STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **14**, November 2025, pp 0–14.
Published online in International Academic Press (www.IAPress.org)

Solving 0 –1 knapsack problem by an improved binary monarch butterfly algorithm

Ghalya Tawfeeq Basheer [1], Lamyaa Jasim Mohammed[1], Zakariya Yahya Algamal [2,*]

[1]*Department of Operations Research and Intelligent Techniques, University of Mosul, Mosul, Iraq*
[2]*Department of Statistics and Informatics, University of Mosul, Mosul, Iraq*

Abstract The binary monarch butterfly optimization algorithm (BMBOA) is a meta-heuristic algorithm that has been applied widely in combinational optimization problems. Binary knapsack problem has received considerable attention in the combinational optimization. In this paper, a new time-varying transfer function is proposed to improve the exploration and exploitation capability of the BMBOA with the best solution and short computing time. Based on small, medium, and high-dimensional sizes of the knapsack problem, the computational results reveal that the proposed time-varying transfer functions obtain the best results not only by finding the best possible solutions but also by yielding short computational times. Compared to the standard transfer functions, the efficiency of the proposed time-varying transfer functions is superior, especially in the high-dimensional sizes.

Keywords 0-1 knapsack problem; monarch butterfly optimization algorithm; transfer function; time-varying parameter.

AMS 2010 subject classifications 46N10, 65K05

DOI: 10.19139/soic-2310-5070-2965

1. Introduction

The knapsack problem is considered as one of the NP-hard combinatorial optimization problems. The knapsack problem cannot be solved efficiently in a practically acceptable time scale using the exact algorithms because the computational time increases exponentially with the problem size. This leads to use approximate algorithms such as meta-heuristic algorithms to getting a good solutions, not necessarily optimal, in a reasonable time [1, 2, 37, 38].

The meta-heuristic algorithms are simple, flexible and they can be deal with the problems with different objective function properties, either discrete problems, continuous problems, or mixed problems [2]. These algorithms include genetic algorithm (GA) [3], particle swarm optimization (PSO) [4], artificial fish swarm algorithm (AFSA) [5], harmony search algorithm (HAS) [6, 7], gravitational search algorithm (GSA) [8], moth search algorithm (MSA) [9], cuckoo search algorithm (CSA) [10, 11], firefly algorithm (FA) [12], artificial bee colony algorithm (ABCA) [13], bat algorithm (BA) [14, 15], flower pollination algorithm (FPA) [1], and monarch butterfly optimization algorithm (MBOA) [16].

The monarch butterfly optimization algorithm is a new meta-heuristic algorithm that was developed by Wang, Deb [16] for solving continuous optimization problems. It mimics the migration behavior of monarch butterflies in nature. For solving different optimization problems, many studies have applied this algorithm. Wang, Deb [17] proposed a new version of MBOA with two strategies: greedy strategy and self-adaptive crossover operator. [18] introduced a new hybrid meta-heuristic algorithm, which combined artificial bee colony algorithm with elements

*Correspondence to: Zakariya Y. Algamal (Email: zakariya.algamal@uomosul.edu.iq). Department of Statistics and Informatics, University of Mosul, Mosul, Iraq.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

from monarch butterfly optimization algorithm. [19] proposed a self-adaptive strategy to dynamically adjust the butterflies in subpopulation 1 and subpopulation 2. Some works have been done for tackling knapsack problem using monarch butterfly optimization algorithm. [20] described a novel binary monarch butterfly optimization for solving knapsack problems, where the repair operator based on greedy optimization algorithm.[21] proposed a novel chaotic monarch butterfly optimization, where twelve classical chaotic maps were used in MBOA with applying Gaussian mutation operator. [22] introduced two strategies: neighborhood mutation based on crowding and Gaussian mutation into MBOA for solving discounted (0-1) knapsack problem.

In the binary monarch butterfly optimization algorithm, a transformation function is used to convert the continuous values generated from the algorithm into binary ones, and, therefore it is able to provide binary MBOA a sufficient amount to balance between exploration and exploitation [23, 31, 32, 33, 39, 40, 41].

In this paper, an efficient time-varying transfer function is proposed to solve the 0 –1 knapsack problem. The proposed transfer function is based on combining the S-shaped and V-shaped transfer functions with time-varying concept.

The remainder of this paper is organized as follows. Section 2 describes the basic 0 –1 knapsack problem. Section 3 introduces the monarch butterfly optimization algorithm. In Section 4, the proposed time-varying transfer function is presented. Section 5 presents and discusses the experimental results. In section 6, conclusions are drawn.

2. Knapsack problem

Knapsack problem is one of the NP-hard combinatorial optimization problems, which has been widely studied in operation research. Knapsack problem consists of a set of n items where each item i has a profit c_i, weight w_i, and maximum weight capacity M. The objective is to maximize the total profit of the selected items in the knapsack such that the total weights of these items are achieved by Eq.(2). Mathematically, the knapsack problem can be written as [24, 25]:

$$f(x) = \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$\begin{array}{l} \text{s.t.} \\ \sum_{i=1}^{n} w_i x_i \leq M \end{array} \tag{2}$$

where

$$x_i = \begin{cases} 1 & \text{if item i is selected} \\ 0 & \text{otherwise} \end{cases}$$

Using the penalty function, the knapsack problem can be written as follows:

$$Min\phi(x) = -f(x) + \lambda Max(0, h) \tag{3}$$

where $h = \sum_{i=1}^{n} w_i x_i - M$ and λ represents the penalty coefficient. In this paper λ is setting to 10^{10} for all tests. The penalty function can be described in Algorithm 1.

Penalty function

for *each* x_i
Calculate total weight of x_i *by* $\left(\sum_{i=1}^{n} w_i x_i\right)$
if $(totalweight \leq knapsackcapacity)$
$\phi(x) = -f(x)$
else
$\phi(x) = -f(x) + \lambda(totalweight - knapsackcapacity)$
end

Algorithm 1: Penalty function

3. Monarch Butterfly optimization (MBO)

Wang, [16] proposed a new meta-heuristic algorithm for continuous optimization problems called monarch butterfly optimization. It is inspired by simulating the migration behavior of the monarch butterflies from northern USA and southern Canada to Mexico every summer.

