The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM)
ISSN: 2602-3199

The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 2026

Volume 40, Pages 425-438

ICBAST 2026: International Conference on Basic Sciences and Technology

A Binary Hiking Optimization Algorithm for 0/1 Knapsack Problem

Aybuke Babadag
Konya Technical University

Abstract: The 0/1 knapsack problem is one of the prominent NP-hard binary optimization problems. Solving such problems using exact methods can be computationally expensive. Therefore, metaheuristic methods which offer optimal or near optimal solutions in a reasonable time, are commonly utilized for such tasks. The Hiking Optimization Algorithm (HOA) is a recently proposed metaheuristic optimization algorithm that is inspired by the similarity between the hikers' navigation of steep terrain and the search process within the optimization problem's search space. However, the basic algorithm is initially proposed for continuous optimization problems. To apply the algorithm to binary optimization tasks, certain adaptations are required. One of the commonly used binarization techniques involves using a transfer function to convert continuous decision variables into binary counterparts. In this study, HOA is binarized using four different variants for each of the following transfer function types: S-shaped, U-shaped, Z-shaped, and T-shaped. Furthermore, a repair function is adopted to tackle the infeasible solutions produced by the algorithm. The performance of the proposed binary HOA (binHOA) algorithm is assessed on two standard benchmark datasets. The results are compared with the binary versions of the well-known optimization algorithms. The experimental results indicate that the proposed binary version of HOA is a competitive alternative for the 0/1 knapsack problem.

Keywords: Hiking optimization algorithm, 0/1 knapsack problem, Transfer functions

Introduction

The knapsack problem is a well-known combinatorial optimization problem that aims to find the combination of items that maximizes total profit while ensuring that the sum of their weights does not exceed the knapsack's capacity (Zhou et al., 2023). The knapsack problem has a variety of real-world applications, including resource allocation (Reniers & Sörensen, 2013), container loading (Pisinger, 2002), and real estate property maintenance (Taillandier et al., 2017). This problem has several variants, such as the multiple-choice knapsack, multi-dimensional knapsack, fractional knapsack, and 0/1 knapsack, that cover numerous application areas (Mahafzah et al., 2025). This study focuses on the 0/1 knapsack problem. In the 0/1 knapsack problem, if an item is selected, the value of the decision variable associated with it becomes 1; otherwise, it becomes 0, indicating that the item is not selected. The knapsack problem belongs to the NP-hard class. Therefore, finding a solution in a reasonable time using exact methods may require excessive computation time for some problem instances, especially high dimensional ones (Pisinger, 2005). Metaheuristic algorithms can be used as an alternative due to their stochastic nature, which makes it relatively efficient to find optimal or near-optimal solutions in a reasonable time.

In recent years, numerous metaheuristic algorithms have been proposed to address the 0/1 knapsack problem. In the study by Abdel-Basset et al. (2021), the marine predators algorithm (MPA) was applied to the 0/1 knapsack problem, and the basic algorithm was binarized using S-shaped and V-shaped transfer functions. Similarly, in the study by Sun et al. (2021), the binary particle swarm optimization (PSO) algorithm was used to solve the 0/1 knapsack problem. Four different variants of Z-shaped transfer functions were employed to convert continuous decision variables into binary ones. To ensure that the generated solutions complied with the constraints, a penalty function was utilized. Shu et al. (2022) proposed a hybrid model combining the binary ant colony

optimization (ACO) algorithm and a modified version of the hybrid rice optimization (HRO) algorithm to solve the large-scale 0/1 knapsack problem. In addition, a two-phase repair strategy was used to handle infeasible solutions. In the study by Yildizdan and Baş (2023), the artificial jellyfish search (AJS) algorithm was binarized to solve the 0/1 knapsack problem, where the continuous search space was mapped to a binary version using S-shaped and V-shaped transfer functions. Büyüköz and Haklı (2023) proposed a binary version of the honey badger algorithm (HBA) for solving the 0/1 knapsack problem. In the binarization process, V-shaped, S-shaped, U-shaped, T-shaped, Tangent Sigmoid, O-shaped, and Z-shaped transfer functions were used. Similarly, Ervural and Hakli (2023) proposed a binary version of the reptile search algorithm (RSA) for the 0/1 knapsack problem. Various transfer functions, including V-shaped, S-shaped, U-shaped, T-shaped, and Z-shaped, were utilized in the binarization process. In addition, a stochastic repair and improvement method was introduced to handle infeasible solutions generated by the algorithm. Limane et al. (2024) introduced a binary version of the electric eel foraging optimization (EEFO) algorithm for solving the 0/1 knapsack problem. S-shaped and V-shaped transfer functions were employed to transfer the continuous search space into binary search space. To further improve the performance of the proposed algorithm, an enhanced repair and improvement (RI) method was introduced to tackle infeasible solutions. Erdoğan et al. (2025) proposed a binary version of the grey wolf optimizer (GWO), namely the binary dynamic grey wolf optimizer (BDGWO), to solve binary optimization problems. The effectiveness of the BDGWO algorithm was assessed using the 0/1 knapsack problem. Ihsan and Sag (2025) introduced a binary version of the puma optimizer (PO) algorithm, which was binarized using S-shaped and V-shaped transfer functions. The performance of the proposed algorithm was evaluated on the 0/1 knapsack problem.

