and 50. For this paper, we choose the swarm

Table 1: Results obtained by TV_T-BPSO on Ks_8a, Ks_16b, Ks_24d, WCI_{100}, UCI_{500}, $ISCI_{1000}$ over 30 independent runs.

Instance	D	m	φ	v_{max}=2	v_{max}=4	v_{max}=6	v_{max}=8
Ks_8a	8	40	4	**3924400.00**	3923992.00	3923584.00	3920320.00
			5	3924400.00	3924400.00	3923584.00	3920728.00
		50	4	3924400.00	3923992.00	3921952.00	3919912.00
			5	3924400.00	3924400.00	3923584.00	3921544.00
Ks_16b	16	40	4	**9352998.00**	9352822.60	9341453.27	9299547.90
			5	9352647.20	9352822.60	9351945.60	9339999.47
		50	4	9352998.00	9352647.20	9338888.57	9327993.80
			5	9352822.60	9352647.20	9344899.37	9341450.33
Ks_24d	24	40	4	11782041.70	**11810362.67**	11796323.70	11777536.27
			5	11768199.07	11806689.43	11798464.23	11788310.37
		50	4	11783470.80	11803592.73	11800329.20	11782663.23
			5	11779794.33	11806170.67	11801253.70	11790710.47
Instance	D	m	φ	v_{max}=10	v_{max}=12	v_{max}=14	v_{max}=16
WCI_{100}	100	40	4	26565.17	26356.53	26125.07	25809.67
			5	**26577.17**	26517.70	26371.07	26197.80
		50	4	26483.20	26487.40	26248.83	25967.20
			5	26546.90	26561.70	26467.50	26295.60
UCI_{500}	500	40	4	190915.27	190089.23	189437.57	186639.97
			5	190838.33	190682.17	190588.47	190443.97
		50	4	189783.00	190539.67	189216.20	186931.80
			5	189943.37	**191836.93**	191058.23	190780.77
$ISCI_{1000}$	1000	30	4	258711.50	258702.63	258066.97	257099.60
			5	258934.27	259088.20	**259114.77**	258280.57
		40	4	257657.10	258074.33	258066.87	257529.03
			5	257906.97	258804.40	258979.37	258837.37

Table 2: Best combinations of m and v_{max} (according to Table 1) in the optimization of ks_8a, ks_16b, ks_24d, WCI_{100}, UCI_{500}, and $ISCI_{1000}$.

Instance	D	m	v_{max}
Ks_8a	8	40	2
Ks_16b	16	40	2
Ks_24d	24	40	4
WCI_100	100	40	10
UCI_500	500	50	12
$ISCI$_1000	1000	30	14

size 40 (which is also a standard swarm size for BPSO [24, 31]) for all the following experiments.

Table 2 also shows that TV_T-BPSO uses the different v_{max} for obtaining the best results of six representative 0-1 knapsack instances. It can be observed that TV_T-BPSO requires a larger v_{max} for high-dimensional problems and a smaller v_{max} for low-dimensional problems. Note that, in practice, the original BPSO [8] and its variants [24, 14, 15] used a constant v_{max} (4 or 6) instead of variable v_{max}. We conduct a logarithmic regression analysis using the dimension D and v_{max} values from Table 2, as shown in Fig. 9. Based on this analysis, we recommended the following rule for choosing v_{max} according to a given D value:

$$v_{max}(\mathtt{D}) = 2.6655 \ln(\mathtt{D}) - 4.10, \quad \mathtt{D} \geq 4. \tag{20}$$

To find out the optimal value for φ, the results on TV_T-BPSO with a fixed swarm size of 40 is presented in Table 3. It can be seen that TV_T-BPSO obtained the best results with φ=1 to φ=5, and thus the upper and lower limit of φ are chosen as φ_{max}=5 and φ_{min}=1, for the experiments on TV_T-BPSO in this paper.

5.3. Experimental results

Following the previous section, we compare TV_T-BPSO with S_T-BPSO, L_T-BPSO, V_{T1}-BPSO, and V_{T2}-BPSO, respectively. For this comparison, first, we use the low-dimensional 0-1 knapsack instances, then the high-dimensional 0-1 knapsack instances. All the comparison results are averaged over 30 independent runs. For each instance, the best BPSO was compared with the other four BPSOs using a t-test with the significance level set at 0.05. If the best BPSO is significantly better than the other four, then the

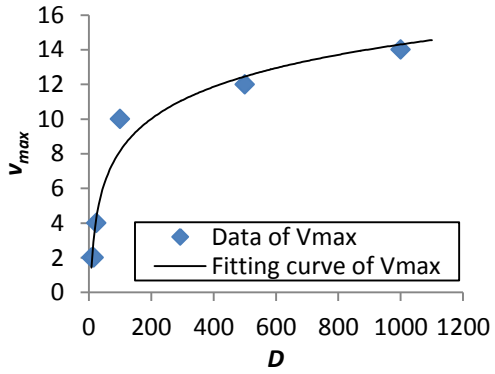


Figure 9: A fitting curve capturing the relationship between D and v_{max} using the logarithmic regression analysis.

Table 3: Results obtained by TV_T-BPSO on Ks_8a, Ks_16b, Ks_24d, WCI_{100}, UCI_{500}, $ISCI_{1000}$ with the swarm size 40.

	Ks_8a	Ks_16b	Ks_24d
	v_{max}=2	v_{max}=2	v_{max}=4
φ=1	3921952	**9352998**	11769796.3
φ=2	**3924400**	9352998	11804350.6
φ=3	3924400	9352471.8	11806885.3
φ=4	3924400	9352998	**11810362.7**
φ=5	3924400	9352647.2	11806689.4
	WCI_{100}	UCI_{500}	$ISCI_{1000}$
	v_{max}=10	v_{max}=12	v_{max}=14
φ=1	25169.23	151145.2	242298.8
φ=2	25959.1	164316.57	256189.03
φ=3	26376.4	185893.37	257066.9
φ=4	26565.17	190089.23	258066.87
φ=5	**26577.17**	**190682.17**	**258979.37**

corresponding value is marked in bold. In addition, if a BPSO consistently achieved the optimal value or better than the best known value (where the optimal value is unknown), the corresponding entry is marked with *.

In Table 4, the parameter settings used for TV_T-BPSO and four other well-known BPSOs are summarized. The value of v_{max} and w of S_T-BPSO, L_T-BPSO, V_{T1}-BPSO, and V_{T2}-BPSO are set according to [8], [14], [16], and [15], respectively.