In MBO algorithm, the entire population can be divided into two subpopulations: subpopulation 1 and subpopulation 2 which lived in land 1 and land 2 respectively. The number of monarch butterflies in land 1 and land 2 is $NP1 = NP \times p$ and $NP2 = NP - NP1$ respectively, where $NP$is the size of entire population and p is the proportion of monarch butterflies in subpopulation 1. The monarch butterfly optimization algorithm has two main operators: the migration operator and the butterfly adjusting operator [16, 18].

3.1. Migration Operator

The migration process can be described as follows:

$$x_{i,k}^{t+1} = x_{r_1,k}^{t} \tag{4}$$

where $x_{i,k}^{t+1}$ is the kth element of x_i at generation $t+1$, which represents the position of the monarch butterfly i. $x_{r_1,k}^{t}$is the kth element of x_{r_1} at generation t, which represents the position of the monarch butterfly r_1. Monarch butterfly r_1 is randomly selected from subpopulation 1. If $r \leq p$, the element $k$in the newly generated monarch butterfly is generated by Eq.(4), else if $r > p$the element $k$is the newly generated monarch butterfly is generated by the following equation:

$$x_{i,k}^{t+1} = x_{r_2,k}^{t} \tag{5}$$

where $x_{r_2,k}^{t}$ is the kth element of x_{r_2} at generation t, that is the newly generated position of the monarch butterfly r_2. Monarch butterfly r_2 is randomly selected from subpopulation 2, where $r$can be computed as follows:

$$r = rand * peri \tag{6}$$

where $peri$ represents migration period and $rand$ is a random number in$(0, 1)$. Based on the above analyses, the migration operator can be expressed in Algorithm 2[16, 17].

Migration Operator
Begin
for *i=1:NP1*
for *k=1:d (all the elements in ith monarch butterfly)*
$r = rand * peri$
if $r \leq p$
selected a monarch butterfly r_1 *randomly in NP1*
Generate new monarch butterfly $x_{i,k}^{t+1}$ *by Eq.* (**??**)
else
selected a monarch butterfly r_2 *randomly in NP2*
Generate new monarch butterfly $x_{i,k}^{t+1}$ *by Eq.* (**??**)
end if
end for
end for
End

Algorithm 2: Migration Operator

3.2. Butterfly Adjusting Operator

In butterfly adjusting process, the position of the monarch butterflies in subpopulation 2 is updating by all the elements in monarch butterfly j, if $rand \leq p$ then it can be updated as:

$$x_{j,k}^{t+1} = x_{best,k}^{t} \tag{7}$$

where $x_{j,k}^{t+1}$ is the kth element of x_j at generation $t+1$, which represents the position of the monarch butterfly $j$and $x_{best,k}^{t}$ is the kth element of x_{best} at generation t, which represents the best monarch butterfly in land 1 and land 2. On the other hand, if the random value $rand > p$, it can be updated as:

$$x_{j,k}^{t+1} = x_{r_3,k}^{t} \tag{8}$$

where $x_{r_3,k}^{t}$is the kth element of x_{r_3}that is randomly selected in subpopulation 2. In addition of this condition, if $rand > BAR$, it can be updated as:

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha\left(dx_k - 0.5\right) \tag{9}$$

where $BAR$is the butterfly adjusting rate, $dx$is the walk step of the monarch butterfly j that can be calculated by performing Levy flight as:

$$dx = Levy\left(x_j^t\right) \tag{10}$$

and α is the weighting factor that can be calculated by following equation:

$$\alpha = \frac{S_{\max}}{t^2} \tag{11}$$

where $S_{\max}$ is the max walk step that an individual monarch butterfly can move in one step and t is the current generation. The butterfly adjusting operator can be described in Algorithm 3 [16, 17].

Butterfly Adjusting Operator

Begin
for *j=1:NP2*
for *k=1:d (all the elements in ith monarch butterfly)*
if $rand \leq p$
Generate new monarch butterfly $x_{j,k}^{t+1}$ *by Eq. (7)*
else
selected a monarch butterfly r_3 *randomly in NP2*
Generate new monarch butterfly $x_{j,k}^{t+1}$ *by Eq. (8)*
if $rand > BAR$
$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha\left(dx_k - 0.5\right)$
end if
end if
end for
end for
End

Algorithm 3: Butterfly Adjusting Operator

Based on the migration operator and the butterfly adjusting operator, the main steps of MBO algorithm can be described in Algorithm 4 [16, 17].

Monarch Butterfly Optimization Algorithm (MBOA)

Begin

Step 1: *Initialize the population of NP monarch butterfly individuals randomly , set migration period $peri$, the migration ratio p, butterfly adjusting rat BAR, and the max step $S_{\max}$. Set the maximum generation.*

Step 2: *Evaluate each monarch butterfly in the population.*

Step 3: while *(stopping criterion)*

Sort the monarch butterfly individuals based on fitness.

Divided butterfly individuals into two subpopulation (land 1 and land 2)

for *i=1:NP1*

Generate new subpopulation by migration operator Algorithm 2.

end

for *j=1:NP2*

Generate new subpopulation by Butterfly Adjusting Operator Algorithm 3.

end

Combine the two newly generated subpopulations into one population .

end

Step 4: *Find the best solution.*

End

Algorithm 4: Monarch Butterfly Optimization Algorithm

4. The proposed time-varying transfer functions

The standard monarch butterfly optimization algorithm was originally proposed to handle a continuous optimization problems. In discrete optimization problems, such as knapsack problem, the standard method cannot be applied directly to deal for such this problems. Therefore the transfer functions are usually employed to convert the continuous search space to discrete search space. There are two families of transfer functions: S-shaped and V-shaped transfer function which were proposed by [26]. The V-shaped transfer functions have also been studied by [27] to tackle the feature selection problem. The most common transfer functions from the S-shaped family is the sigmoid function [28, 29]:

$$S\left(x_i{}^t\right) = \frac{1}{1+e^{-x_i^t}} \tag{12}$$

$$x_i^t = \begin{cases} 1 & if \quad S\left(x_i^t\right) > rand \\ 0 & \text{O.W} \end{cases} \tag{13}$$

On the other hand, the inverse tangent hyperbolic function is the most common used transfer function from the V-shaped family. It is defined as:

$$V\left(x_i^t\right) = \left|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x_i^t\right)\right| \tag{14}$$

$$x_i^t = \begin{cases} 1 & if \quad V\left(x_i^t\right) > rand \\ 0 & \text{O.W} \end{cases} \tag{15}$$