As seen from the studies in the literature, various metaheuristic optimization algorithms have been applied to the 0/1 knapsack problem. One of the recently proposed metaheuristic optimization algorithms is the hiking optimization algorithm (HOA), which is inspired by the resemblance between a hiker's traversal of the mountainous terrain and the exploration of an optimization problem's search space. However, it is observed from the literature that there is no notable binary variant of the HOA introduced to solve the knapsack problem. To address this gap, the recently introduced hiking optimization algorithm is binarized in this study to solve the 0/1 knapsack problem. It is also observed that transfer functions for converting the search space from continuous to binary are commonly used in the literature. In this study, four different types of transfer functions, namely S-shaped (Mirjalili & Lewis, 2013), U-shaped (Mirjalili et al., 2020), Z-shaped (Guo et al., 2020), and T-shaped (He et al., 2022), are used to binarize the HOA. The performance of the proposed binary version of the HOA, called binary HOA (binHOA), is evaluated on two well-known benchmark datasets. The performance of the binHOA is compared with six well-known metaheuristic algorithms.

Method

The 0/1 Knapsack Problem

In the knapsack problem it is assumed that there is a knapsack with a capacity of W and n items with weight of w and a profit of p. Let x_i be a vector holds the information whether the item i is included in the knapsack. The objective function of the knapsack problem is as follows (Sakib et al., 2025):

$$maximize \sum_{i=1}^{N} p_i x_i$$

Subject to constraints:

$$\sum_{i=1}^{N} w_i x_i \leq W$$

$$x_i \in \{0,1\}$$

where x_i=1 means *i-th* item is included and x_i=0 means *i-th* item is not included. The objective is to determine an x vector that maximizes the total profit and does not exceed the capacity of the knapsack.

Hiking Optimization Algorithm

HOA is a human-inspired metaheuristic algorithm based on the hiking activity (Oladejo et al., 2024). The algorithm is mathematically modeled using Tobler's hiking function, an exponential function that determines a hiker's walking speed, as a function of terrain slope. The Tobler's hiking function given as follows:

$$W_{i,t} = 6e^{-3.5|S_{i,t}+0.05|}$$

where $W_{i,t}$ represents the velocity of the i-th hiker at the t-th iteration. $S_{i,t}$ represents the slope of the terrain and calculated as follows:

$$S_{i,t} = \frac{\mathrm{d}h}{\mathrm{d}x} = tan\theta_{i,t}$$

where dh represents the difference in elevation, dx represents the distance traversed by the hiker, and the θ denotes the inclination angle of the terrain, ranging from 0º to 50º. The current velocity of the i-th hiker is calculated as follows:

$$W_{i,t} = W_{i,t-1} + \gamma_{i,t}\left(\beta_{best} - \alpha_{i,t}\beta_{i,t}\right)$$

$\gamma_{i,t}$ is a uniformly distributed number in the range [0,1], $W_{i,t}$ and $W_{i,t-1}$ represents the current and initial velocity of the i-th hiker respectively. β_{best} is the position of the lead hiker and $\alpha_{i,t}$ is the sweep factor (SF) of the i-th hiker ranging from 1 to 3. SF prevents hikers from deviating excessively from the lead hiker, ensuring both navigational guidance and signal reception are maintained. The position of the i-th hiker is updated as follows:

$$\beta_{i,t+1} = \beta_{i,t} + W_{i,t}$$

The pseudocode of the HOA is given in Algorithm 1.