Table 4: Parameter settings of five different BPSOs for the experiments over the 0-1 knapsack benchmark problems.

Algorithm	m	c_1=c_2	v_{max}	φ_{max}	φ_{min}	w_{max}	w_{min}
S_T-BPSO	40	2	4	-	-	1	1
L_T-BPSO	40	2	4	-	-	1	1
V_{T1}-BPSO	40	2	6	-	-	0.9	0.4
V_{T2}-BPSO	40	2	6	-	-	0.9	0.4
TV_T-BPSO	40	2	Eq. (20)	5	1	1	1

5.3.1. Low-dimensional 0-1 knapsack instances

In this section, three different sets of 0-1 knapsack instances have been used for the comparison purpose, which are widely used test instances in the literature [1, 14, 22, 23]. The first two sets belong to the deterministic group [1, 23] and [14]. Since the optimal solution of these instances is known, the comparison is made on the basis of average profit ($AvgPft$), standard deviation SD)), success rate (SR), and an average number of function evaluations ($AvgFEs$). Here, the success rate measures the percentage of successful runs where a successful run is defined as a run that produces the optimal solution before the run is terminated. For these two sets of instances, all the BPSOs in Table 4 will be terminated when they find the global optimum or they reach the maximum number of iterations Itr_{max}.

The third set of 0-1 knapsack instances belong to the non-deterministic group. This set consists of four 100-dimensional and four 500-dimensional instances generated from the procedures in Section 5.1. More specifically, the instances of 100 dimensions are generated using the value for R=1000, S=0.5, and n=100. Likewise, the instances of 500 dimensions are generated using the value for R=1000, S=0.5, and n=500. Since these instances have been randomly generated, the optimal solutions of these 0-1 knapsack instances are unknown. The comparison is made on the basis of average profit ($AvgPft$), standard deviation (SD), and the p-value obtained by t-test. For this set of instances, all BPSOs will be terminated when they reach the maximum number of iterations Itr_{max}.

The results on the first and second set of 0-1 knapsack instances are summarized in Table 5 and 6, respectively. The column "BK" of Table 6 represents the best known profit for each of the instances of the second set.

Table 5 shows that TV_T-BPSO and V_{T2}-BPSO obtained similar results on the first set of 0-1 knapsack instances. However, if we compare these two BPSOs in terms of the results of SR and AFE, then TV_T-BPSO is better,

Table 5: The obtained results on $AvgPft \pm SD$ (first line), SR (second line), and $AvgFEs$ (third line) obtained by the five BPSO variants over the test instances f_1 to f_{10} [1, 23].

Instance	D	Opt	TV_T-BPSO*	BPSO	L-BPSO	V_{T1}-BPSO	V_{T2}-BPSO
f_3	4	35	35.00±0.00 100 **41**	35.00±0.00 100 92	35.00±0.00 100 45	34.70±1.32 93.33 2712	35.00±0.00 100 43
f_4	4	23	23.00±0.00 100 **41**	23.00±0.00 100 461	22.97±0.18 96.67 1372	22.97±0.18 96.67 1372	23.00±0.00 100 43
f_9	5	130	130.00±0.00 100 **54.67**	130.00±0.00 100 1058.67	127.20±5.16 76.67 9373.33	128.40±4.15 86.67 5377.33	130.00±0.00 100 56
f_7	7	170	107.00 ±0.00 100 **140**	103.90 ±4.28 30 28019	105.73±1.41 46.67 21364	103.53±4.54 43.33 22685	107.00±0.00 100 167
f_1	10	295	295.00±0.00 100 **413**	294.57±1.38 86.67 8556	290.77±10.66 56.67 17429	281.83±21.11 26.67 29369	295.00±0.00 100 851
f_6	10	52	52.00 ±0.00 100 **180**	51.80±0.55 86.67 7913	51.63±0.85 83.33 6973	51.33±1.15 66.67 13469	52.00±0.00 100 280
f_5	15	481.07	481.07±0.00 100 **1976**	478.19±10.94 93.33 4223	474.80±17.01 73.33 10920	427.19±39.87 16.67 33364	481.07±0.00 100 14715
f_2	20	1024	1024.00±0.00 100 **6820**	1024.00±0.00 100 3533	1002.53±22.32 26.67 29449	931.27±53.80 0 40000	1024.00±0.00 100 31289
f_{10}	20	1025	1025.00±0.00 100 4707	1025.00±0.00 100 **1980**	989.70±36.43 13.33 34703	933.07±50.78 3.33 38681	1025.00±0.00 100 29440
f_8	23	9767	9767.00±0.00 **100** 11060	9766.93±0.37 96.67 6001	9766.30±2.59 90 6424	9757.33±8.93 20 32371	9766.97±0.18 96.67 36060

since it has a higher success rate for all the 0-1 knapsack instances with the least amount of AFE. This proves that TV_T-BPSO is reliable and faster than the four other BPSOs over the first set of 0-1 knapsack instances. Compared to TV_T-BPSO, V_{T1}-BPSO has the lowest success rate and the highest amount of AFE for all the instances, except the instance ks_24d. Therefore, it can be said that V_{T1}-BPSO is not a reliable neither a faster algorithm for this set of instances.

Table 6 shows that TV_T-BPSO performs well on the optimization of second set of 0-1 knapsack instances. In particular, TV_T-BPSO obtained better results than the best known results and the results obtained by the other four BPSOs with the least amount of $AvgFEs$. In addition, TV_T-BPSO obtained these results with the highest success rate. These results show that TV_T-BPSO is more reliable and faster for this set of instances. Table 6 also

Table 6: The results on $AvgPft$ (first line), SD (second line), SR (third line), $AvgFEs$ (fourth line) of five different BPSOs over the test instances Ks_16a to Ks_24e [14].