The transfer function is the main key to the balance between exploitation and exploration [23, 30, 34, 35, 36]. In our proposed time-varying transfer function, a new control parameter φ is added in the original transfer function. This φ is a time-varying variable which starts with a large value and gradually decreases over time and it is expressed

in Eq.(16).

$$\varphi = \varphi_{\min} + (\varphi_{\max} - \varphi_{\min}) \times e^{-t} \tag{16}$$

where $\varphi_{\max}$and $\varphi_{\min}$are, respectively, the minimum and maximum values of the control parameter φ, and T is the maximum iteration of the BMBO. Accordingly, the two proposed transfer functions are defined as, respectively,

$$TVS\left({x_i}^t\right) = \frac{1}{1 + e^{\frac{-x_i^t}{\varphi}}} \tag{17}$$

and

$$TVV\left(x_i^t\right) = \left|\frac{2}{\pi}\arctan\left(\frac{\pi x_i^t}{2\varphi}\right)\right| \tag{18}$$

Figure 1 explains the behavior of the proposed time-varying transfer function for both Eq. (12) and Eq. (14), respectively. It is obvious that these proposed functions coverage to be a vertical line when iteration increasing.

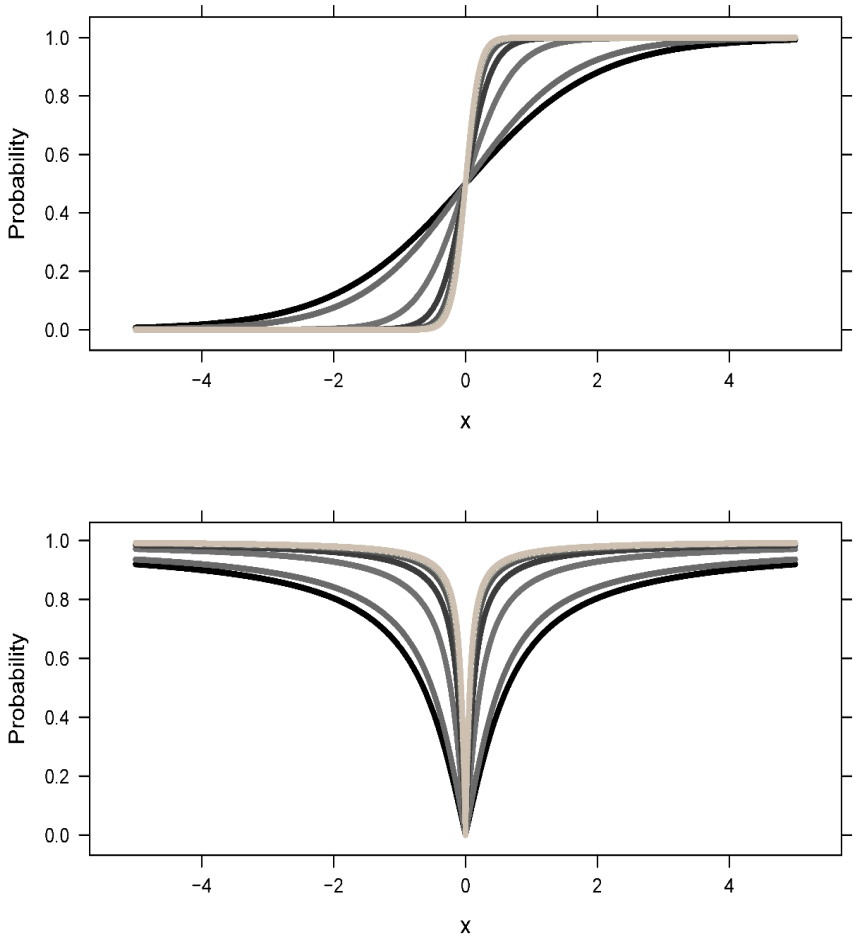

Figure 1. Explanation of the time-varying transfer function when $\varphi_{\max} = 2$and $\varphi_{\min} = 0.1$ during 10 iteration. The top panel is the sigmoid transfer function and the bottom panel is the inverse tangent hyperbolic transfer function.

5. Computational results

5.1. Parameter setting

For the binary monarch butterfly optimization algorithm, we set the parameters as follows: the population size =50, migration ratio = 5/12, migration period= 1.2, butterfly adjusting rate = 5/12, and Max step = 1. In addition, we used linear decreasing time varying with $\varphi_{\max} = 2$ and $\varphi_{\min} = 0.1$.

5.2. Comparison results

To verify the feasibility and effectiveness of the proposed time-varying transfer functions method for solving 0–1 Knapsack problem, three scales of the knapsack problem are considered: low, medium, and high-dimensional sizes. In this paper, all the results are obtained from 50 independent trials. The Best, Mean, Worst, SD, Mean iterations are reported as evaluation criteria. All of the computational experiments were conducted in Matlab 13a on a PC with an Intel Pentium Core i7-7500 processor (2.9 GHz) with 16GB of RAM in the Windows 10 OS.

5.2.1. Low size 0-1 KP The performance of improved algorithm is investigated to solve ten low scale 0-1 KP instances (kp-1 to kp-10), which are taken from [1, 14]. The dimensions in this case are ranging from 4 to 23. The information dimension, capacity, weights and profits for these ten instances are described in Table S1 (supplementary file). Table 1 shows the comparison results for all the used different transfer functions for the kp1 - kp10.

As observed from the results in Table 1, for the low scale knapsack problems, there is no difference among the results of using the proposed time-varying transfer functions and the standard transfer functions in terms of the best, worse, mean, and SD. The major difference among the performance of the proposed time-varying transfer functions and the standard transfer functions in not expected because of relatively small numbered items. Contrary, the proposed time-varying transfer functions give optimal results with less number of iterations. The mean iterations of the proposed time-varying transfer functions are obviously better than the standard transfer functions for kp4, kp5, kp8, kp9, and kp10 where the number of items is higher than the others. Moreover, comparing between the two proposed transfer function, the required iterations to get optimal solution using TVV is less than of TVS for kp4, kp5, kp6, kp8, kp9, and kp10.

