Research Article
Journal of Intelligent Systems: Theory and Applications 6(2) (2023) 108-118
DOI: 10.38016/jista.1200225

Binary Honey Badger Algorithm for 0-1 Knapsack Problem

Gülşen Orucova Büyüköz[1*], Hüseyin Haklı[2]

[1] Department of Mathematics And Computer Science, Necmettin Erbakan University, Konya, Türkiye

[2] Department of Computer Engıneerıng, Necmettin Erbakan University, Konya, Türkiye

gorucova@erbakan.edu.tr, hhakli@erbakan.edu.tr

Abstract
Honey Badger Algorithm (HBA) is one of the recently proposed optimization techniques inspired by the foraging behavior of honey badger. Although it has been successfully applied in solving continuous problems, the algorithm cannot be implemented directly in binary problems. A binary version of HBA is proposed in this study for the 0-1 Knapsack Problem (0-1 KP). To adapt the binary version of HBA, V- Shaped, S-Shaped, U-Shaped, T-Shaped, Tangent Sigmoid, O-Shaped, and Z-Shaped transfer functions are used. Each transfer function was tested by computational experiments over 25 instances of 0-1 KP and compared results. According to the results obtained, it was observed that O1 was the best TF among 25 TFs. In addition, the proposed algorithm was compared with three different binary variants, such as BPSO, MBPSO, and NGHS. Experimental results and comparison show that the proposed method is a promising and alternative algorithm for 0-1 KP problems.

Keywords: Binary Honey Badger Algorithm, 0-1 Knapsack Problems, Transfer Functions, Binary Optimization.

0-1 Sırt Çantası Problemi İçin İkili Bal Porsuğu Algoritması

Öz
Bal Porsuğu Algoritması (HBA), son zamanlarda önerilen optimizasyon tekniklerinden biridir ve bal porsuğunun yiyecek arama davranışından esinlenmiştir. Sürekli problemlerin çözümünde başarılı bir şekilde uygulanmasına rağmen, algoritma doğrudan ikili problemlerde uygulanamaz. Bu çalışmada 0-1 Sırt Çantası Problemi (0-1 KP) için HBA'nın ikili versiyonu önerilmiştir. HBA'nın ikili versiyonunu uyarlamak için V-Şekilli, S-Şekilli, U-Şekilli, T-Şekilli, Tanjant Sigmoid, O-Şekilli, Z-Şekilli transfer fonksiyonları (TF) kullanılmaktadır. Her transfer fonksiyonu 25 0-1 KP problemi için test edilmiş ve sonuçlar karşılaştırılmıştır. Elde edilen sonuçlara göre 25 TF arasından en iyi TF'nin O1 olduğu görülmüştür. Ayrıca bu algoritma BPSO, MBPSO, NGHS gibi üç farklı ikili varyant ile karşılaştırılmıştır. Deneysel sonuçlar ve karşılaştırmalar önerilen yöntemin 0-1 KP problemleri için umut verici ve alternatif bir araç olduğunu göstermektedir.

Anahtar kelimeler: İkili Bal Porsuğu Algoritması, 0-1 Sırt Çantası Problemleri, Transfer Fonksiyonları, İkili Optimizasyon.

1. Introduction

Many real-world problems, such as scheduling problems (Kaya et al., 2020), (Deng, Xu and Zhao, 2019) placement of wind turbines (Deng, Xu and Zhao, 2019; Hakli, 2019), vehicle routing (Halat and Ozkan, 2021), optimization of seismic isolation parameters (Çerçevik and Avşar, 2020), etc. use meta-heuristic optimization methods due to traditional solution methods that are insufficient. One of these problems is the knapsack problem.

0-1 KP has a prominent part in many real-world applications such as decision-making processes, exploiting resources optimally, database storage, investment strategies, and network formation. This problem is one of the fundamental NP-hard problems that achieves the maximum profit and the minimum cost in combinatorial optimization (Bansal and Deep, 2012), (Rooderkerk and van Heerde, 2016).

Recently, 0-1 KP has been applied by many swarm-intelligence and population-based optimization algorithms. Meta-heuristic optimization algorithms presented for continuous search space must be adapted to binary structure for tackling discrete optimization problems. Transfer functions are widely preferred approaches to discretization of a continuous algorithm. The binary Particle Swarm Optimization algorithm was modified (MBPSO) and used to solve some 0-1 KPs and