Instance	D	Opt.	BK[14, 22]	TV_T-BPSO*	S_T-BPSO	L_T-BPSO	V_{T1}-BPSO	V_{T2}-BPSO
Ks_16a	16	7850983	7848800	7850983.00 ±0.00 100 **9252**	7771152.37 ±76092.77 16.67 35263	7797246.87 ±67539.40 26.67 29888	7693936.93 ±122910.45 16.67 33775	7850983.00 ±0.00 100 21676
Ks_16b	16	9352998	9352800	9352998.00 ±0.00 100 **7835**	9245414.87 ±84778.02 26.67 30153	9250490.53 ±120292.36 26.67 29907	9127967.97 ±159529.14 3.33 38725	9352998.00 ±0.00 100 19859
Ks_16c	16	9151147	9149200	9150326.30 ±4495.16 96.67 **9223**	9036801.60 ±105374.18 13.33 35448	9052814.63 ±83568.41 16.67 33597	8924175.00 ±147544.37 6.67 37441	9150326.30 ±4495.16 96.67 19669
Ks_16d	16	9348889	9345000	9347669.20 ±3721.96 **90** **13440**	9271454.97 ±69041.39 13.33 35053	9294715.13 ±52798.58 20 32740	9188651.17 ±95956.06 0 40000	9347262.60 ±4217.41 86.67 25804
Ks_16e	16	7769117	7767300	**7768496.13** **±3400.63** **96.67** **13660**	7693213.00 ±75474.04 13.33 35517	7713892.53 ±78597.11 30 28467	7587932.27 ±125788.08 3.33 38681	7767875.27 ±4725.57 93.33 21015
Ks_20a	20	10727049	10720314	**10724840.50** **±7523.13** **90** **18976**	10669618.47 ±60759.13 26.67 32543	10692481.67 ±58716.61 33.33 27853	10563639.87 ±120918.17 3.33 38672	10714666.40 ±13060.22 43.33 38615
Ks_20b	20	9818261	9805480	**9815420.10** **±9323.38** **90** **18389**	9735538.37 ±79571.34 20 34121	9766711.50 ±57579.60 30 29204	9632312.53 ±112517.84 0 40000	9797837.57 ±20116.05 40 38263
Ks_20c	20	10714023	10710947	**10712635.83** **±4077.25** **86.67** **19208**	10615440.83 ±115323.68 23.33 32559	10688081.47 ±45556.95 36.67 26319	10486056.43 ±133805.32 0 40000	10709990.30 ±6230.09 60 36317
Ks_20d	20	8929156	8923712.21	**8929156.00** **±0.00** **100** **15155**	8866124.10 ±74952.33 33.33 31507	8883763.17 ±70473.79 43.33 24383	8735162.43 ±104543.78 0 40000	8916392.47 ±15587.49 53.33 37295
Ks_20e	20	9357969	9355930.35	**9356953.67** **±3038.76** **66.67** **24236**	9306943.83 ±54249.45 23.33 34183	9318815.13 ±68243.98 20 32973	9181466.33 ±126241.20 0 40000	9353699.90 ±9589.14 56.67 36537
Ks_24a	24	13549094	13532060	**13533425.33** **±20555.05** **56.67** **30016**	13507548.40 ±58236.73 36.67 31949	13512451.77 ±31391.28 26.67 31051	13323722.73 ±137591.62 3.33 38796	13499799.43 ±22346.71 0 40000
Ks_24b	24	12233713	12223443	**12222004.63** **±15052.33** **53.33** **29381**	12152367.53 ±71669.10 16.67 36640	12191466.07 ±51448.35 30 30167	12031294.90 ±107235.86 0 40000	12182354.87 ±25839.01 3.33 39867
Ks_24c	24	12448780	12443349	**12444101.20** **±10802.00** **70** 29713	12395398.53 ±78169.05 36.67 31723	12420324.73 ±47375.30 40 26820	12255888.23 ±114864.01 3.33 39008	12414359.70 ±27122.87 16.67 38471
Ks_24d	24	11815315	11803712	**11809811.83** **±9023.92** **50** 32332	11778345.20 ±58548.76 46.67 29028	11776442.47 ±59364.89 26.67 30903	11632526.57 ±128661.51 0 40000	11776692.07 ±21915.93 10 39400
Ks_24e	24	13940099	13932526	**13937049.83** **±6653.97** **76.67** 28019	13914845.70 ±45334.60 43.33 30343	13904779.23 ±67627.36 43.33 25769	13777674.00 ±150279.17 13.33 35296	13897801.70 ±26691.06 13.33 39533

shows that V_{T2}-BPSO obtained similar results to those of TV_T-BPSO on the instances of 16-D. However, the performance of V_{T2}-BPSO decreases significantly with the increase of dimensions of the instances Ks_20a to Ks_24e. It

Table 7: The results on $AvgPft \pm SD$ (first line) and p-value (second line) of five different BPSOs over 100-D and 500-D non-deterministic 0-1 knapsack instances. Under the t-test, if TV_T-BPSO is statistically better than an existing BPSO, then the p-value corresponding to that BPSO is highlighted.

Instance	D	TV_T-BPSO*	S_T-BPSO	L_T-BPSO	V_{T1}-BPSO	V_{T2}-BPSO
UCI_{100}	100	41082.83±40.06 -	40712.53±220.53 **3.38E-10**	40676.07±692.24 **0.003193397**	38131.00±1802.62 **7.31E-10**	34830.10±534.03 **4.75E-33**
WCI_{100}	100	26656.13±29.68 -	25537.77±172.07 **2.37E-26**	26441.40±90.42 **2.34E-14**	26410.83±127.55 **1.15E-11**	25243.77±116.74 **5.13E-36**
SCI_{100}	100	30499.50±40.15 -	29684.67±124.12 **1.71E-28**	30257.10±80.75 **2.70E-18**	30355.13±148.75 **1.23E-05**	29507.93±90.99 **4.20E-39**
$ISCI_{100}$	100	23215.43±37.03 -	22345.37±115.84 **2.02E-30**	22921.5±101.12 **3.57E-17**	23038.53±148.30 **3.76E-07**	22182.30±84.09 **3.95E-41**
UCI_{500}	500	194552.37±599.16 -	149200.87±2716.56 **9.09E-40**	175225.43±3178.31 **1.22E-25**	173451.57±11976.25 **1.44E-10**	141932.70±1510.21 **5.89E-57**
WCI_{500}	500	135212.10±230.51 -	128062.43±375.87 **5.26E-55**	129691.47±250.54 **2.48E-63**	134244.50±511.36 **8.92E-12**	127765.13±257.35 **3.98E-70**
SCI_{500}	500	150899.93±204.06 -	145025.33±279.12 **1.65E-60**	146495.30±279.03 **6.24E-54**	150448.47±458.71 **1.50E-05**	144913.63±178.81 **2.06E-70**
$ISCI_{500}$	500	130218.50±196.51 -	125561.10±212.23 **3.31E-63**	126668.97±204.35 **3.75E-57**	129927.20±325.56 **0.000117656**	125454.73±157.00 **4.67E-65**
w/t/l		-	8/0/0	8/0/0	8/0/0	8/0/0

can be seen that V_{T1}-BPSO shows the worse performance in terms of $AvgPft$, SD, SR, and $AvgFEs$.