5.2.2. Medium size 0-1 KP To further evaluate the performance of proposed time-varying transfer functions in medium size 0-1 Knapsack problem, ten medium size 0-1 KP instances (kp-11 to kp-20) are taken from [1, 14] in which the items are between 30 and 75. The description of these ten instances are described in Table S2 (supplementary file). Table 2 summarizes the comparison results for all the used different transfer functions.

Obviously, it is evident from Table 2 that the proposed time-varying transfer functions obtained the same best, worse, mean, and SD values as the standard transfer functions. From Tables 2, for the mean iterations, the proposed time-varying transfer functions are superior to the standard transfer functions on kp11 to kp20. This indicates that the proposed time-varying transfer functions is comparatively fast. For example, in kp20, the reduction in mean iteration of TVS function was 63.15% lower than that of S function. On the other hand, the reduction in mean iteration of TVV function was 57.94% lower than that of V function.

Further, it was noted that the v-shaped transfer functions are usually yielded the least iterations compared to S-shaped transfer functions. On the other hand, comparing between the two proposed transfer function, the required iterations to get optimal solution using TVV is less than of TVS for all the 0-1 Knapsack problems.

5.2.3. High-dimensional size 0-1 KP To further highlight the benefits of our proposed time-varying transfer functions, three cases have been investigated. The first case handles the uncorrelated problem (kp21 – kp25) where the weights w_i are uncorrelated with the profits c_i. Each w_i and c_i is randomly chosen from 5 to 20 and from 5 to 40, respectively. The second case handles the weakly correlated problem (kp26 – kp30). In this case, the weights w_i and the profits c_i can be expressed as follows: $w_i \in [5, 20]$ and $c_i \in [w_i - 5, w_i + 5]$. The third case handles the strongly correlated problem (kp31 – kp35). In this case, $w_i$and c_i can be calculated as: $w_i \in [5, 20]$ and $c_i \in [w_i + 5]$. The knapsack capacity for the kp-21-kp35 can be calculated as $M = 0.75 \times \sum_{i=1}^{n} w_i$. The dimension sizes varying

Table 1. Results obtained by the transfer functions for the low scale 0–1 KP

Instance	Transfer function	Best	Mean	Worst	SD	Mean iterations
kp-1	S	35	35	35	0	1
	V	35	35	35	0	1
	TVS	35	35	35	0	1
	TVV	35	35	35	0	1
kp-2	S	23	23	23	0	1
	V	23	23	23	0	1
	TVS	23	23	23	0	1
	TVV	23	23	23	0	1
kp-3	S	130	130	130	0	1
	V	130	130	130	0	1
	TVS	130	130	130	0	1
	TVV	130	130	130	0	1
kp-4	S	107	107	107	0	2.15
	V	107	107	107	0	1.08
	TVS	107	107	107	0	1
	TVV	107	107	107	0	1
kp-5	S	295	295	295	0	3.26
	V	295	295	295	0	2.11
	TVS	295	295	295	0	1
	TVV	295	295	295	0	1
kp-6	S	52	52	52	0	1.12
	V	52	52	52	0	1.04
	TVS	52	52	52	0	1
	TVV	52	52	52	0	1
kp-7	S	481.07	481.069	481.07	0	1
	V	481.07	481.069	481.07	0	1
	TVS	481.07	481.069	481.07	0	1
	TVV	481.07	481.069	481.07	0	1
kp-8	S	1025	1025	1025	0	2.64
	V	1025	1025	1025	0	1.91
	TVS	1025	1025	1025	0	1.67
	TVV	1025	1025	1025	0	1
kp-9	S	1024	1024	1024	0	2.86
	V	1024	1024	1024	0	1.08
	TVS	1024	1024	1024	0	1.02
	TVV	1024	1024	1024	0	1
kp-10	S	9767	9767	9767	0	5.31
	V	9767	9767	9767	0	3.14
	TVS	9767	9767	9767	0	4.02
	TVV	9767	9767	9767	0	2.35

from 100 to 2000 items. For all used transfer functions, the maximum iteration is set to 10000. Tables 3 – 5 reports the comparison results for all the used different transfer functions. Based on the obtained results, several points are concluded.

1. It can be seen that the proposed time-varying transfer functions significantly outperform the standard transfer functions on all evaluation measures including the best, mean, worst, and standard deviations.
2. As observed from the results, the proposed time-varying V-shaped transfer functions, TVV, can easily find the optimal values with small SD in all uncorrelated, weakly correlated, and strongly correlated problems.

Table 2. Results obtained by the transfer functions for the medium size 0–1 KP

Instance	Transfer function	Best	Mean	Worst	SD	Mean iterations
kp-11	S	1437	1437	1437	0	8.15
	V	1437	1437	1437	0	4.24
	TVS	1437	1437	1437	0	5.06
	TVV	1437	1437	1437	0	2.27
kp-12	S	1689	1689	1689	0	9.51
	V	1689	1689	1689	0	3.94
	TVS	1689	1689	1689	0	4.83
	TVV	1689	1689	1689	0	1.95
kp-13	S	1821	1821	1821	0	41.08
	V	1821	1821	1821	0	13.22
	TVS	1821	1821	1821	0	28.65
	TVV	1821	1821	1821	0	5.83
kp-14	S	2033	2033	2033	0	29.57
	V	2033	2033	2033	0	8.96
	TVS	2033	2033	2033	0	19.35
	TVV	2033	2033	2033	0	3.14
kp-15	S	2440	2440	2440	0	35.19
	V	2440	2440	2440	0	12.36
	TVS	2440	2440	2440	0	22.28
	TVV	2440	2440	2440	0	6.57
kp-16	S	2651	2648.5	2643	2.86	698.4
	V	2651	2651	2651	0	17.25
	TVS	2651	2651	2651	0	475.61
	TVV	2651	2651	2651	0	9.28
kp-17	S	2917	2917	2917	0	195.73
	V	2917	2917	2917	0	25.36
	TVS	2917	2917	2917	0	75.31
	TVV	2917	2917	2917	0	9.6
kp-18	S	2818	2815.6	2794	1.73	894.1
	V	2818	2818	2818	0	11.58
	TVS	2818	2818	2818	0	528.7
	TVV	2818	2818	2818	0	5.89
kp-19	S	3223	3221.6	3219	0.93	784.5
	V	3223	3223	3223	0	11.21
	TVS	3223	3223	3223	0	5.82
	TVV	3223	3223	3223	0	6.18
kp-20	S	3614	3614	3614	0	589.7
	V	3614	3614	3614	0	9.25
	TVS	3614	3614	3614	0	217.3
	TVV	3614	3614	3614	0	3.89

3. It is obvious that there is an improvement for searching the global optimal solution when using TVV compared to TVS. This leads to the performance dominance of the inverse tangent hyperbolic transfer function against the sigmoid transfer function.
4. The mean iteration values of time-varying V-shaped transfer functions, TVV, are obviously superior to S and V functions for all high-dimensional size problems.
5. Compared to the proposed time-varying V-shaped transfer functions, TVV is significantly improve the performance metrics with lower SD and mean iterations.