* Corresponding Author
E-mail: gorucova@erbakan.edu.tr

Recieved : 6 Nov 2022
Revision : 9 Mar 2023
Accepted : 1 Jun 2023

multidimensional KPs, results were compared with Binary PSO algorithm (Bansal and Deep, 2012). Cuckoo Search (CS) algorithm was transformed to a binary version using the sigmoid function (Gherboudj, Layeb and Chikhi, 2012). The binary Monkey Algorithm (BMA) was developed by (Zhou, Chen and Zhou, 2016). BMA was employed with the greedy algorithm to strengthen the local search ability to overcome fall into local optimal solutions. Also, 0-1 KP was considered by Binary Monarch Butterfly Optimization (BMBO) using S-shaped transfer functions and repair operator (Feng *et al.*, 2017). Social Spider Algorithm was adapted to binary search space with sigmoid function and repair algorithm to overcome 0-1 KPs (Nguyen, Wang and Truong, 2017). In another study (Rizk-Allah and Hassanien, 2018), Binary Bat Algorithm (BBA) was established based on the V-shaped and S-shaped transfer functions and used to cope with 0-1 KP. The Differential Evolution Algorithm was designed to apply to binary problems and the binary version was tested on the 0-1 KPs (Ismail M. Ali, 2018). A binary variant of Flower Pollination Algorithm (BFPA) with sigmoid transfer function was introduced, and repair operator and penalty function were employed to improve the solution quality (Abdel-Basset, El-Shahat and El-Henawy, 2019). Using V-Shaped and S-Shaped transfer functions, Marine Predators Algorithm (MPA) was moved from continuous to discrete space (Abdel-Basset *et al.*, 2021). The binary version of the Equilibrium Algorithm (BEA) was proposed for tackling 0–1 KP. Because the standard Equilibrium Optimizer (EO) was presented to overcome continuous optimization problems, EO was transformed to BEO with V-Shaped and S-Shaped TFs. Results showed that among those transfer functions, V3 was the best one (Abdel-Basset, Mohamed and Mirjalili, 2021). Ali et al. proposed a new binary technique that makes a simple differential evolution algorithm adequate for solving binary optimization issues (Ali, Essam and Kasmarik, 2021). The Giza Pyramids Construction (GPC) algorithm was proposed with accumulative and multiplicative penalty functions to determine infeasible solutions in the binary version of GPC (Harifi, 2022). On the other hand binary version of Slime Mould Algorithm (SMA) was presented to convert a continuous variable to a binary by employing eight different transfer functions (Abdollahzadeh *et al.*, 2021). A Quantum Inspired Social Evolution Algorithm (QSE) (Pavithr and Gursaran, 2016) was obtained by hybridizing Social Evolution Algorithm with the QSE, and the method was compared with different algorithms. Cohort Intelligence (CI) (Kulkarni and Shabir, 2016) was inspired by individuals' social, natural and social learning to learn from each other. Several cases of 0–1 KP were applied using CI, and the various parameters influencing the solution quality were discussed. One of the global optimization strategies was the complex-valued encoding method. Zhou, Li, et al. applied the method to the bat algorithm, and the sigmoid function was used for obtaining the discrete value (Zhou, Li and Ma, 2016). In order to achieve the good solutions that increases the total value without overcapacity of knapsack by Grey Wolf Optimization (GWO) and K-means algorithm was merged and dealt with the complexity of the algorithm (Yassien *et al.*, 2017). Genetic Algorithm, Branch and Bound, Simulated Annealing, Dynamic Programming, Greedy Search algorithms were compared for obtained 0-1 KPs results and discussed in (Ezugwu *et al.*, 2019). Improved Whale Optimization Algorithm (IWOA) was performed by the sigmoid transfer function to convert the real-valued solutions into binary and combining the penalty function with the fitness function to evaluate performance single and multidimensional 0-1 KPs are solved (Abdel-Basset, El-Shahat and Sangaiah, 2019). Due to fact that Dragonfly Algorithm (DA) performs on continuous search space, angle modulation mechanism was used for DA to adapt the algorithm works in the binary space (Wang, Shi and Dong, 2021). Moreover, Hybrid Harmony Search Algorithm with distribution estimation was introduced (Liu *et al.*, 2022), Hybrid Rice Optimization (HRO) was merged with Binary Ant Colony Optimization (BACO) algorithm to increase the convergence speed and search efficiency (Shu *et al.*, 2022).

Although many meta-heuristic optimization methods have been applied to overcome the 0-1 KPs, HBA has not been used to this problem. However, HBA presents a successful performance for continuous optimization problems, but a remarkable binary version of HBA is not seen in literature (Hashim *et al.*, 2022). This paper proposes binary versions of HBA with transfer functions and applies to several 0-1 KPs. The rest of this paper are formed as follows: Section 2 explains 0-1 KPs and original HBA. Section 3 presents binary version of HBA and implementation of transfer functions. In Section 4, the experiment results and comparison of transfer functions are conducted. The last section covers the conclusion of the study and provides some possible future directions.

2. Materials and Method

2.1. 0-1 Knapsack Problem

The 0-1 KP problem, proposed by Dantzig (Dantzig, 1957), is based on the knapsack, which has a capacity $C > 0$ and contains a set of n items $(x_1, x_2, \dots, x_n)$. For each x_i item has $p_i > 0$ profit and $w_i > 0$ weight $(i = 1,2, \dots, n)$. If x_i item is selected x_i=1 and $x_i = 0$ if x_i is not selected into the knapsack. The goal of this issue is to achieve a maximum profit from the items selected for the knapsack and the weights of all chosen items must be less or equal to the capacity of the knapsack. Mathematically formulation is given below (Rooderkerk and van Heerde, 2016);

fitness function: $\max_{i} \sum_{i=1}^{n} x_i p_i$ (1)

subject to

$$\sum_{i=1}^{n} x_i w_i \leq C, \quad x_i \in \{0,1\}, i = 1,2,\ldots,n \tag{2}$$

2.2. Honey Badger Algorithm

Honey Badger Algorithm (HBA) is a search strategy used to solve mathematical optimization problems inspired by the honey badger's foraging behavior. This algorithm is proposed by Hashim et al. (Hashim *et al.*, 2022). The honey badger's digging and dynamic foraging behavior are formulated in the exploration and exploitation phases. In the case of digging, it uses its sense of smell to predict the location of prey; Once reached, it moves around the prey to catch the prey. In the case of honey, the honey badger takes the guide of the honey guide bird to find the beehive directly. The steps of the Honey Badger algorithm are given below.

The algorithm starts by generating a randomly population of candidate solutions with the help of the following equation.

$$x_i = lb_i + r_1(ub_i - lb_i) \tag{3}$$

where x_i is honey badger's $i$ts position, lb_i and ub_i are the lower and upper bounds of the search space, respectively. r_1 is a random number between 0 and 1. The other important notation is intensity (I) which is related to the concentration power of the prey and the distance between it and the prey. I_i is the odor intensity of the prey; if the odor is high, the honey badger will move quickly and vice versa. This odor intensity is inversely proportional to the distance of the honey badger from the prey. There S is the source power or concentration power, d_i is the distance between prey and honey badger, and r_2 is a random number between 0 and 1.

$$I_i = r_2 \frac{S}{4\pi d_i^2} \tag{4}$$

$$S = (x_i - x_{i+1})^2 \tag{5}$$

$$d_i = x_{prey} - x_i \tag{6}$$

The intensity factor (α) controls the time change randomness to provide a soft transition from exploration to exploitation. The decreasing factor α is updated with decreasing iterations in a random time. Where $C \geq 1$ is constant (default value is 2), t_{max} is maximum number of iterations.

$$\alpha = C.exp\left(\frac{-t}{t_{max}}\right) \tag{7}$$

For escaping from the local optimum algorithm is used an F flag that changes the search direction of the algorithm. In equation (9), the property of flag F is given.