Table 7 summarized the results on 8 non-deterministic 0-1 knapsack instances of 100 and 500 dimensions. In this table, the first and second lines have been used to present $AvgPft \pm SD$ and the p-values obtained by t-test, respectively. A highlighted p-value (< 0.05) means that the results obtained by TV_T-BPSO are much better than the results obtained by the existing BPSOs or vice verse. The last row of this table represents the number of instances that TV_T-BPSO wins/ties/loses over the existing BPSOs.

Table 7 shows that TV_T-BPSO significantly performs well on the optimization of 100-D, 500-D non-deterministic instances. In contrast, V_{T2}-BPSO shows the worst performance on the optimization of this set of test instances. Specifically, the performance of V_{T2}-BPSO has been decreased with the number of dimensions of the problems. It is mentioned in [15], V_{T2}-BPSO significantly performs well compared to the well-known BPSOs [31, 32, 33] on the set of low-dimensional continuous optimization problems [34]. However, our experimental results show that V_{T2}-BPSO is not suitable for the higher dimensional problems. In this set of instances, V_{T1}-BPSO obtained the better optimization results compared to S_T-BPSO and L_T-BPSO.

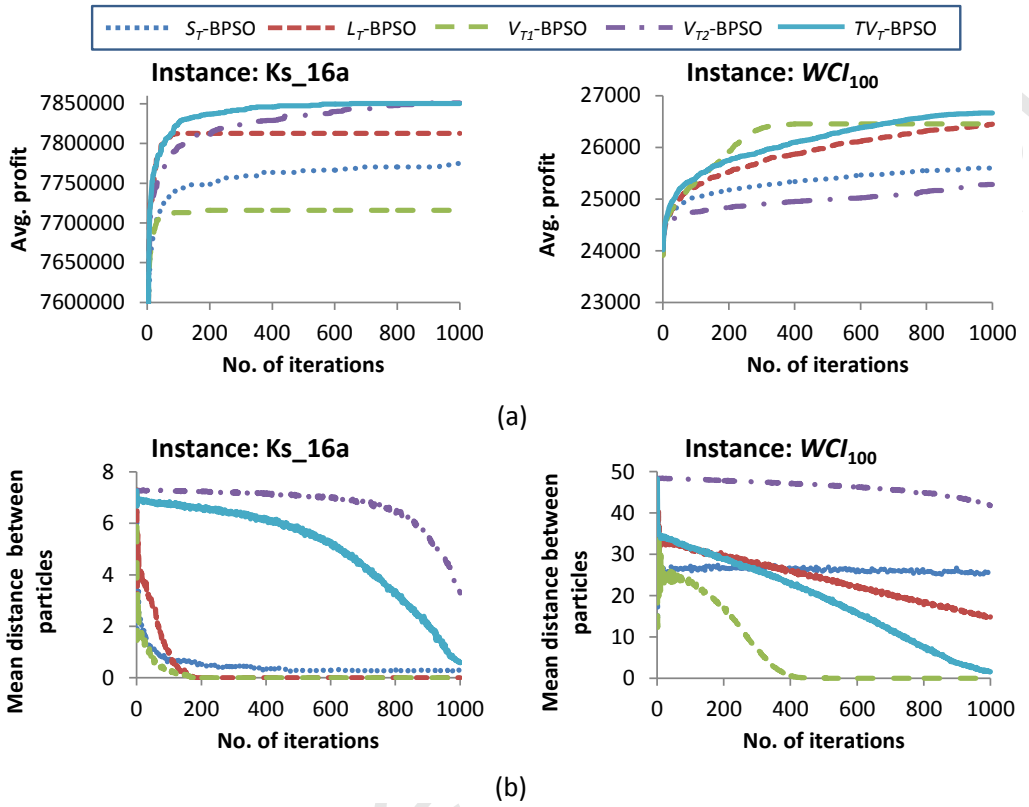


Figure 10: Comparing five different BPSOs on the benchmark instances ks_16a and WCI_{100} on their a) convergences, and b) diversity profiles.

Figure 10 shows the convergence and diversity profiles of the five BPSO variants on the instances ks_16a and WCI_{100}. From Fig. 10a, it can be seen that TV_T-BPSO was able to outperform the other BPSO variants consistently on both the deterministic and non-deterministic 0-1 knapsack benchmark instances. V_{T1}-BPSO performed poorly ks_16a. However, though better on WCI_{100} of 100-D. Comparing to V_{T1}-BPSO, V_{T2}-BPSO performed better on the instance ks_16a but poorly on the instance WCI_{100}.

Fig. 10b shows the diversity profiles on instances ks_16a and WCI_{100}. This figure shows that TV_T-BPSO can maintain a better diversity for both instances resulting in a good balance between exploration and exploitation during the run. It can be noted that the swarm diversity decreases to zero towards the end of the run, which shows that TV_T-BPSO can provide a stronger exploitation in the final stage. This helps explain the better results

by TV_T-BPSO on the above sets of 0-1 knapsack instances. In contrast, Fig. 10b shows that S_T-BPSO, L_T-BPSO, V_{T1}-BPSO, and V_{T2}-BPSO are unable to maintain a good balance between exploration and exploitation, resulting in their poorer results.

5.3.2. High-dimensional 0-1 knapsack instances

Table 5 to Table 7 show that TV_T-BPSO performs well on instances up to 500 dimensions. In order to investigate the scalibity of TV_T-BPSO to even higher dimensions, additional experiments have been carried out on twelve 0-1 knapsack instances of 1000, 2000, and 5000 dimensions. For these experiments, we use the same parameter settings as shown in Table 4.

Table 8 presents the results by the five different BPSOs on 1000-D, 2000-D, and 5000-D instances, where the last row represents the number of instances that TV_T-BPSO wins/ties/loses. It can be observed that TV_T-BPSO significantly outperformed other four BPSO variants on the non-deterministic 0-1 knapsack instances with more than 1000 dimensions. In contrast, the performance of V_{T2}-BPSO deteriorates a great deal on this set of instances. It is

Table 8: The results on $AvgPft \pm SD$ (first line) and p-value (second line) of five different BPSOs over 1000-D to 5000-D non-deterministic 0-1 knapsack instances.