Table 3. MSE values when $n = 100$

Method			
	$\rho = 0.90$	$\rho = 0.95$	$\rho = 0.99$
SPRM	*3.612*	*4.699*	*6.216*
$HK1$	*1.39*	*2.695*	*2.905*
$HK2$	*1.431*	*1.819*	*2.772*
$K1$	*1.446*	*1.245*	*2.575*
$K2$	*1.158*	*2.142*	*2.629*
$K3$	*1.49*	*2.474*	*2.342*
$K4$	*1.332*	*2.316*	*2.368*
$K5$	*1.467*	*2.451*	*2.429*
$K6$	*1.374*	*2.358*	*2.604*
$K7$	*1.503*	*2.491*	*2.53*
$K8$	*1.15*	*2.135*	*2.709*
$K9$	*1.509*	*2.493*	*2.5*
$K10$	*1.183*	*2.167*	*2.392*
$K11$	*1.507*	*2.491*	*2.346*
SMA	*1.145*	*1.169*	*1.186*

Table 4. Comparison results of uncorrelated high-dimensional size 0–1 KP

Instance	*Dimension*	*Transfer function*	*Best*	*Mean*	*Worst*	*SD*	*Mean iterations*
kp-21	*100*	*S*	*2126*	*2120.8*	*2116*	*15.6*	*1085*
		V	*2126*	*2126*	*2126*	*0*	*69*
		TVS	*2126*	*2126*	*2126*	*0*	*520*
		TVV	*2126*	*2126*	*2126*	*0*	*34*
kp-22	*500*	*S*	*11025*	*11017.5*	*11012*	*18.5*	*3025*
		V	*11025*	*11024.2*	*11023*	*1.46*	*127*
		TVS	*11025*	*11023.8*	*11022*	*2.18*	*1582*
		TVV	*11025*	*11025*	*11025*	*0*	*65*
kp-23	*1000*	*S*	*21963*	*21958.6*	*21950*	*17.1*	*6853*
		V	*21968*	*21967.1*	*21965*	*2.01*	*839*
		TVS	*21969*	*21966.9*	*21963*	*4.92*	*2976*
		TVV	*21969*	*21969*	*21969*	*0*	*491*
kp-24	*1500*	*S*	*32633*	*32626.2*	*32620*	*20.65*	*5628*
		V	*32639*	*32637.8*	*32636*	*2.61*	*1957*
		TVS	*32637*	*32635.4*	*32634*	*3.48*	*3273*
		TVV	*32640*	*32639.2*	*32638*	*0.34*	*978*
kp-25	*2000*	*S*	*43711*	*43705*	*43692*	*31.6*	*8854*
		V	*43725*	*43722.6*	*43720*	*5.22*	*3761*
		TVS	*43723*	*43720.7*	*43718*	*3.67*	*5842*
		TVV	*43726*	*43725*	*43722*	*1.93*	*2072*

Table 5. Comparison results of weakly correlated high-dimensional size 0–1 KP

Instance	*Dimension*	*Transfer function*	*Best*	*Mean*	*Worst*	*SD*	*Mean iterations*
kp-26	*100*	*S*	*2015*	*2012.3*	*2009*	*4.29*	*926*
		V	*2015*	*2015*	*2015*	*0*	*48*
		TVS	*2015*	*2015*	*2015*	*0*	*352*
		TVV	*2015*	*2015*	*2015*	*0*	*25*
kp-27	*500*	*S*	*10450*	*10447.2*	*10446*	*6.41*	*1854*
		V	*10450*	*10449.3*	*10448*	*0.84*	*98*
		TVS	*10450*	*10447.8*	*10446*	*1.68*	*851*
		TVV	*10450*	*10450*	*10450*	*0*	*49*
kp-28	*1000*	*S*	*20856*	*20852.1*	*20849*	*6.85*	*3458*
		V	*20856*	*20854.6*	*20853*	*0.93*	*215*
		TVS	*20856*	*20854*	*20852*	*2.24*	*2851*
		TVV	*20856*	*20856*	*20856*	*0*	*137*
kp-29	*1500*	*S*	*31625*	*31620.3*	*31618*	*8.02*	*4183*
		V	*31630*	*31629.5*	*31626*	*1.85*	*1957*
		TVS	*31632*	*31628.7*	*31626*	*2.94*	*3273*
		TVV	*31632*	*31631.4*	*31631*	*0.26*	*895*
kp-30	*2000*	*S*	*42050*	*42046*	*42041*	*10.93*	*7794*
		V	*42055*	*42053.1*	*42049*	*2.04*	*2536*
		TVS	*42057*	*42051.6*	*42047*	*7.32*	*4582*
		TVV	*42057*	*42056*	*42054*	*1.34*	*1057*