The main phases of HBA are the digging phase and the honey phase. In first phase, honey badger draws the path in the form of Cardioid. This motion is simulated by equation (8)

$$\begin{aligned} x_{new} &= x_{prey} + F.\beta.I.x_{prey} \\ &+ F.r_3.\alpha.d_i.|cos(2\pi r_4).[1 - cos(2\pi r_5)]| \end{aligned} \tag{8}$$

where, x_{prey} is the prey position, x_{new} honey badger's new position, $\beta \geq 1$ (default 6) honey badger's ability to reach food, d_i is the distance between prey and the ith honey badger and r_3, r_4, r_5 are random numbers different from each other between 0 and 1. The F flag changes the search direction and is defined as follows.

$$F = \begin{cases} 1, & r_6 \leq 0.5 \\ -1, & else \end{cases} \tag{9}$$

where r_6 is a random value in range [0,1].

In the second phase honey badger's pursuit of the honey guide bird is shown by the equation below.

$$x_{new} = x_{prey} + F.r_7.\alpha.d_i \tag{10}$$

where r_7 is a random like r_6. For more information on HBA, see (Hashim *et al.*, 2022).

The flowchart of the HBA method is demonstrated in Figure 1.

2.3. Transfer Functions (TFs)

Selecting an appropriate transfer function is an important decision to increase efficiency, as transfer functions play a significant role in converting the continuous search space to binary space. PSO algorithm is adapted to binary space with the help of the sigmoid function, which is defined as follows (J. Kennedy, 1997):

$$sigm\left(v_i^d(t)\right) = \frac{1}{1 + e^{-v_i^d(t)}} \tag{11}$$

where $v_i^d(t)$ is the following velocity of the i^{th} particle in the d^{th} dimension. The position, $x_i^d(t+1)$ is updated by the following equation:

$$x_i^d(t+1) = \begin{cases} 1, & if\ r \geq sigm\left(v_i^d(t)\right) \\ 0, & else \end{cases} \tag{12}$$

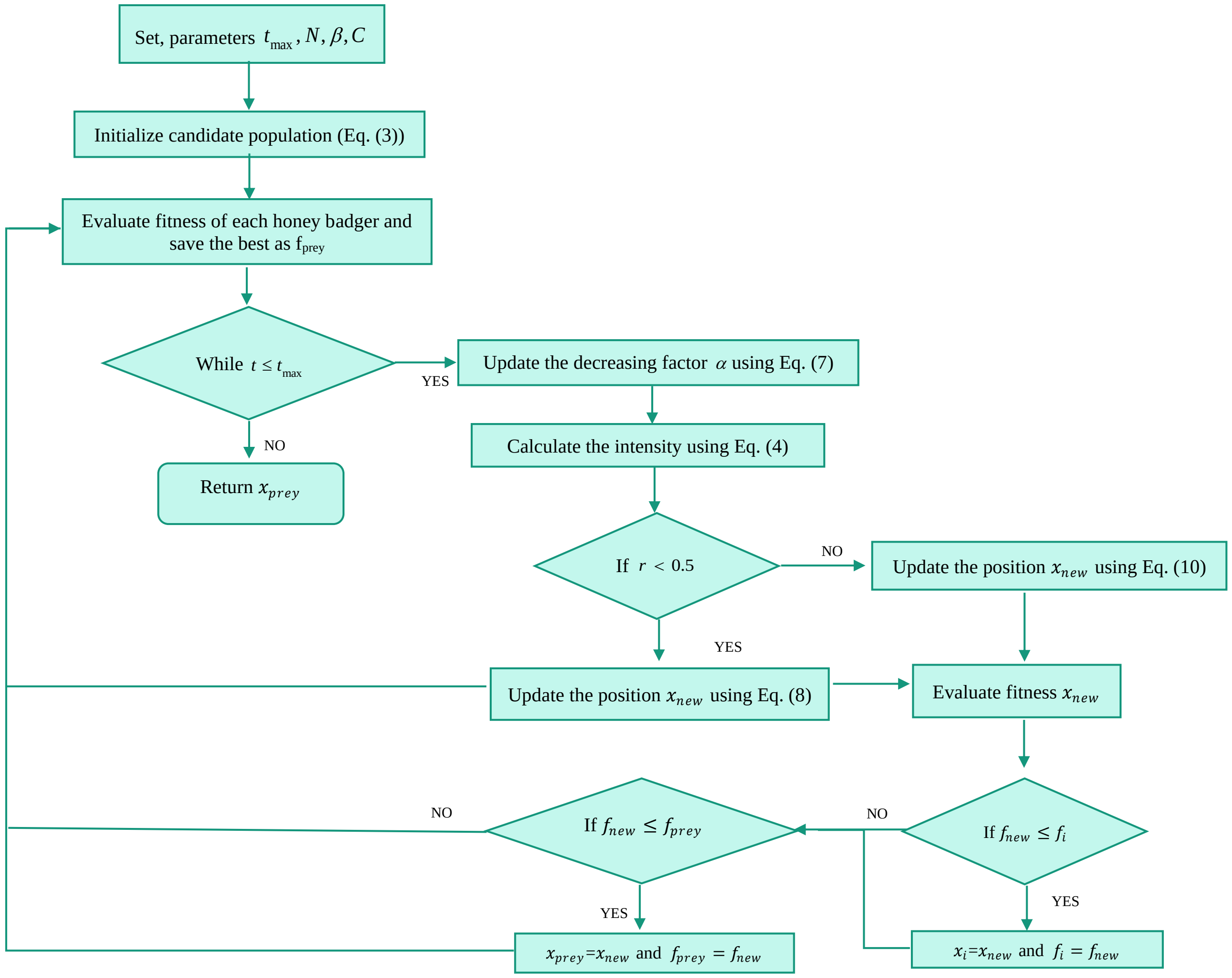

Figure 1. Flowchart of the Honey Badger Algorithm (HBA)

where r is a number between 0 and 1, which is generated with uniform distribution

Seyedali Mirjalili et. al. proposed six new TFs, S-shaped and V-shaped (Mirjalili and Lewis, 2013). The formula of each function is denoted in Table 1. The value obtained between 0 and 1 is converted to a binary value using Eq. (12).