Algorithm 1 Pseudocode of the HOA

input: *Ub* (upper bound), *Lb* (lower bound), *T* (maximum iteration number), *I* (number of hikers), *d* (dimension)
F← initialize fitness vector
F_{best} ← initialize best fitness vector
β ← initialize hikers' position randomly
for i ← 1 to I
    F_i ← Calculate the fitness of the i-th hiker
end for
$F_{best,1}$ ← initial best fitness of hikers
for t ← 1 to T
    $F_{best,t}$ ← best fitness value
    β_{best} ← position of the best solution
    for i ←1 to I
        $\beta_{i,t}$ ← initial position of the i-th hiker
        $\theta_{i,t}$ ← elevation angle
        $S_{i,t}$ ← calculate the slope
        $W_{i,t-1}$ ← calculate the initial velocity of the i-th hiker
        $W_{i,t}$ ← calculate the actual velocity of the i-th hiker
        $\beta_{i,t+1}$ ← update the position of the i-th hiker
        Apply boundary constraints to $\beta_{i,t}$ within [*Lb, Ub*]
        if $F_n \leq F(n)$
            $\gamma(n,:) \leftarrow \beta_{i,t+1}$
            $F(n) \leftarrow F_n$
        end if
    end for
        F_{sol} (i+1) ←min(F)
end for
        F_{opt_sol} ←argmin(F_{sol})
return F_{opt_sol}

Proposed Binary Hiking Optimization Algorithm

Hiking optimization algorithm is initially proposed for solving continuous optimization problems. It requires certain modifications to adapt the HOA to solve binary optimization problems such as the 0/1 knapsack. There are various techniques that transform real-valued decision variables into binary counterparts. Transfer functions are one of the common methods for this purpose. In this study, four different types of transfer functions are used. Four variants of each transfer function, each with different parameters, were evaluated and the best-performing one was determined. The formulations of the transfer functions are given in Table 1.

Table 1. The formulations of the transfer functions: S-shaped, U-shaped, Z-shaped, and T-shaped

S-shaped	U-shaped	Z-shaped	T-shaped
$s_1(x) = \frac{1}{1+e^{-2x}}$	$u_1(x) = x^{1.5}$	$z_1(x) = \sqrt{1-2^x}$	$t_1(x) = \frac{\sqrt{x}}{\sqrt{A}}$
$s_2(x) = \frac{1}{1+e^{-x}}$	$u_2(x) = x^2$	$z_2(x) = \sqrt{1-5^x}$	$t_2(x) = \frac{\lvert x\rvert}{\lvert A\rvert}$
$s_3(x) = \frac{1}{1+e^{-x/2}}$	$u_3(x) = x^3$	$z_3(x) = \sqrt{1-8^x}$	$t_3(x) = \frac{\sqrt[3]{x}}{\sqrt[3]{A}}$
$s_4(x) = \frac{1}{1+e^{-x/3}}$	$u_4(x) = x^4$	$z_4(x) = \sqrt{1-20^x}$	$t_4(x) = \frac{\sqrt[4]{x}}{\sqrt[4]{A}}$

After each position updates in the basic algorithm, the continuous position vector is converted to a binary vector as follows:

$$X_{bin} = \begin{cases} 1 & if\ TF(x) \le T \\ 0 & else \end{cases}$$

Algorithm 2 Pseudocode of the binHOA

input: *Ub* (upper bound), *Lb* (lower bound), *T* (maximum iteration number), *I* (number of hikers), *d* (dimension)
F← initialize fitness vector
F_{best} ← initialize best fitness vector
β ← initialize continuous position vector randomly
β_{bin} ← initialize binary position vector randomly
for *i* ← 1 to *I*
    F_i ← Calculate the fitness of the *i*-th hiker
end for
$F_{best,1}$ ← initial best fitness of hikers
for *t* ← 1 to *T*
    $F_{best,t}$ ← best fitness value
    β_{best} ← position of the best solution
    for *i* ←1 to *I*
        $\beta_{i,t}$ ← initial position of the *i*-th hiker
        $\theta_{i,t}$ ← elevation angle
        $S_{i,t}$ ← calculate the slope
        $W_{i,t-1}$ ← calculate the initial velocity of the *i*-th hiker
        $W_{i,t}$ ← calculate the actual velocity of the *i*-th hiker
        $\beta_{i,t+1}$ ← update the position of the *i*-th hiker
        Apply boundary constraints to $\beta_{i,t}$ within [*Lb, Ub*]
        β_{bin} ← binarize the continuous position vector
        Apply the repair function
        if $F_n \le F(n)$
            γ(n,:) ← $\beta_{i,t+1}$
            $F(n)$ ← F_n
        end if
    end for
        F_{sol} (*i*+1) ←min(*F*)
end for
        F_{opt_sol} ←argmin(F_{sol})
return F_{opt_sol}