Table 6. Comparison results of strongly correlated high-dimensional size 0–1 KP

Instance	*Dimension*	*Transfer function*	*Best*	*Mean*	*Worst*	*SD*	*Mean iterations*
kp-31	*100*	*S*	*2669*	*2669*	*2669*	*0*	*283*
		V	*2669*	*2669*	*2669*	*0*	*36*
		TVS	*2669*	*2669*	*2669*	*0*	*107*
		TVV	*2669*	*2669*	*2669*	*0*	*12*
kp-32	*500*	*S*	*13657*	*13654.1*	*13652*	*0.62*	*564*
		V	*13657*	*13657*	*13657*	*0*	*50*
		TVS	*13657*	*13657*	*13657*	*0*	*322*
		TVV	*13657*	*13657*	*13657*	*0*	*28*
kp-33	*1000*	*S*	*27164*	*27162.5*	*27159*	*1.29*	*921*
		V	*27166*	*27164.6*	*27164*	*0.90*	*127*
		TVS	*27166*	*27164.4*	*27162*	*0.12*	*619*
		TVV	*27166*	*27166*	*27166*	*0*	*88*
kp-34	*1500*	*S*	*40461*	*40459.8*	*40455*	*2.58*	*1766*
		V	*40466*	*40465*	*40463*	*1.61*	*293*
		TVS	*40468*	*40466.3*	*40464*	*1.58*	*835*
		TVV	*40468*	*40468*	*40468*	*0*	*146*
kp-35	*2000*	*S*	*42050*	*42048.9*	*42042*	*3.01*	*3905*
		V	*42054*	*42053.1*	*42048*	*2.18*	*559*
		TVS	*42057*	*42055*	*42051*	*2.33*	*2376*
		TVV	*42057*	*42056.2*	*42055*	*0.89*	*381*

6. Conclusion

In this paper, a time-varying transfer function was proposed to improve the exploration and exploitation capability of the binary monarch butterfly optimization algorithm in solving the 0–1 KP problem efficiently. The experimental results show that the introduction of time-varying parameter in the transfer function can improve the performance of binary MBOA in solving small, medium, and high-dimensional sizes 0–1 KP problems. Additionally, the experimental results show that proposed time-varying V-shaped transfer function outperforms the S-shaped transfer function in terms of the best, worse, mean, SD, and the mean iterations.

Appendix

Table 7. The description of the low size 0-1 KP instances

Instance	dimension	capacity M	weights W	profits C
kp-1	4	20	w=[6 5 9 7]	c=[9 11 13 15]
kp-2	4	11	w=[2 4 6 7]	c=[6 10 12 13]
kp-3	5	80	w=[15 20 17 8 31]	c=[33 24 36 37 12]
kp-4	7	50	w=[31 10 20 19 4 3 6]	c=[70 20 39 37 7 5 10]
kp-5	10	269	w=[95 4 60 32 23 72 80 62 65 46]	c=[55 10 47 5 4 50 8 61 85 87]
kp-6	10	60	w=[30 25 20 18 17 11 5 2 1 1]	c=[20 18 17 15 15 10 5 3 1 1]
kp-7	15	375	w=[56.358531 80.874050 47.987304 89.596240 74.660482 85.894345 51.353496 1.498459 36.445204 16.589862 44.569231 0.466933 37.788018 57.118442 60.716575]	c=[0.125126 19.330424 58.500931 35.029145 82.284005 17.410810 71.050142 30.399487 9.140294 14.731285 98.852504 11.908322 0.891140 53.166295 60.176397]
kp-8	20	879	w=[84 83 43 4 44 6 82 92 25 83 56 18 58 14 48 70 96 32 68 92]	c=[91 72 90 46 55 8 35 75 61 15 77 40 63 75 29 75 17 78 40 44]
kp-9	20	878	w=[92 4 43 83 84 68 92 82 6 44 32 18 56 83 25 96 70 48 14 58]	c=[44 46 90 72 91 40 75 35 8 54 78 40 77 15 61 17 75]
kp-10	23	10000	w=[983 982 981 980 979 978 488 976 972 486 486 972 972 485 485 969 966 483 964 963 961 958 959]	c=[981 980 979 978 977 976 487 974 970 485 485 970 970 484 484 976 974 482 962 961 959 958 857]

Table 8. Medium size 0–1 KP test problems

Instance	*dimension*	*capacity M*	*weights W*	*profits C*
kp-11	*30*	*577*	*w=[46 17 35 1 26 17 17 48 38 17 32 21 29 48 31 8 42 37 6 9 15 22 27 14 42 40 14 31 6 34]*	*c=[57 64 50 6 52 6 85 60 70 65 63 96 18 48 85 50 77 18 70 92 17 43 5 23 67 88 35 3 91 48]*
kp-12	*35*	*655*	*w=[7 4 36 47 6 33 8 35 32 3 40 50 22 18 3 12 30 31 13 33 4 48 5 17 33 26 27 19 39 15 33 47 17 41 40]*	*c=[35 67 30 69 40 40 21 73 82 93 52 20 61 20 42 86 43 93 38 70 59 11 42 93 6 39 25 23 36 93 51 81 36 46 96]*
kp-13	*40*	*819*	*w=[28 23 35 38 20 29 11 48 26 14 12 48 35 36 33 39 30 26 44 20 13 15 46 36 43 19 32 2 47 24 26 39 17 32 17 16 33 22 6 12]*	*c=[13 16 42 69 66 68 1 13 77 85 75 95 92 23 51 79 53 62 56 74 7 50 23 34 56 75 42 51 13 22 30 45 25 27 90 59 94 62 26 11]*
kp-14	*45*	*907*	*w=[18 12 38 12 23 13 18 46 1 7 20 43 11 47 49 19 50 7 39 29 32 25 12 8 32 41 34 24 48 30 12 35 17 38 50 14 47 35 5 13 47 24 45 39 1]*	*c=[98 70 66 33 2 58 4 27 20 45 77 63 32 30 8 18 73 9 92 43 8 58 84 35 78 71 60 38 40 43 43 22 50 4 57 5 88 87 34 98 96 99 16 1 25]*
kp-15	*50*	*882*	*w=[15 40 22 28 50 35 49 5 45 3 7 32 19 16 40 16 31 24 15 42 29 4 14 9 29 11 25 37 48 39 5 47 49 31 48 17 46 1 25 8 16 9 30 33 18 3 3 3 4 1]*	*c=[78 69 87 59 63 12 22 4 45 33 29 50 19 94 95 60 1 91 69 8 100 32 81 47 59 48 56 18 59 16 45 54 47 84 100 98 75 20 4 19 58 63 37 64 90 26 29 13 53 83]*
kp-16	*55*	*1050*	*w=[27 15 46 5 40 9 36 12 11 11 49 20 32 3 12 44 24 1 24 42 44 16 12 42 22 26 10 8 46 50 20 42 48 45 43 35 9 12 22 2 14 50 16 29 31 46 20 35 11 4 32 35 15 29 16]*	*c=[98 74 76 4 12 27 90 98 100 35 30 19 75 72 19 44 5 66 79 87 79 44 35 6 82 11 1 28 95 68 39 86 68 61 44 97 83 2 15 49 59 30 44 40 14 96 37 84 5 43 8 32 95 86 18]*
kp-17	*60*	*1006*	*w=[7 13 47 33 38 41 3 21 37 7 32 13 42 42 23 20 49 1 20 25 31 4 8 33 11 6 3 9 26 44 39 7 4 34 25 25 16 17 46 23 38 10 5 11 28 34 47 3 9 22 17 5 41 20 33 29 1 33 16 14]*	*c=[81 37 70 64 97 21 60 9 55 85 5 33 71 87 51 100 43 27 48 17 16 27 76 61 97 78 58 46 29 76 10 11 74 36 59 30 72 37 72 100 9 47 10 73 92 9 52 56 69 30 61 20 66 70 46 16 43 60 33 84]*
kp-18	*65*	*1319*	*w=[47 27 24 27 17 17 50 24 38 34 40 14 15 36 10 42 9 48 37 7 43 47 29 20 23 36 14 2 48 50 39 50 25 7 24 38 34 44 38 31 14 17 42 20 5 44 22 9 1 33 19 19 23 26 16 24 1 9 16 38 30 36 41 43 6]*	*c=[47 63 81 57 3 80 28 83 69 61 39 7 100 67 23 10 25 91 22 48 91 20 45 62 60 67 27 43 80 94 47 31 44 31 28 14 17 50 9 93 15 17 72 68 36 10 1 38 79 45 10 81 66 46 54 53 63 65 20 81 20 42 24 28 1]*