Transfer functions U-shaped has been defined as $U(x) = \alpha|x^{\beta}|$ (Mirjalili *et al.*, 2020). Where α, β are the control parameters. U1, U2, U3, U4 transfer functions which used in our study is shown in Table 2. Obtained value with the U-shaped TF is converted into binary space using Eq. (12).

In order to effectively perform the binary optimization problems, Taper-shaped TF was introduced (He *et al.*, 2022). Formulas of T1, T2, T3, T4 TFs are given in Table 2. There are upper bounds of the search space $[-A, A]$. The calculated real value with the T-shaped TF is converted into binary space using Eq. (13).

$$Bin_{val} = \begin{cases} 1, if\ 0.5 \ge TF(x) \\ 0, else \end{cases} \tag{13}$$

Z-shaped probability transfer function is proposed by Guo et al. (Guo *et al.*, 2020). The formula of each Z-shaped function is given in Table 3. A real number obtained between 0 and 1 using the Z-shaped TF is converted to the binary value using Eq. (12).

In addition to the TFs mentioned above, Other-shaped transfer functions also appear in the literature. O1 TF is proposed by Pampará et. al. (Pampará and Engelbrecht, 2011). O2 TF is introduced by Costa et al. (Costa *et al.*, 2014). O3 TF is linear normalization function (Wang *et al.*, 2008), O4 TF is taken as unit function (Zhu *et al.*, 2017). The value calculated with the O1, O4, O2, O3 TFs is converted to binary value with Eq. (14), Eq. (15), Eq. (12), respectively. The formula of each Other-shaped function family is given in Table 3.

$$Bin_{val} = \begin{cases} 1, if\ 0 \leq TF(x) \\ 0, else \end{cases} \quad (14)$$

$$Bin_{val} = TF(x) \quad (15)$$

The last TF we use is Hyperbolic tangent sigmoid (TanSig) TF is given Table 4 (Yonaba, Anctil and Fortin, 2010). The real value is converted to binary according to Eq. (16).

$$Bin_{val} = \begin{cases} 1, if\ 0.6 < TF(x) \\ 0, else \end{cases} \quad (16)$$

Table 1. S-Shaped and V-Shaped TFs

Name	Formulation of TF	Equation
S- Shaped	S1: $TF(x) = \frac{1}{(1+e^{-2x})}$	$x_i^d(t+1) = \begin{cases} 1, & if\ r \geq sigm\left(v_i^d(t)\right) \\ 0, & else \end{cases}$ (12)
	S2: $TF(x) = \frac{1}{(1+e^{-x})}$	
	S3: $TF(x) = \frac{1}{(1+e^{-x/2})}$	
	S4: $TF(x) = \frac{1}{(1+e^{-x/3})}$	
V- Shaped	V1: $TF(x) = \left\|erf\left(\frac{\sqrt{\pi}}{2}x\right)\right\|$	
	V2: $TF(x) = \|tanh(x)\|$	
	V3: $TF(x) = \left\|\frac{x}{\sqrt{1+x^2}}\right\|$	
	V4: $TF(x) = \left\|\frac{2}{\pi}arctan\left(\frac{\pi}{2}x\right)\right\|$	

Table 2. U- Shaped and Taper-Shaped TFs

Name	Formulation of TF	Equation
U- Shaped	U1: $TF(x) = \|x^{1.5}\|,\ \alpha = 1,\ \beta = 1.5$	$x_i^d(t+1) = \begin{cases} 1, & if\ r \geq sigm\left(v_i^d(t)\right) \\ 0, & else \end{cases}$ (12)
	U2: $TF(x) = \|x^2\|,\ \alpha = 1,\ \beta = 2$	
	U3: $TF(x) = \|x^3\|,\ \alpha = 1,\ \beta = 3$	
	U4: $TF(x) = \|x^4\|,\ \alpha = 1,\ \beta = 4$	
Taper- Shaped	T1: $TF(x) = \frac{\sqrt{\|x\|}}{\sqrt{\|A\|}}$	$Bin_{val} = \begin{cases} 1, if\ 0.5 \geq TF(x) \\ 0, else \end{cases}$ (13)
	T2: $TF(x) = \frac{\|x\|}{\|A\|}$	
	T3: $TF(x) = \frac{\sqrt[3]{\|x\|}}{\sqrt[3]{\|A\|}}$	
	T4: $TF(x) = \frac{\sqrt[4]{\|x\|}}{\sqrt[4]{\|A\|}}$	

Table 3. Z- Shaped and Other-Shaped TFs

Name	Formulation of TF	Equation
Z- Shaped	Z1: $TF(x) = \sqrt{1-2^x}$ Z2: $TF(x) = \sqrt{1-5^x}$ Z3: $TF(x) = \sqrt{1-8^x}$ Z4: $TF(x) = \sqrt{1-20^x}$	$x_i^d(t+1) = \begin{cases} 1, & if\ r \geq sigm\left(v_i^d(t)\right) \\ 0, & else \end{cases}$ (12)
Other- Shaped	O1: $TF(x) = sin\big(2\pi(x-a) * b * cos(2\pi(x-a) * c)\big) + d$ $(a = d = 0, \quad b = c = 1)$	$Bin_{val} = \begin{cases} 1, if\ 0 \leq TF(x) \\ 0, else \end{cases}$ (14)
	O2: $TF(x) = [\![x\ mod\ 2]\!]$	$Bin_{val} = TF(x)$ (15)
	O3: $TF(x) = \frac{(x-A_{min})}{A_{max}-A_{min}}, \quad (A_{min} \leq x \leq A_{max})$	$x_i^d(t+1) = \begin{cases} 1, & if\ r \geq sigm\left(v_i^d(t)\right) \\ 0, & else \end{cases}$ (12)
	O4: $TF(x) = x$	$Bin_{val} = \begin{cases} 1, if\ 0 \leq TF(x) \\ 0, else \end{cases}$ (14)