Instance	D	TV_T-BPSO*	S_T-BPSO	L_T-BPSO	V_{T1}-BPSO	V_{T2}-BPSO
UCI_{1000}	1000	380640.03±1397.89 -	285096.00±2907.05 **4.53E-60**	312836.60±4455.70 **8.09E-41**	340616.60±19044.61 **2.29E-12**	276968.53±2047.78 **2.09E-75**
WCI_{1000}	1000	269081.53±578.08 -	256849.47±628.30 **2.94E-60**	259007.63±391.60 **5.33E-55**	268128.43±1091.88 **0.000118**	256723.57±296.24 **1.42E-53**
SCI_{1000}	1000	312629.20±520.02 -	303167.17±429.02 **2.08E-55**	305246.57±420.93 **2.79E-50**	312332.83±775.88 0.099603	303130.40±283.43 **9.66E-50**
$ISCI_{1000}$	1000	262444.47±525.65 -	254392.30±333.35 **4.28E-51**	256063.80±319.78 **1.36E-45**	262017.30±630.85 **0.006114**	254470.63±374.25 **1.12E-52**
UCI_{2000}	2000	756871.80±3380.62 -	559550.23±5350.69 **1.24E-69**	598293.10±6658.42 **2.14E-55**	660994.97±32962.45 **5.27E-16**	550112.60±3337.33 **5.27E-16**
WCI_{2000}	2000	530579.83±751.70 -	512181.07±695.33 **6.32E-66**	515252.33±795.73 **7.39E-60**	529786.27±1093.49 **0.001892**	512122.00±520.41 **5.30E-63**
SCI_{2000}	2000	624930.57±898.16 -	611012.67±491.29 **1.04E-48**	613746.70±597.27 **2.14E-47**	625499.53±1073.54 **0.030007**	611051.63±429.11 **2.45E-46**
$ISCI_{2000}$	2000	516189.63±785.10 -	504070.47±503.50 **1.90E-51**	506166.50±462.34 **4.01E-46**	516505.00±735.64 0.113843	503910.47±371.00 **2.25E-46**
UCI_{5000}	5000	1758217.77±7747.40 -	1327171.07±7616.02 **4.09E-86**	1385014.63±9786.66 **9.39E-76**	1527048.27±48696.11 **3.35E-22**	1311025.30±3667.60 **8.20E-70**
WCI_{5000}	5000	1302697.10±1494.24 -	1268726.00±964.20 **7.84E-60**	1273625.77±1047.98 **4.89E-58**	1299083.80±2204.27 **1.13E-09**	1268303.47±666.19 **3.70E-52**
SCI_{5000}	5000	1283927.03±1314.99 -	1250033.87±1000.27 **7.72E-66**	1255255.63±756.80 **1.90E-56**	1281307.33±1937.18 **1.29E-07**	1249858.17±540.63 **1.06E-52**
$ISCI_{5000}$	5000	1262279.00±1304.29 -	1239822.40±768.42 **3.58E-52**	1243482.67±714.20 **2.52E-47**	1262350.17±1562.29 0.848801	1240059.40±678.69 **1.30E-49**
w/t/l		-	12/0/0	12/0/0	12/3/0	12/0/0

also worth noting that for this set of instances, V_{T1}-BPSO performed better than S_T-BPSO, L_T-BPSO, and V_{T2}-BPSO.

5.3.3. High-dimensional engineering optimization problems

In this section, the performance of TV_T-BPSO is examined over a high-dimensional ground structure. A ground structure is a complete truss with all possible member (m) connections among all nodes (n) [43]. In truss, the nodes with a load or support are called basic nodes, and the rest of the nodes are called non-basic nodes, as shown in Fig. 11.

The ultimate optimization goal of a truss structure is to minimize its structural weight, subject to various constraints. This can be achieved by optimization on both the topology and size of the truss structure. The topology optimization selects a subset of the members of the ground structure to form a stable topology, whereas the size optimization finds the optimal cross-sectional areas of those members that are selected in the topology optimization phase subject to stress, displacement, and kinematic stability constraints. In this study, the set of topology variables is denoted by $\mathbf{x}$=$[x_1, x_2, \cdots, x_m]$ subject to $x_m \in \{0, 1\}$ and the set of size variables is denoted by $\mathbf{A}$=$[A_1, A_2, \cdots, A_m]$ subject to A_m within the range $[A_{min}, A_{max}]$, where A_{min} and A_{max} are the lower and upper bounds of the member cross-section areas of the truss structure. Considering its topology variable $\mathbf{x}$ and size variable $\mathbf{A}$, optimization of the topology and size of the truss structure

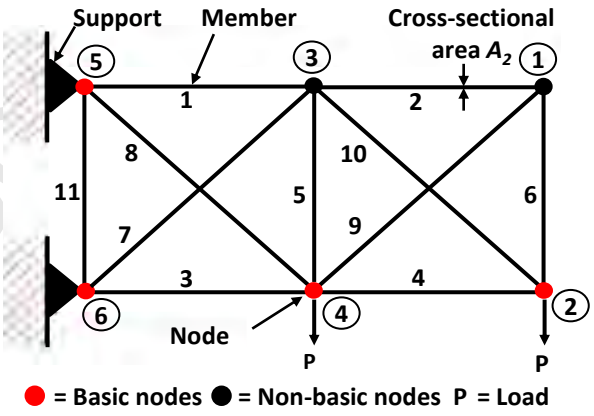


Figure 11: Illustration of a 11-member, 6-node ground structure.

can be formulated as:

$$\begin{aligned} \text{Find} \quad & \mathbf{x} \\ \text{to minimize} \quad & W(\mathbf{x}) = \sum_{i=1}^{m} \rho_i \ell_i A_i x_i \\ \text{subject to} \quad & g_1 : \textit{Truss is acceptable to the user}, \\ & g_2 : \textit{Truss is internally and externally stable}, \\ & g_3 : \textit{Truss is kinematically stable}, \\ & g_4 : \sigma_i \leq S_i, \ i = 1, 2, \ldots, m, \\ & g_5 : \delta_j \leq \delta_j^{max}, \ j = 1, 2, \ldots, n, \end{aligned} \tag{21}$$

where W denotes the weight of the truss structure and x_i represents the decision variable of the i-th member. The parameters ρ_i, ℓ_i, and A_i represent the density, length, and cross-sectional area of the i-th member, respectively. In this equation, constraint g_1 is used to ensure that the found topology contains all the basic nodes, g_2 ensures the internal and external stability of the found topology to satisfy the following equations:

$$m + 3 \geq 2n, \tag{22}$$

and

$$m + r \geq 2n, \tag{23}$$

where r denotes the number of reaction components of the truss. Constraint g_3 ensures the kinematic stability of the truss by checking the positive definiteness of the stiffness matrix of the truss. Constraint g_4 ensures that the stress of the i-th member is less than or equal to the user specified limits. Similarly, constraint g_5 ensures that the displacement of the j-th node is less than or equal to the user specified limits. Note that if a solution violates one of these constraints, then a penalty is assigned to indicate that this solution is not a good solution.