Table 9. Medium size 0–1 KP test problems (continue)

Instance	*dimension*	*capacity M*	*weights W*	*profits C*
kp-19	*70*	*1426*	*w=[4 16 16 2 9 44 33 43 14 45 11 49 21 12 41 19 26 38 42 20 5 14 40 47 29 47 30 50 39 10 26 33 44 31 50 7 15 24 7 12 10 34 17 40 28 12 35 3 29 50 19 28 47 13 42 9 44 14 43 41 10 49 13 39 41 25 46 6 7 43]*	*c=[66 76 71 61 4 20 34 65 22 8 99 21 99 62 25 52 72 26 12 55 22 32 98 31 95 42 2 32 16 100 46 55 27 89 11 83 43 93 53 88 36 41 60 92 14 5 41 60 92 30 55 79 33 10 45 3 68 12 20 54 63 38 61 85 71 40 58 25 73 35]*
kp-20	*75*	*1433*	*w=[24 45 15 40 9 37 13 5 43 35 48 50 27 46 24 45 2 7 38 20 20 31 2 20 3 35 27 4 21 22 33 11 5 24 37 31 46 13 12 12 41 36 44 36 34 22 29 50 48 17 8 21 28 2 44 45 25 11 37 35 24 9 40 45 8 47 1 22 1 12 36 35 14 17 5]*	*c=[2 73 82 12 49 35 78 29 83 18 87 93 20 6 55 1 83 91 71 25 59 94 90 61 80 84 57 1 26 44 44 88 7 34 18 25 73 29 24 14 23 82 38 67 94 43 61 97 37 67 32 89 30 30 91 50 21 3 18 31 97 79 68 85 43 71 49 83 44 86 1 100 28 4 16]*

REFERENCES

1. Abdel-Basset, M., D. El-Shahat, and I. El-Henawy, *Solving 0–1 knapsack problem by binary flower pollination algorithm.* , Neural Computing and Applications, 2018.
2. Yang, X.-S., *Cuckoo Search and Firefly Algorithm Theory and Applications* , Springer International Publishing Switzerland, 2014.
3. Zhao, J.F., et al., *Genetic Algorithm Bassed on Greedy Strategy in the 0-1 Knapsack problem* , Third international conference on genetic and evolutionary computing, 2009.
4. Beheshti, Z., *S.M. Shamsuddin, and S.S. Yuhaniz, Binary Accelerated Particle Swarm Algorithm (BAPSA) for discrete optimization problems.*, Journal of Global Optimization, vol. 57, no. 2 , p. 549-573, 2012.
5. Azad, M.A.K., *A.M.A.C. Rocha, and E.M.G.P. Fernandes, A simplified binary artificial fish swarm algorithm for 0–1 quadratic knapsack problems* , Journal of Computational and Applied Mathematics, vol. 259 , p. 897-904, 2014.
6. Kong, X., et al., *A simplified binary harmony search algorithm for large scale 0–1 knapsack problems* , Expert Systems with Applications, vol. 42 , no.12, p. 5337-5355, 2015.
7. Tuo, S., *L. Yong, and F. Deng, A novel harmony search algorithm based on teaching-learning strategies for 0-1 knapsack problems* , ScientificWorldJournal, vol. 2014, p. 637412. 7 , 2014.
8. Sajedi, H. and S.F. Razavi, *DGSA: discrete gravitational search algorithm for solving knapsack problem* , Operational Research, vol. 17, no. 2 p. 563-591, 2016.
9. Feng, Y., H. An, and X. Gao, *The Importance of Transfer Function in Solving Set-Union Knapsack Problem Based on Discrete Moth Search Algorithm* , Mathematics, vol. 7 ,no. 1 p. 17 , 2018.
10. Feng, Y., K. Jia, and Y. He, *An improved hybrid encoding cuckoo search algorithm for 0-1 knapsack problems* , Comput Intell Neurosci, vol. 2014 , p. 970456. 2014.
11. Feng, Y., et al., *An effective hybrid cuckoo search algorithm with improved shuffled frog leaping algorithm for 0-1 knapsack problems* , Comput Intell Neurosci, vol. 2014, p. 857254, 2014.
12. Feng, Y., G.-G. Wang, and L. Wang, *Solving randomized time-varying knapsack problems by a novel global firefly algorithm* , Engineering with Computers, vol. 34 ,no.3 , p. 621-635, 2017.
13. He, Y., et al., *A novel binary artificial bee colony algorithm for the set-union knapsack problem* , Future Generation Computer Systems, vol . 78 , p. 77-86, 2018.
14. Rizk-Allah, R.M. and A.E. Hassanien, *New binary bat algorithm for solving 0–1 knapsack problem* , Complex Intelligent Systems, vol, 4 , no. 1, p. 31-53, 2017.
15. Zhou, Y., L. Li, and M. Ma, *A Complex-valued Encoding Bat Algorithm for Solving 0–1 Knapsack Problem* , Neural Processing Letters, vol . 44 , no. 2 , p. 407-430 , 2015.
16. Wang, G.-G., S. Deb, and Z. Cui, *Monarch butterfly optimization* , Neural Computing and Applications, 2015.
17. Wang, G.-G., et al., *A new monarch butterfly optimization with an improved crossover operator* , Operational Research, vol .18 , no. 3 ,p. 731-755, 2016.
18. Ghanem, W.A.H.M. and A. Jantan, *Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems* , Neural Computing and Applications, vol. 30, no. 1, p. 163-181, 2016.