Table 4. Hyperbolic Tangent Sigmoid TF

Name	Formulation of TF	Equation
Hyperbolic tangent sigmoid	$TF(x) = \frac{2}{1+e^{-2x}} - 1$	$Bin_{val} = \begin{cases} 1, if\ 0.6 \leq TF(x) \\ 0, else \end{cases}$ (16)

3. Binary HBA with Transfer Functions

The HBA algorithm performs for continuous problems due to its structure. It is clear that the candidate solutions formed by Eq. (3) consist of continuous values. The transfer functions mentioned in Table 1, 2, 3 and 4 take a continuous value as input, then normalized to a value between 0 and 1 using the corresponding equation. Pseudocode of Binary HBA is given in Algorithm 1.

Algorithm 1. Pseudocode of Binary HBA

```
Set parameters t_max, N, β, C.
Generate a random real-valued population with Eq. (3).
Convert each candidate solution to binary representation using TF.
Calculate the fitness value of each candidate solution x_i in the binary representation. (i=1, 2, …, N)
Save best solution x_prey and assign fitness to f_prey.
while t ≤ t_max do
    Update α using Eq. (7).
    for i = 1 to N do
        Calculate I_i using Eq. (4).
            if r < 0.5 then
                Update the real valued candidate solution x_new using Eq. (8).
            else
                Update the real valued candidate solution x_new using Eq. (10).
            end if
            Convert new candidate solution to binary representation using TF.
            Compare the existing candidate solution with the x_new by fitness value.
            if fitness(x_new) ≤ fitness(x_i) then
                x_i = x_new and fitness(x_i) = fitness(x_new).
            end if
            if fitness(x_new) ≤ f_prey then
                x_prey = x_new and f_prey = fitness(x_new .)
            end if
        end for
end while Stop criteria satisfied.
Return x_prey
```

As can be seen from Algorithm 1, in Binary HBA, before calculate fitness, continuous values are transformed to binary with the help of TF. The transformation of a candidate solution consisting of 5 dimensional real values $x = [-5.48, -3.30, 4.46, 9.71, -6.35]$ into binary representation [1, 1, 0, 0, 1] with the help of the S2 TF is given in Table 5.

Table 5. Conversion of continuous value to binary with S2 TF

i	x_i	$S2(x_i)$	r	Binary
1	-5.48	0.0041	0.1072	1
2	-3.30	0.0356	0.0736	1
3	4.46	0.9885	0.0917	0
4	9.71	0.9999	0.7845	0
5	-6.35	0.0017	0.3039	1

4. Experimental Results

In this section, the HBA algorithm is adapted for solving 0-1 KPs. The HBA is an algorithm that performs continuous search space due to its structure. 0-1 KPs, on the other hand, have a binary structure. For this reason, first, N real-valued candidate solutions, each of which is D-dimensional, are created. After each candidate honey badger position is converted to binary with the help of transfer functions, fitness value is evaluated. A total of 25 transfer functions as V-Shaped, S-Shaped, U-Shaped, T-Shaped, Tangent Sigmoid, O-Shaped, Z-Shaped TFs are used to adapt the binary version of the HBA. Each transfer function is tested by computational experiments over 25 instances of 0-1 KP and compared results.

Our experiment was carried on the problems in the benchmark dataset, which can be taken from (https://pages.mtu.edu/~kreher/cages/Data.html) in Table 6.

Table 6. Benchmark datasets

Problem	Capacity	Dimension	Optimal
KP8a	1.863.633	8	3.924.400
KP8b	1.822.718	8	3.813.669
KP8c	1.609.419	8	3.347.452
KP8d	2.112.292	8	4.187.707
KP8e	2.493.250	8	4.955.555
KP12a	2.805.213	12	5.688.887
KP12b	3.259.036	12	6.473.019
KP12c	2.489.815	12	5.170.626
KP12d	3.453.702	12	6.941.564
KP12e	2.520.392	12	5.337.472
KP16a	3.780.355	16	7.850.983
KP16b	4.426.945	16	9.352.998
KP16c	4.323.280	16	9.151.147
KP16d	4.550.938	16	9.348.889
KP16e	3.760.429	16	7.769.117
KP20a	5.169.647	20	10.727.049
KP20b	4.681.373	20	9.818.261
KP20c	5.063.791	20	10.714.023
KP20d	4.286.641	20	8.929.156
KP20e	4.476.000	20	9.357.969
KP24a	6.404.180	24	13.549.094
KP24b	5.971.071	24	12.233.713
KP24c	5.870.470	24	12.448.780
KP24d	5.762.284	24	11.815.315
KP24e	6.654.569	24	13.940.099

For a fair comparison, the number of *maxFes*, population size and runtime illustrate in Table 7. GAP values are calculated using Eq. (17).

$$GAP = \frac{optimal - mean}{optimal} \tag{17}$$

Table 7. The parameter values

Parameters	Value
Maximum Fes	1000 (For KP8a to KP12e)
	5000 (For KP16a to KP24e)
Population size	40
Runtime	50