where X_{bin} is a binary position vector for the corresponding continuous position vector x, T is a threshold value, and TF denotes the transfer function. Following Ervural and Hakli (2023), the threshold values are set to a random number in the range [0, 1] for S-, U-, and Z-shaped functions, and to 0.5 for the T-shaped function. To further improve the performance of the proposed binHOA, a stochastic repair function (Ervural & Hakli, 2023) is adopted. Following the binarization of the continuous position vectors, the repair function is applied, thereby ensuring that all solutions in the population are feasible. The pseudocode of the binHOA is given in Algorithm2.

Results and Discussion

Datasets

This study employs two standard benchmark datasets. The small-scale dataset consists of 10 problem instances, with problem dimensions ranging from 4 to 20. The details of the small-scale dataset (Zou et al., 2011) are given in Table 2.

Table 2. The small-scale dataset

Problem	Dimension	Optimum Value
f1	10	295
f2	20	1024
f3	4	35
f4	4	23
f5	15	481.0694
f6	10	50
f7	7	107
f8	23	9767
f9	5	130
f10	20	1025

The other benchmark dataset is a medium-scale dataset (Bansal & Deep, 2012), which consists of 25 instances. The dimensions of the problems in the dataset range from 8 to 24. The details of the dataset are given in Table 3.

Table 3. The medium-scale dataset

Problem	Dimension	Optimum Value	Problem	Dimension	Optimum Value
ks_8a	8	3924400	ks_16d	16	9348889
ks_8b	8	3813669	ks_16e	16	7769117
ks_8c	8	3347452	ks_20a	20	10727049
ks_8d	8	4187707	ks_20b	20	9818261
ks_8e	8	4955555	ks_20c	20	10714023
ks_12a	12	5688887	ks_20d	20	8929156
ks_12b	12	6473019	ks_20e	20	9357969
ks_12c	12	5170626	ks_24a	24	13549094
ks_12d	12	6941564	ks_24b	24	12233713
ks_12e	12	5337472	ks_24c	24	12448780
ks_16a	16	7850983	ks_24d	24	11815315
ks_16b	16	9352998	ks_24e	24	13940099
ks_16c	16	9151147			

Experimental Results

The proposed binHOA is evaluated on two different scale benchmark datasets. The results are compared with the binary versions of six different well-known optimization algorithms namely, particle swarm optimization (PSO) (Kennedy & Eberhart, 1995), differential evolution (DE) (Storn & Price, 1997), artificial bee colony (ABC) (Karaboga, 2005; Karaboga & Basturk, 2007), grey wolf optimizer (GWO) (Mirjalili et al., 2014), whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016), and sparrow search algorithm (SSA) (Xue & Shen, 2020). Due to the stochastic nature of the algorithms, each is independently run 20 times. The maximum number of iterations and the population size are set to 5000 and 20, respectively, as in (Abdel-Basset et al., 2021; Yildizdan & Baş, 2023). To ensure fairness in the experiments, the same parameter settings are applied to all algorithms, as summarized in Table 4.

Table 4. Parameter setting

Population size	20
Maximum number of iterations	5000
Number of runs	20

The performances of the algorithms are evaluated in terms of standard deviation, gap values, mean fitness value, best fitness value, and worst fitness value. The percentage difference between the actual optimum value and the optimum value generated by the algorithm is measured using the Gap metric, which is calculated as follows:

$$Gap = \begin{cases} \dfrac{opt - m}{opt} \times 100, & \text{for maximization} \\ \dfrac{m - opt}{opt} \times 100, & \text{for minimization} \end{cases}$$

where m is the mean fitness value across all runs and opt is the optimal value of the problem instance. The experimental results of the binHOA on small-scale dataset are presented in Tables 5 and 6.