In this study, we use TV_T-BPSO to optimize the topology of the truss structure, and a standard PSO to optimize the size of the truss structure (see Section 2.1). For the topology optimization, TV_T-BPSO uses the same parameter settings as those in Section 2.1. For the size optimization, PSO uses the default values for its parameters. In addition, the population size and the number of maximal iterations of PSO are set to 40 and 200, respectively.

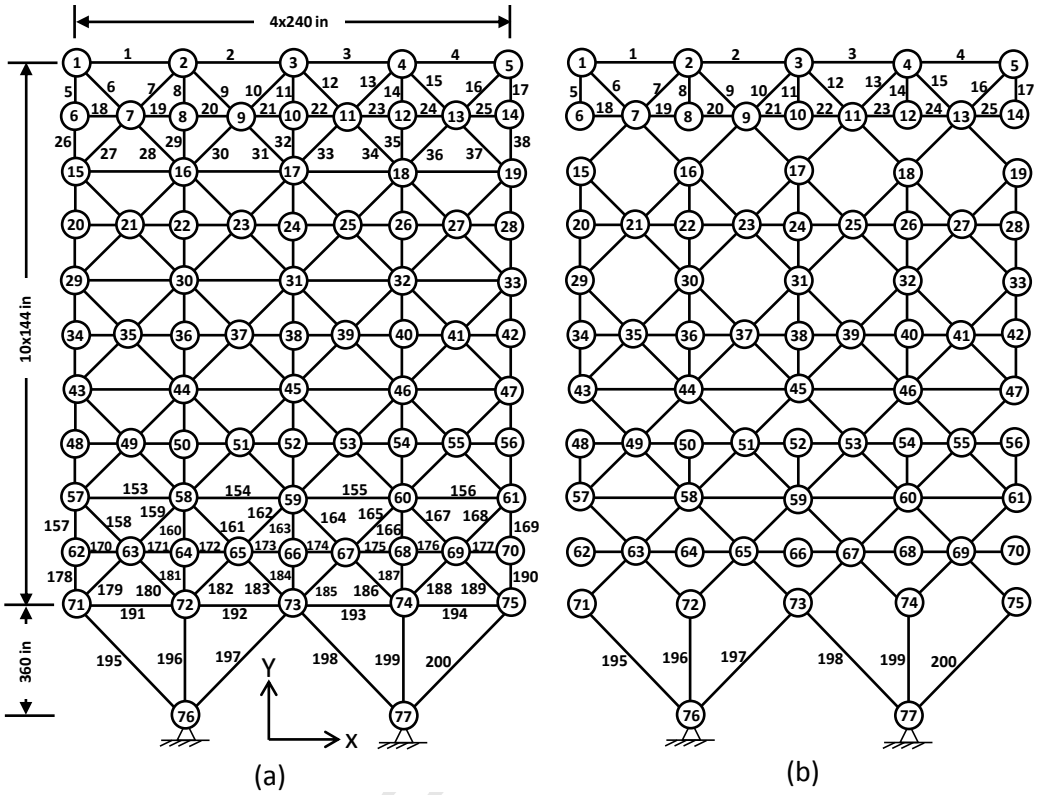


Figure 12: (a) A 200-member, 77-node ground structure and (b) its corresponding topological solution.

The velocity v_{max} of PSO is set to A_{max} i.e., the maximum allowed cross-sectional area of a member of the truss. The lower and upper bounds of the position x_{id} of PSO are set to 0.0 and A_{max}, respectively.

With the above settings, the experiment is conducted using a well-known truss problem, namely 200-member, 77-node ground structure [44, 45, 47, 48], as shown in Fig. 12(a). This truss is considered to be a high-dimensional engineering problem since the topology optimization involves 200 binary variables and the size optimization involves 200 continuous variables, totaling 400 variables. To optimize this truss, the following material properties and design parameters are considered. Young's modulus and density of the materials are 30000 ksi and 0.283 lb/in^2, respectively. The allowable stress and displacement are set to 10 ksi and 0.5 inches, respectively. The lower (A_{min})

Table 9: Comparison of optimized designs for the 200-member truss problem.

Group No.	TV_T-BPSO	Sonmez [44]	Kaveh [45]	Kaveh [46]	Bekdas [47]	Dede [48]
1	1.3280	0.1039	0.382	0.1480	0.1425	0.113546
2	20.000	0.9463	2.116	0.9460	0.9637	0.948427
3	0.9734	0.1037	0.102	0.1010	0.1005	0.107798
4	0.1319	0.1126	0.141	0.1010	0.1000	0.100009
5	-	1.9520	3.662	1.9461	1.9514	1.934462
6	0.8823	0.2930	0.176	0.2979	0.2957	0.288872
7	-	0.1064	0.121	0.1010	0.1156	0.211586
8	1.1956	3.1249	3.544	3.1072	3.1133	3.090253
9	0.3598	0.1077	0.108	0.1010	0.1006	0.103114
10	3.0457	4.1286	5.565	4.1062	4.1100	4.090254
11	0.7899	0.4250	0.542	0.4049	0.4165	0.450150
12	-	0.1046	0.138	0.1944	0.1843	0.100707
13	1.3257	5.4803	5.139	5.4299	5.4567	5.479848
14	2.9063	0.1060	0.101	0.1010	0.1000	0.101144
15	1.2331	6.4853	8.742	6.4299	6.4559	6.479849
16	1.1068	0.5600	0.431	0.5755	0.5800	0.532949
17	1.7802	0.1825	0.998	0.1349	0.1547	0.132492
18	-	8.0445	7.212	7.9747	8.0132	7.944450
19	0.0229	0.1026	0.152	0.1010	0.1000	0.100486
20	-	9.0334	8.452	8.9747	9.0135	8.944437
21	2.5888	0.7844	0.835	0.7064	0.7391	0.701077
22	4.3040	0.7506	0.413	0.4225	0.7870	1.377693
23	-	11.3057	10.146	10.8685	11.1795	11.239401
24	6.7601	0.2208	0.874	0.1010	0.1462	0.228718
25	-	12.273	11.384	11.8684	12.1799	12.239392
26	5.9606	1.4055	1.197	1.03599	1.3424	1.684935
27	-	5.1600	5.747	6.6859	5.4844	4.913586
28	4.0867	9.9930	7.823	10.8111	10.1372	9.718956
29	19.9921	14.7014	13.655	13.8464	14.5262	15.021916
Weight (lb)	**24765.22**	25533.79	25193.22	25467.9	25521.81	25,664.0023

and upper (A_{max}) bounds on the member cross-sectional areas are set to 0.1 and 20.0 in^2, respectively. The members of this truss are grouped considering their symmetry, as done by Sonmez [44], Kaveh [45], Bekdas [47], and Dede [48]. According to [44, 45, 47, 48], this truss is optimized by applying 1.0 kip force which is acting in the positive x-direction at nodes 1, 6, 15, 20, 29, 43, 48, 57, 62, and 71.