In order to show performance of the proposed method, a total of 10 problems, Kp8a-Kp8e and Kp12a-Kp12e taken from the benchmark dataset, were performed for 50 runtimes and 1000 Fes number. Also, 15 problems, including Kp16a-Kp16e, Kp20a-Kp20e, Kp24a-Kp24e, run with 50 runtimes and 5000 Fes number. In all of the problems, the search space in the HBA algorithm was adapted to binary space with the help of S, V, U, T, Hyperbolic Tangent Sigmoid, Z and Other-shaped transfer functions. The gap value between the approximate solutions obtained for each transfer function and the optimal solutions was given in Table 8 and Table 9. Table 8 and Table 9 show that optimal solutions were obtained by V, U shaped transfer functions and T1, T3, T4, O1, O2 shaped transfer functions for problems with a problem size of 8. In addition, V, U1, U4, T4, O1, O2 shaped transfer functions reached the optimum value for all runs in 4 problems with problem size 12. In the dataset with a problem size of 16, the optimal value was reached in 3 of the five problems with the help of V, U, T, O2 shaped transfer functions. Although it is seen that it is difficult for the results obtained to reach the optimum value when the problem size is 20 and 24, optimum values were obtained in 2 or 1 of the five problems depending on the transfer functions used. It has been seen that the lowest gap values are in the solutions obtained with the O1 and O2 transfer functions. When we consider Table 8 and Table 9 in general, it can be said that O1 and O2 transfer functions are in the front according to the efficiency of the transfer functions for the 25 problems. In the continuation of this sorting, it has been seen that U1-shaped transfer function gives efficient results.

In this study, the 25 KPs run 50 times to test each transfer function. The optimum number of values obtained for each transfer function due to running 1250 times is given as hit value in Table 10. According to the table, it was seen that the optimal value was reached with the O1 transfer function in 1017 and O2 in 1009 of 1250, respectively. After these functions, U and T transfer functions get the maximum optimum value. As a result, it was observed that the HBA algorithm performed successful results for the O1 and O2 transfer functions.

Table 8. Gap values for each TF (S, V, U) and KP problem with HBA algorithm

Problem	S- Shaped				V- Shaped				U- Shaped			
Name	S1	S2	S3	S4	V1	V2	V3	V4	U1	U2	U3	U4
KP8a	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP8b	0.163	0.228	0.081	0.114	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP8c	0.013	0.020	0.020	0.007	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP8d	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP8e	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP12a	0.070	0.063	0.105	0.084	0.093	0.080	0.075	0.057	0.034	0.032	0.039	0.025
KP12b	0.110	0.118	0.142	0.118	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.008	0.024	**0.000**
KP12c	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP12d	0.018	0.049	0.018	0.027	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP12e	0.036	0.054	0.054	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP16a	0.177	0.156	0.169	0.164	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP16b	0.117	0.105	0.054	0.104	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP16c	**0.000**	0.005	**0.000**	0.017	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP16d	0.147	0.146	0.149	0.121	0.523	0.523	0.523	0.523	0.073	0.063	0.076	0.078
KP16e	0.198	0.193	0.191	0.190	0.286	0.282	0.286	0.288	0.182	0.157	0.199	0.144
KP20a	**0.000**	**0.000**	0.008	0.006	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP20b	0.103	0.056	0.074	0.048	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP20c	0.056	0.058	0.051	0.052	0.044	0.044	0.044	0.044	0.037	0.037	0.033	0.037
KP20d	0.136	0.144	0.140	**0.099**	0.154	0.154	0.154	0.154	0.151	0.154	0.148	0.154
KP20e	0.082	0.065	0.067	0.058	0.124	0.099	0.099	0.033	0.010	0.009	0.007	0.016
KP24a	0.227	0.232	0.181	0.268	0.326	0.332	0.334	0.321	0.259	0.232	0.280	0.239
KP24b	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP24c	0.017	0.021	0.031	0.017	0.052	0.052	0.044	0.043	0.006	0.004	0.001	0.002
KP24d	0.103	0.092	0.099	0.132	0.045	0.045	0.045	0.045	0.043	0.045	0.045	0.044
KP24e	0.088	0.065	0.057	0.053	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.001	0.001	0.001
Friedman Rank	16.1	16.56	16.44	15.46	13.12	12.76	12.72	11.92	8.68	9.06	9.46	8.84
Rank	18	20	19	17	14	13	12	11	3	5	8	4

Table 9. Gap values for each TF (T, Hyp.Tan, O, Z) and KP problem with HBA algorithm

Problem	Taper- Shaped				Hyp.Tan.	Other- Shaped				Z- Shaped			
Name	T1	T2	T3	T4	TanSig	O1	O2	O3	O4	Z1	Z2	Z3	Z4
KP8a	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP8b	**0.000**	0.065	**0.000**	**0.000**	0.130	**0.000**	**0.000**	0.601	0.195	0.293	0.098	0.130	0.195
KP8c	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.623	0.007	**0.000**	0.033	0.013	0.007
KP8d	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP8e	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP12a	0.043	0.053	0.043	0.052	0.079	0.019	**0.018**	0.643	0.088	0.083	0.082	0.077	0.098
KP12b	0.016	0.071	**0.000**	**0.000**	0.087	**0.000**	**0.000**	0.397	0.142	0.118	0.079	0.118	0.118
KP12c	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP12d	**0.000**	**0.000**	0.009	**0.000**	0.018	**0.000**	**0.000**	0.519	0.040	**0.000**	0.019	0.027	0.066
KP12e	0.018	0.018	0.018	**0.000**	0.144	**0.000**	**0.000**	0.664	0.036	0.036	0.072	0.000	0.054
KP16a	**0.000**	**0.000**	**0.000**	**0.000**	0.168	0.010	**0.000**	0.356	0.195	0.203	0.196	0.212	0.152
KP16b	**0.000**	**0.000**	**0.000**	**0.000**	0.054	**0.000**	**0.000**	0.791	0.175	0.157	0.132	0.059	0.078
KP16c	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.021	0.017	0.005	**0.000**	0.015
KP16d	0.068	0.104	0.063	0.167	0.164	0.068	**0.057**	0.130	0.173	0.132	0.110	0.138	0.188
KP16e	0.181	0.156	0.207	0.226	0.205	0.068	**0.054**	0.434	0.186	0.188	0.228	0.181	0.214
KP20a	**0.000**	**0.000**	**0.000**	**0.000**	0.021	**0.000**	**0.000**	0.275	0.016	0.021	0.036	0.012	0.011
KP20b	**0.000**	**0.000**	**0.000**	**0.000**	0.073	**0.000**	**0.000**	0.023	0.102	0.061	0.069	0.078	0.065
KP20c	0.033	0.029	0.037	0.039	0.073	**0.015**	0.024	0.114	0.045	0.035	0.047	0.043	0.055
KP20d	0.145	0.114	0.151	0.154	0.129	0.133	0.136	0.154	0.129	0.114	0.129	0.126	0.138
KP20e	0.008	0.035	0.021	0.039	0.065	**0.013**	0.048	0.268	0.041	0.053	0.048	0.049	0.063