Table 5. The experimental results of the binHOA on small-scale dataset (s1-u4)

TF	metric	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
	mean	295	1024	35	23	481.0694	52	107	9766.6	130	1025
	min	295	1024	35	23	481.0694	52	107	9765	130	1025
s1	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0.8208	0	0
	gap	0	0	0	0	0	0	0	0.0041	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
s2	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
s3	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
s4	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
u1	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
u2	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
u3	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
u4	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0

Table 6. The experimental results of the binHOA on small-scale dataset (z1-t4)

TF	metric	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
z1	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
z2	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
z3	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
z4	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
t1	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
t2	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0
	mean	295	1024	35	23	481.0694	52	107	9766.8	130	1025
	min	295	1024	35	23	481.0694	52	107	9765	130	1025
t3	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0.6156	0	0
	gap	0	0	0	0	0	0	0	0.002	0	0
	mean	295	1024	35	23	481.0694	52	107	9767	130	1025
	min	295	1024	35	23	481.0694	52	107	9767	130	1025
t4	max	295	1024	35	23	481.0694	52	107	9767	130	1025
	std	0	0	0	0	0	0	0	0	0	0
	gap	0	0	0	0	0	0	0	0	0	0

As seen in Tables 5 and 6, the optimum values are achieved in all runs with all transfer functions for all problems, except for problem f8 with transfer functions s1 and t3. Therefore, the gap and standard deviation values are 0 for the problems for which the optimum values were achieved in all runs. The experimental results of the binHOA on medium-scale dataset are presented in Tables 7 and 8.

Table 7. The experimental results of the binHOA on medium-scale dataset (s1-u4)

TF	metric	ks_8a	ks_8b	ks_8c	ks_8d	ks_8e	ks_12a	ks_12b	ks_12c	ks_12d	ks_12e	ks_16a	ks_16b	ks_16c
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7838659	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
s1	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	9278.275	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.157	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7837711	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
s2	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	8914.278	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.1691	0	0

	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7837711	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
s3	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	8914.278	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.1691	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7834867	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
s4	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	6945.95	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.2053	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7837711	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
u1	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	8914.278	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.1691	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7838659	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
u2	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	9278.275	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.157	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7837711	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
u3	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	8914.278	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.1691	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7837711	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
u4	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	8914.278	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.1691	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7839607	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
z1	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	9529.764	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.1449	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7841503	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
z2	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	9726.275	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.1207	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7840555	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
z3	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	9677.521	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.1328	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7844347	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
z4	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	9278.275	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.0845	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7838659	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
t1	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	9278.275	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.157	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7840555	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
t2	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	9677.521	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.1328	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688563	6498597	5170626	6992404	5337472	7836763	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5682404	6498597	5170626	6992404	5337472	7832023	9352998	9151147
t3	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	1449.643	0	0	0	0	8423.201	0	0
	gap	0	0	0	0	0	0.0057	0	0	0	0	0.1811	0	0
	mean	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7838659	9352998	9151147
	min	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7832023	9352998	9151147
t4	max	3924400	3813669	3347452	4187707	4955555	5688887	6498597	5170626	6992404	5337472	7850983	9352998	9151147
	std	0	0	0	0	0	0	0	0	0	0	9278.275	0	0
	gap	0	0	0	0	0	0	0	0	0	0	0.157	0	0

Table 8. The experimental results of the binHOA on medium-scale dataset (z1-t4)

TF	metric	ks_16d	ks_16e	ks_20a	ks_20b	ks_20c	ks_20d	ks_20e	ks_24a	ks_24b	ks_24c	ks_24d	ks_24e
	mean	9339131	7763733	10718043	9802868	10712717	8919683	9357969	13549094	12207002	12448780	11810051	13935497
	min	9336691	7750491	10717569	9774200	10712572	8893733	9357969	13549094	12196303	12448780	11810051	13929872
s1	max	9348889	7769117	10727049	9818261	10714023	8929156	9357969	13549094	12210480	12448780	11810051	13940099
	std	5005.954	7570.012	2119.792	21523.03	446.6084	15509.06	0	0	6181.478	0	0	5220.043
	gap	0.1044	0.0693	0.084	0.1568	0.0122	0.1061	0	0	0.2183	0	0	0.033
	mean	9339741	7758802	10718043	9811242	10712717	8923843	9357969	13549094	12212803	12448780	11810051	13936589
	min	9336691	7748170	10717569	9765998	10712572	8893733	9357969	13549094	12210480	12448780	11810051	13929872
s2	max	9348889	7769117	10727049	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5419.104	8931.956	2119.792	17211.94	446.6084	12977.13	0	0	7150.967	0	0	4911.345
	gap	0.0979	0.1328	0.084	0.0715	0.0122	0.0595	0	0	0.1709	0	0	0.0252
	mean	9340350	7762744	10717569	9813855	10712790	8923155	9357969	13549094	12211642	12448780	11810051	13936043
	min	9336691	7748170	10717569	9774200	10712572	8893733	9357969	13549094	12210480	12448780	11810051	13929872
s3	max	9348889	7769117	10717569	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5735.04	9022.258	0	13561.69	531.5703	13045.18	0	0	5195.057	0	0	5098.916
	gap	0.0913	0.082	0	0.0449	0.0115	0.0672	0	0	0.1804	0	0	0.0291