Fig. 12(b) shows the topological solution found by the proposed TV_T-BPSO for the 200-member, 77-node ground structure. To the best of our knowledge, this is the first attempt to optimize the topology of such a high-

dimensional truss structure. Table 9 compares the results of TV_T-BPSO with those reported in [44, 45, 46, 47, 48]. It can be observed that TV_T-BPSO outperforms significantly the other methods. The weight (24765.22 lb) obtained by TV_T-BPSO is much lower than the weights obtained by the other compared methods. This again demonstrates that TV_T-BPSO can perform competently in combinatorial optimization problems.

6. Conclusion

In this paper we have proposed a modified version of BPSO termed TV_T-BPSO, which adopts a time-varying transfer function to address the shortcoming of existing transfer functions by providing a better balance between exploration and exploitation for the BPSO during its optimization run. We have presented some empirical analyses of the search behaviours of these BPSO variants using different transfer functions, in an effort to understand how this time-varying transfer function makes a good balance between exploration and exploitation during the search. The behaviours of TV_T-BPSO are compared with that of the original BPSO (S_T-BPSO) and three other well-known modified BPSOs over several low-dimensional (D$\leq$500) 0-1 knapsack benchmark instances. Our experimental results have demonstrated that TV_T-BPSO outperforms the existing BPSOs over these low-dimensional benchmark instances. Furthermore, we have examined the scalability of TV_T-BPSO on high-dimensional (1000 $\geq$D$\leq$5000) 0-1 knapsack benchmark instances and a 200-member truss design problem. The experimental results have shown that TV_T-BPSO can scale well to high-dimensional combinatorial optimization problems.

[1] D. Zou, L. Gao, S. Li, J. Wu, Solving 0-1 knapsack problem by a novel global harmony search algorithm, Applied Soft Computing 11 (2011) 1556–1564.

[2] D. Pisinger, Where are the hard knapsack problems?, Computers and Operations Research 32 (2005) 2271–2284.

[3] X. Geng, Z. Chen, W. Yang, D. Shi, K. Zhao, Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search, Applied Soft Computing 11 (2011) 3680–3689.

[4] M. A. A. Pedrasa, T. D. Spooner, I. F. MacGill, Scheduling of demand side resources using binary particle swarm optimization, IEEE Transactions on Power Systems 24 (2009) 1173–1181.

[5] P.-H. Chen, C.-C. Kuo, F.-H. Chen, C.-C. Chen, Refined binary particle swarm optimization and application in power system, WSEAS TRANSACTIONS on SYSTEMS (2009) 169–178.

[6] R. G. Parker, R. L. Rardin, Discrete Optimization, Computer science and applied mathematics, Academic Press, 1988.

[7] Yang, Xin-She, Nature-inspired metaheuristic algorithms, Luniver press, 2010.

[8] J. Kennedy, R. C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, volume 5, pp. 4104–4108.

[9] C.-J. Liao, C.-T. Tseng, P. Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, Computers and Operations Research 34 (2007) 3099–3111.

[10] B. Jarboui, N. Damak, P. Siarry, A. Rebai, A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems, Applied Mathematics and Computation 195 (2008) 299–308.

[11] M. Naeem, U. Pareek, D. C. Lee, Swarm intelligence for sensor selection problems, IEEE Sensors Journal 12 (2012) 2577–2585.

[12] J. C.-W. Lin, L. Yang, P. Fournier-Viger, T.-P. Hong, M. Voznak, A binary pso approach to mine high-utility itemsets, Soft Computing (2016) 1–19.

[13] L. Han, C. Huang, S. Zheng, Z. Zhang, L. Xu, Vanishing point detection and line classification with bpso, Signal, Image and Video Processing 11 (2017) 17–24.

[14] J. C. Bansal, K. Deep, A modified binary particle swarm optimization for knapsack problems, Applied Mathematics and Computation 218 (2012) 11042–11061.

[15] S. Mirjalili, A. Lewis, S-shaped versus v-shaped transfer functions for binary particle swarm optimization, Swarm and Evolutionary Computation 9 (2013) 1–14.

[16] L. Jian-Hua, Y. Rong-Hua, S. Shui-Hua, The analysis of binary particle swarm optimization, Journal of Nanjing University (Natural Sciences) 47 (2011) 504–514.

[17] T. Ting, M. V. C. Rao, C. K. Loo, A novel approach for unit commitment problem via an effective hybrid particle swarm optimization, IEEE Transactions on Power Systems 21 (2006) 411–418.

[18] I. X. Tassopoulos, G. N. Beligiannis, Solving effectively the school timetabling problem using particle swarm optimization, Expert Systems with Applications 39 (2012) 6029 – 6040.

[19] I. X. Tassopoulos, G. N. Beligiannis, A hybrid particle swarm optimization based algorithm for high school timetabling problems, Applied Soft Computing 12 (2012) 3472 – 3489.

[20] D. Liu, Q. Jiang, J. X. Chen, Binary inheritance learning particle swarm optimisation and its application in thinned antenna array synthesis with the minimum sidelobe level, IET Microwaves, Antennas Propagation 9 (2015) 1386–1391.

[21] F. Han, C. Yang, Y. Q. Wu, J. S. Zhu, Q. H. Ling, Y. Q. Song, D. S. Huang, A gene selection method for microarray data based on binary pso encoding gene-to-class sensitivity information, IEEE/ACM Transactions on Computational Biology and Bioinformatics 14 (2017) 85–96.

[22] J. Liu, Y. Mei, X. Li, An analysis of the inertia weight parameter for binary particle swarm optimization, IEEE Transactions on Evolutionary Computation PP (2015) 1–1.

[23] L. Wang, R. Yang, Y. Xu, Q. Niu, P. M. Pardalos, M. Fei, An improved adaptive binary harmony search algorithm, Information Sciences 232 (2013) 58–87.

[24] L. Wang, X. Wang, J. Fu, L. Zhen, A novel probability binary particle swarm optimization algorithm and its application, Journal of software 3 (2008) 28–35.