19. Hu, H., et al., *Improving Monarch Butterfly Optimization Algorithm with Self-Adaptive Population* , Algorithms, vol. 11 , no. 5, p. 71, 2018.
20. Feng, Y., et al., *Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization* , Neural Computing and Applications, vol. 28 ,no. 7 , p. 1619-1634, 2015.
21. Feng, Y., et al., *Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation* , Memetic Computing, vol. 10 , no. 2 , p. 135-150 ,2016.
22. Feng, Y., et al., *Multi-strategy monarch butterfly optimization algorithm for discounted 0-1 knapsack problem* , Neural Computing and Applications, vol .30 , no. 10 , p. 3019-3036, 2017.
23. Islam, M.J., X. Li, and Y. Mei, *A time-varying transfer function for balancing the exploration and exploitation ability of a binary PSO* , Applied Soft Computing, vol. 59 , p. 182-196 , 2017.
24. Cao, J., et al., *A modified artificial bee colony approach for the 0-1 knapsack problem* , Applied Intelligence, vol. 48 , no. 6 , p. 1582-1595. 2017.
25. Abdel-Basset, M., D. El-Shahat, and A.K. Sangaiah, *A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem* , International Journal of Machine Learning and Cybernetics, 2017.
26. Mirjalili, S. and A. Lewis, *S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization* , Swarm and Evolutionary Computation, vol . 9 ,p. 1-14 , 2013.
27. Teng, X., H. Dong, and X. Zhou, *Adaptive feature selection using v-shaped binary particle swarm optimization* , PLoS One, vol . 12 ,no. 3 , p. e0173907, 2017.
28. Tilahun, S.L. and J.M.T. Ngnotchouye, *Firefly algorithm for discrete optimization problems: A survey* , KSCE Journal of Civil Engineering, vol. 21, no. 2 , p. 535-545, 2017.
29. Bhattacharjee, K.K. and S.P. Sarmah, *Modified swarm intelligence based techniques for the knapsack problem* , . Applied Intelligence, vol . 46 , no.1 , p. 158-179 , 2016.
30. Mafarja, M., et al., *Binary dragonfly optimization for feature selection using time-varying transfer functions* , Knowledge-Based Systems, vol . 161 , p. 185-204 , 2018.
31. Almishlih1, Zaynab Ayham, Qasim, Omar Saber, Algamal, Zakariya Yahya *Binary Arithmetic Optimization Algorithm Using a New Transfer Function for Fusion Modeling*, Fusion: Practice and Applications, vol. 18, no.2, p. 157-168, 2025
32. Kahya, M. A., Altamir, S. A., & Algamal, Z. Y. *Improving whale optimization algorithm for feature selection with a time-varying transfer function* , Numerical Algebra, Control and Optimization, vol. 11, no. 1 , p. 87-98 , 2020.
33. Algamal, Z. Y., Qasim, M. K., Lee, M. H., Ali, H. T. M. *QSAR model for predicting neuraminidase inhibitors of influenza A viruses (H1N1) based on adaptive grasshopper optimization algorithm*, SAR and QSAR in Environmental Research, vol.31, no.11, p.803-814, 2020.
34. Al-Fakih, A. M., Algamal, Z. Y., Lee, M. H., Aziz, M., Ali, H. T. M. *QSAR classification model for diverse series of antifungal agents based on improved binary differential search algorithm*, SAR and QSAR in Environmental Research, vol.30, no.2, p.131-143, 2019.
35. Al-Fakih, A. M., Qasim, M. K., Algamal, Z. Y., Alharthi, A. M., Zainal-Abidin, M. H. *QSAR classification model for diverse series of antifungal agents based on binary coyote optimization algorithm* SAR and QSAR in Environmental Research, vol.34, no.4, p. 285-298, 2023.
36. Alharthi, A. M., Kadir, D. H., Al-Fakih, A. M., Algamal, Z. Y., Al-Thanoon, N. A., Qasim, M. K. *Improving golden jackel optimization algorithm: An application of chemical data classification*, Chemometrics and Intelligent Laboratory Systems, vol.250, 105149, 2024.
37. Basheer, G.T. and Z.Y. Algamal, *Improving Flower Pollination Algorithm for Solving 0–1 Knapsack Problem*, Journal of Physics: Conference Series, vol.1879, no. 2, 2021.
38. Al-Thanoon, N.A., O.S. Qasim, and Z.Y. Algamal, *A New Hybrid Pigeon-Inspired Optimization Algorithm for Solving Multidimensional Knapsack Problems*, in 2021 7th International Conference on Contemporary Information Technology and Mathematics (ICCITM). p. 226-229.2021.
39. Al-Fakih, A. M., Algamal, Z. Y., Lee, M. H., Aziz, M., Ali, H. T. M., *A QSAR model for predicting antidiabetic activity of dipeptidyl peptidase-IV inhibitors by enhanced binary gravitational search algorithm*, SAR and QSAR in Environmental Research. vol. 30, no. 6, p. 403-416.2019.
40. Algamal, Z. Y., *A particle swarm optimization method for variable selection in beta regression model*, Electronic Journal of Applied Statistical Analysis. vol. 12, no. 2, p. 403-416. 2019.
41. Alkhateeb, A. N., Algamal, Z. Y., *Variable selection in gamma regression model using chaotic firefly algorithm with application in chemometrics*, Electronic Journal of Applied Statistical Analysis. vol. 14, no. 1, p. 266-276. 2021.