	mean	9340350	7760839	10717569	9813855	10712717	8923155	9357969	13549094	12213965	12448780	11810051	13938669
	min	9336691	7748170	10717569	9774200	10712572	8893733	9357969	13549094	12210480	12448780	11810051	13930566
s4	max	9348889	7769117	10717569	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5735.04	9567.416	0	13561.69	446.6084	13045.18	0	0	8511.353	0	0	3492.391
	gap	0.0913	0.1066	0	0.0449	0.0122	0.0672	0	0	0.1614	0	0	0.0103
	mean	9339741	7765960	10718043	9813855	10712645	8925614	9357969	13549094	12214440	12448780	11810051	13938600
	min	9336691	7750491	10717569	9774200	10712572	8893733	9357969	13549094	12196745	12448780	11810051	13929872
u1	max	9348889	7769117	10727049	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5419.104	6522.447	2119.792	13561.69	324.4535	10902.97	0	0	10346.96	0	0	3664.195
	gap	0.0979	0.0406	0.084	0.0449	0.0129	0.0397	0	0	0.1575	0	0	0.0108
	mean	9340960	7766149	10718991	9813855	10712862	8927780	9357969	13549094	12215127	12448780	11810051	13938669
	min	9336691	7754276	10717569	9774200	10712572	8915396	9357969	13549094	12210480	12448780	11810051	13930566
u2	max	9348889	7769117	10727049	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5969.219	6090.618	3472.975	13561.69	595.4778	4235.239	0	0	9534.622	0	0	3492.391
	gap	0.0848	0.0382	0.0751	0.0449	0.0108	0.0154	0	0	0.1519	0	0	0.0103
	mean	9341570	7760897	10717569	9811652	10712790	8926768	9357969	13549094	12214126	12448780	11810051	13937681
	min	9336691	7748170	10717569	9774200	10712572	8895152	9357969	13549094	12210480	12448780	11810051	13929872
u3	max	9348889	7769117	10717569	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	6131.016	8629.159	0	16141.64	531.5703	8050.955	0	0	8472.317	0	0	4299.169
	gap	0.0783	0.1058	0	0.0673	0.0115	0.0267	0	0	0.1601	0	0	0.0173
	mean	9339131	7762381	10717569	9811242	10713080	8927385	9357969	13549094	12216288	12448780	11810051	13937135
	min	9336691	7748170	10717569	9765998	10712572	8893733	9357969	13549094	12210480	12448780	11810051	13929872
u4	max	9348889	7769117	10717569	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5005.954	8639.617	0	17211.94	710.0621	7920.824	0	0	10321.53	0	0	4649.295
	gap	0.1044	0.0867	0	0.0715	0.0088	0.0198	0	0	0.1424	0	0	0.0213
	mean	9341570	7765844	10717569	9818261	10712790	8926697	9357969	13549094	12215127	12448780	11810051	13939588
	min	9336691	7748170	10717569	9818261	10712572	8893733	9357969	13549094	12210480	12448780	11810051	13929872
z1	max	9348889	7769117	10717569	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	6131.016	6825.751	0	0	531.5703	8345.116	0	0	9534.622	0	0	2286.827
	gap	0.0783	0.0421	0	0	0.0115	0.0275	0	0	0.1519	0	0	0.0037
	mean	9342790	7765218	10717569	9818261	10712862	8929156	9357969	13549094	12216288	12448780	11810051	13939588
	min	9336691	7750491	10717569	9818261	10712572	8929156	9357969	13549094	12210480	12448780	11810051	13929872
z2	max	9348889	7769117	10717569	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	6257.442	6972.982	0	0	595.4778	0	0	0	10321.53	0	0	2286.827
	gap	0.0652	0.0502	0	0	0.0108	0	0	0	0.1424	0	0	0.0037
	mean	9344620	7767022	10717569	9818261	10712572	8929156	9357969	13549094	12217450	12448780	11810051	13940099
	min	9336691	7748170	10717569	9818261	10712572	8929156	9357969	13549094	12210480	12448780	11810051	13940099
z3	max	9348889	7769117	10717569	9818261	10712572	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5969.219	6447.351	0	0	0	0	0	0	10923.28	0	0	0