[25] Y. W. Jeong, J. B. Park, S. H. Jang, K. Y. Lee, A new quantum-inspired binary pso: Application to unit commitment problems for power systems, IEEE Transactions on Power Systems 25 (2010) 1486–1495.

[26] A. Modiri, K. Kiasaleh, Modification of real-number and binary pso algorithms for accelerated convergence, IEEE Transactions on Antennas and Propagation 59 (2011) 214–224.

[27] Z. Beheshti, S. M. Shamsuddin, S. S. Yuhaniz, Binary accelerated particle swarm algorithm (BAPSA) for discrete optimization problems, Journal of Global Optimization 57 (2013) 549–573.

[28] H. Banka, S. Dara, A hamming distance based binary particle swarm optimization (hdbpso) algorithm for high dimensional feature selection, classification and validation, Pattern Recognition Letters 52 (2015) 94 – 100.

[29] A. R. Jordehi, J. Jasni, Particle swarm optimisation for discrete optimisation problems: a review, Artificial Intelligence Review 43 (2015) 243–258.

[30] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft Computing (2017) 1–22.

[31] S. Lee, S. Soak, S. Oh, W. Pedrycz, M. Jeon, Modified binary particle swarm optimization, Progress in Natural Science 18 (2008) 1161–1166.

[32] Q. Shen, J.-H. Jiang, C.-X. Jiao, G. li Shen, R.-Q. Yu, Modified particle swarm optimization algorithm for variable selection in MLR and PLS model: QSAR studies of antagonism of angiotensin II antagonists, European Journal of Pharmaceutical Sciences 22 (2004) 145–152.

[33] L. Wang, X. Wang, J. Fu, L. Zhen, A novel probability binary particle swarm optimization algorithm and its application, Journal of Software 3 (2008) 28–35.

[34] 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 report 2005005 (2005) 2005.

[35] R. C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in: Proceedings of the 2001 congress on evolutionary computation., volume 1, IEEE, pp. 81–86.

[36] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, R. G. Harley, Particle swarm optimization: Basic concepts, variants and applications in power systems, IEEE Transactions on Evolutionary Computation 12 (2008) 171–195.

[37] N. Jin, Y. Rahmat-Samii, Advances in particle swarm optimization for antenna designs: Real-number, binary, single-objective and multiobjective implementations, IEEE Transactions on Antennas and Propagation 55 (2007) 556–567.

[38] I. Montalvo, J. Izquierdo, R. Prez, M. M. Tung, Particle swarm optimization applied to the design of water supply systems, Computers & Mathematics with Applications 56 (2008) 769–776.

[39] L.-Y. Chuang, H.-W. Chang, C.-J. Tu, C.-H. Yang, Improved binary pso for feature selection using gene expression data, Computational Biology and Chemistry 32 (2008) 29–38.

[40] F. Shahzad, A. R. Baig, S. Masood, M. Kamran, N. Naveed, Opposition-based particle swarm optimization with velocity clamping (OVCPSO), in: Advances in Computational Intelligence, Springer, 2009, pp. 339–348.

[41] J. Yang, H. Zhang, Y. Ling, C. Pan, W. Sun, Task allocation for wireless sensor network using modified binary particle swarm optimization, IEEE Sensors Journal 14 (2014) 882–892.

[42] R. W. Hamming, Error detecting and error correcting codes, Bell System technical journal 29 (1950) 147–160.

[43] W. Spillers, K. MacBain, Structural Optimization, Springer US, 2009.

[44] M. Sonmez, Artificial bee colony algorithm for optimization of truss structures, Applied Soft Computing 11 (2011) 2406 – 2418.

[45] A. Kaveh, M. Khayatazad, Ray optimization for size and shape optimization of truss structures, Computers and Structures 117 (2013) 82–94.

[46] A. Kaveh, R. Sheikholeslami, S. Talatahari, M. Keshvari-Ilkhichi, Chaotic swarming of particles: A new method for size optimization of truss structures, Advances in Engineering Software 67 (2014) 136 – 147.

[47] G. Bekda, S. M. Nigdeli, X.-S. Yang, Sizing optimization of truss structures using flower pollination algorithm, Applied Soft Computing 37 (2015) 322 – 331.

[48] T. Dede, Y. Ayvaz, Combined size and shape optimization of structures with a new meta-heuristic algorithm, Applied Soft Computing 28 (2015) 250 – 258.

Highlights

- Provide an analysis on the behaviour of BPSO employing existing transfer functions and identify their shortcomings in terms of balancing between exploration and exploitation.
- Propose and design a time-varying transfer function considering the shortcomings identified. The BPSO adopting the proposed time-varying transfer function is named as TV_T -BPSO.
- We validate the superiority of TV_T-BPSO by comparing it with the original BPSO and three existing BPSO variants on a diverse set of 0-1 knapsack benchmark instances and a truss design problem.

Graphical Abstract

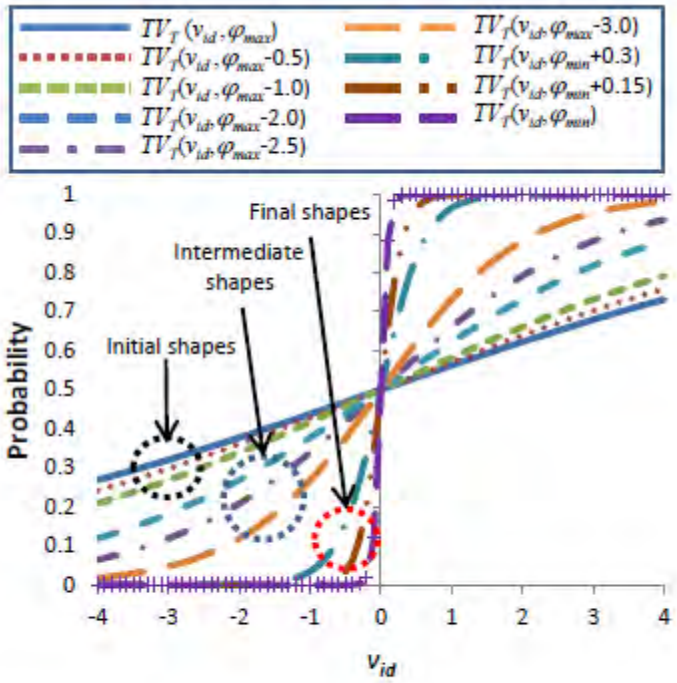


Figure : An illustration of different shapes of the time-varying transfer function with different values of the control parameter φ.