KP24a	0.248	0.196	0.265	0.289	0.278	**0.101**	0.154	0.344	0.261	0.174	0.209	0.216	0.231
KP24b	**0.000**	**0.000**	**0.000**	**0.000**	0.016	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
KP24c	0.003	0.004	0.006	0.019	0.036	0.007	0.032	**0.000**	0.053	0.030	0.034	0.045	0.015
KP24d	0.044	0.045	0.045	0.045	0.127	**0.042**	0.040	0.046	0.158	0.072	0.087	0.089	0.087
KP24e	0.001	0.008	0.001	**0.000**	0.071	0.034	0.001	0.122	0.055	0.080	0.059	0.085	0.056
Friedman Rank	9.06	9.38	10.26	11.06	17.1	7.82	7.88	19.56	17.94	14.96	16.58	15.24	17.04
Rank	6	7	9	10	23	1	2	25	24	15	21	16	22

Table 10. Hit values for each TF with HBA algorithm

TF	Hit value	TF	Hit value	TF	Hit value
S1	810	U1	946	TanSig	792
S2	816	U2	950	O1	1017
S3	816	U3	935	O2	1009
S4	821	U4	950	O3	511
V1	821	T1	942	O4	794
V2	827	T2	925	Z1	789
V3	833	T3	933	Z2	802
V4	845	T4	893	Z3	809
				Z4	789

The binary version of HBA for O1 TF is compared with BPSO, MBPSO (Modified Binary Particle Swarm Optimization) and NGHS (Novel Global Harmony Search) algorithms to evaluate its performance and accuracy. All methods were performed with the same parameters as 50 runs, 1000 Fes number for Kp8a to Kp12e and 5000 Fes number for Kp16a to Kp24e. Experimental results of algorithms were directly taken from (Zhou, Chen and Zhou, 2016), (Hakli, 2020). The gap values of 50 runs for the algorithms and the proposed algorithm are presented in Table 11. The binary version of HBA with O1 TF found the optimum value or the closest results for 22 of 25 problems. Thus, it has been seen that HBA with O1 TF offers more effective solutions than BPSO, MBPSO, and NGHS algorithms for selected 0-1 KP problems.

Table 11. Experimental results of proposed method and binary variants of the different algorithms.

Problem Name	HBA-O1-Shaped	BPSO	MBPSO	NGHS
KP8a	**0.000**	0.065	**0.000**	**0.000**
KP8b	**0.000**	0.151	**0.000**	**0.000**
KP8c	**0.000**	0.563	**0.000**	**0.000**
KP8d	**0.000**	0.039	**0.000**	**0.000**
KP8e	**0.000**	0.460	0.020	**0.000**
KP12a	0.019	0.091	**0.006**	0.020
KP12b	**0.000**	0.308	0.084	0.187
KP12c	**0.000**	0.071	0.003	0.107
KP12d	**0.000**	0.037	0.004	0.006
KP12e	**0.000**	0.386	**0.000**	1.190
KP16a	**0.010**	0.205	0.101	0.711
KP16b	**0.000**	0.199	0.028	1.068
KP16c	**0.000**	0.353	0.077	1.041
KP16d	**0.068**	0.291	0.117	0.406
KP16e	0.068	0.136	**0.064**	0.390
KP20a	**0.000**	0.184	0.063	1.256
KP20b	**0.000**	0.275	0.130	0.909
KP20c	**0.015**	0.099	0.029	1.203
KP20d	0.133	0.213	**0.061**	0.782
KP20e	**0.013**	0.090	0.022	0.356
KP24a	**0.101**	0.285	0.126	0.296
KP24b	**0.000**	0.232	0.084	0.595
KP24c	**0.007**	0.168	0.044	0.195
KP24d	**0.042**	0.197	0.098	0.669
KP24e	**0.034**	0.124	0.054	0.805

5. Conclusions

This study proposed the binary version of HBA algorithm with TFs. The binary variants performed with the help of 25 transfer functions were applied to benchmark datasets for 0-1 KP problem. The results for 25 binary variants were compared to examine the efficiency of each transfer function. The O1 and O2 TFs showed the best successful performances among the TFs. Also, HBA algorithm with O1 TF was compared with three different binary variants, and the results show that binary HBA is the first in the ranking. For future work, the validity of the proposed approach can be enlarged by applying it to different 0-1 KPs. It can be impressive work to adapt the HBA algorithm to binary space without TFs directly.

References

Abdel-Basset, M. et al., 2021. New binary marine predators optimization algorithms for 0-1 knapsack problems. Computers and Industrial Engineering, 151.

Abdel-Basset, M., El-Shahat, D. and El-Henawy, I., 2019. Solving 0-1 knapsack problem by binary flower pollination algorithm. Neural Computing and Applications, 31(9), pp. 5477-5495.

Abdel-Basset, M., El-Shahat, D. and Sangaiah, A.K., 2019. A modified nature inspired meta-heuristic whale optimization algorithm for solving 0-1 knapsack problem. International Journal of Machine Learning and Cybernetics, 10(3), pp. 495-514.

Abdel-Basset, M., Mohamed, R. and Mirjalili, S., 2021. A binary equilibrium optimization algorithm for 0-1 knapsack problems. Computers and Industrial Engineering, 151.

Abdollahzadeh, B. et al., 2021. An enhanced binary slime mould algorithm for solving the 0-1 knapsack problem. Engineering with Computers.

Ali, I.M., Essam, D. and Kasmarik, K., 2021. Novel binary differential evolution algorithm for knapsack problems. Information Sciences, 542, pp. 177-194.

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

Costa, M.F.P. et al., 2014. Heuristic-based firefly algorithm for bound constrained nonlinear binary optimization. Advances in Operations Research, 2014.

Çerçevik, A.E. and Avşar, Ö., 2020. Optimization of linear seismic isolation parameters via crow search algorithm. Pamukkale University Journal of Engineering Sciences, 26(3), pp. 440-447.

Dantzig, G.B., 1957. Discrete-Variable Extremum Problems, Source: Operations Research.

Deng, W., Xu, J. and Zhao, H., 2019. An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. IEEE Access, 7, pp. 20281-20292.

Ezugwu, A.E. et al., 2019. A comparative study of meta-heuristic optimization algorithms for 0-1 knapsack problem: some initial results. IEEE Access, 7, pp. 43979-44001.

Feng, Y. et al., 2017. Solving 0-1 knapsack problem by a novel binary monarch butterfly optimization. Neural Computing and Applications, 28(7), pp. 1619-1634.

Gherboudj, A., Layeb, A. and Chikhi, S., 2012. Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm. International Journal of Bio-Inspired Computation, 4(4), pp. 229-236.

Guo, S.S. et al., 2020. Z-shaped transfer functions for binary particle swarm optimization algorithm. Computational Intelligence and Neuroscience, 2020.

Hakli, H., 2019. A new approach for wind turbine placement problem using modified differential evolution algorithm. Turkish Journal of Electrical Engineering and Computer Sciences, 27(6), pp. 4659-4672.

Hakli, H., 2020. BinEHO: a new binary variant based on elephant herding optimization algorithm. Neural Computing and Applications, 32(22), pp. 16971-16991.

Halat, M. and Ozkan, O., 2021. The optimization of UAV routing problem with a genetic algorithm to observe the damages of possible Istanbul earthquake. Pamukkale University Journal of Engineering Sciences, 27(2), pp. 187-198.

Harifi, S., 2022. A binary ancient-inspired Giza pyramids construction metaheuristic algorithm for solving 0-1 knapsack problem. Application of Soft Computing.

Hashim, F.A. et al., 2022. Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. Mathematics and Computers in Simulation, 192, pp. 84-110.

He, Y. et al., 2022. Novel binary differential evolution algorithm based on taper-shaped transfer functions for binary optimization problems. Swarm and Evolutionary Computation, 69.

Ismail M. Ali, D.E. and K.K., 2018. An efficient differential evolution algorithm for solving 0-1 knapsack problems. 2018 IEEE Congress on Evolutionary Computation (CEC): 2018 proceedings.

J. Kennedy, R.C.E., 1997. Discrete binary version of the particle swarm algorithm. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 5 (1997) 4104-4108. IEEE.

Kaya, S. et al., 2020. The effects of initial populations in the solution of flow shop scheduling problems by hybrid firefly and particle swarm optimization algorithms. Pamukkale University Journal of Engineering Sciences, 26(1), pp. 140-149.

Kulkarni, A.J. and Shabir, H., 2016. Solving 0-1 knapsack problem using cohort intelligence algorithm. International Journal of Machine Learning and Cybernetics, 7(3), pp. 427-441.

Liu, K. et al., 2022. A hybrid harmony search algorithm with distribution estimation for solving the 0-1 knapsack problem. Mathematical Problems in Engineering.

Mirjalili, S. and Lewis, A., 2013. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm and Evolutionary Computation, 9, pp. 1-14.

Mirjalili, Seyedehzahra et al., 2020. A novel U-shaped transfer function for binary particle swarm optimisation. Advances in Intelligent Systems and Computing. Springer, pp. 241-259.

Nguyen, P.H., Wang, D. and Truong, T.K., 2017. A novel binary social spider algorithm for 0-1 knapsack problem. International Journal of Innovative Computing.

Pampará, G. and Engelbrecht, A.P., 2011. Binary artificial bee colony optimization. IEEE SSCI 2011- Symposium Series on Computational Intelligence- SIS 2011: 2011 IEEE Symposium on Swarm Intelligence, pp. 170-177.

Pavithr, R.S. and Gursaran, 2016. Quantum inspired social evolution (QSE) algorithm for 0-1

knapsack problem. Swarm and Evolutionary Computation, 29, pp. 33-46.
Rizk-Allah, R.M. and Hassanien, A.E., 2018. New binary bat algorithm for solving 0-1 knapsack problem. Complex & Intelligent Systems, 4(1), pp. 31-53.
Rooderkerk, R.P. and van Heerde, H.J., 2016. Robust optimization of the 0-1 knapsack problem: Balancing risk and return in assortment optimization. European Journal of Operational Research, 250(3), pp. 842-854.
Shu, Z. et al., 2022. A modified hybrid rice optimization algorithm for solving 0-1 knapsack problem. Applied Intelligence, 52(5), pp. 5751-5769.
Wang, L. et al., 2008. A novel probability binary particle swarm optimization algorithm and its application.
Wang, L., Shi, R. and Dong, J., 2021. A hybridization of dragonfly algorithm optimization and angle modulation mechanism for 0-1 knapsack problems. Entropy, 23(5).
Yassien, E. et al., 2017. Grey wolf optimization applied to the 0/1 knapsack problem. International Journal of Computer Applications, 169(5), pp. 11-15.
Yonaba, H., Anctil, F. and Fortin, V., 2010. Comparing sigmoid transfer functions for neural network multistep ahead streamflow forecasting. Journal of Hydrologic Engineering, 15(4), pp. 275-283.
Zhou, Y., Chen, X. and Zhou, G., 2016. An improved monkey algorithm for a 0-1 knapsack problem. Applied Soft Computing Journal, 38, pp. 817-830.
Zhou, Y., Li, L. and Ma, M., 2016. A complex-valued encoding bat algorithm for solving 0-1 knapsack problem. Neural Processing Letters, 44(2), pp. 407-430.
Zhu, H. et al., 2017. Discrete differential evolutions for the discounted {0-1} knapsack problem. Chinese Journal of Computers and so on.