	gap	0.0457	0.0270	0	0	0	0	0	0	0.1329	0	0	0
	mean	9342790	7765538	10718517	9818261	10713007	8927385	9357969	13549094	12219773	12448780	11810051	13938669
	min	9336691	7748170	10717569	9818261	10712572	8893733	9357969	13549094	12210480	12448780	11810051	13930566
z4	max	9348889	7769117	10727049	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	6257.442	7475.954	2917.882	0	682.2056	7920.824	0	0	11677.48	0	0	3492.391
	gap	0.0652	0.0461	0.0795	0	0.0095	0.0198	0	0	0.1139	0	0	0.0103
	mean	9339131	7764796	10718517	9802047	10713007	8922071	9357969	13549094	12214601	12448780	11810051	13936078
	min	9336691	7748170	10717569	9765998	10712572	8893733	9357969	13549094	12196745	12448780	11810051	13929872
t1	max	9348889	7769117	10727049	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5005.954	7830.186	2917.882	22784.1	682.2056	14537.29	0	0	10307.06	0	0	5056.898
	gap	0.1044	0.0556	0.0795	0.1651	0.0095	0.0793	0	0	0.1562	0	0	0.0288
	mean	9339741	7762424	10718043	9816058	10712717	8922071	9357969	13549094	12215288	12448780	11810051	13938158
	min	9336691	7750491	10717569	9774200	10712572	8893733	9357969	13549094	12210480	12448780	11810051	13929872
t2	max	9348889	7769117	10727049	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5419.104	8499.847	2119.792	9852.339	446.6084	14537.29	0	0	9479.02	0	0	3985.849
	gap	0.0979	0.0862	0.084	0.0224	0.0122	0.0793	0	0	0.1506	0	0	0.0139
	mean	9340350	7758234	10718517	9802457	10712790	8918600	9357969	13549094	12209990	12448780	11810051	13935020
	min	9336691	7748170	10717569	9765998	10712572	8893733	9357969	13549094	12196303	12448780	11810051	13929872
t3	max	9348889	7769117	10727049	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5735.04	9333.396	2917.882	22166.53	531.5703	16546.04	0	0	9911.065	0	0	5212.928
	gap	0.0913	0.1401	0.0795	0.161	0.0115	0.1182	0	0	0.1939	0	0	0.0364
	mean	9339131	7761028	10718043	9809039	10713007	8917912	9357969	13549094	12209743	12448780	11810051	13933486
	min	9336691	7748170	10717569	9765998	10712572	8893733	9357969	13549094	12196745	12448780	11810051	13929872
t4	max	9348889	7769117	10727049	9818261	10714023	8929156	9357969	13549094	12233713	12448780	11810051	13940099
	std	5005.954	9387.702	2119.792	18993.79	682.2056	16369.14	0	0	7626.967	0	0	4980.915
	gap	0.1044	0.1041	0.084	0.0939	0.0095	0.1259	0	0	0.1959	0	0	0.0474

Tables 7 and 8 show that the optimum values are achieved in all runs with all transfer functions for the problems ks8_a – ks_12e, ks_16b-16c, ks_20e-24a, and ks_24c-24e. Therefore, the gap and standard deviation values are 0 for these problems.

The gap metric is considered an appropriate measure for evaluating the performance of the transfer functions, since it summarizes the algorithm's success as the percentage difference between the obtained and the actual optimum values. The gap values produced by the binHOA for the small-scale dataset are presented in Table 9. Table 9 shows that the optimum values are achieved in all runs, meaning that the gap values are 0 with all transfer functions for all problems, except for problem f8 with transfer functions s1 and t3. However, to assess the performance of the transfer functions, only one problem instance is insufficient; therefore, the medium-scale dataset will also be considered.

Table 9. Gap values produced by the binHOA for the small-scale dataset

TF	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
s1	0	0	0	0	0	0	0	0.0041	0	0
s2	0	0	0	0	0	0	0	0	0	0
s3	0	0	0	0	0	0	0	0	0	0
s4	0	0	0	0	0	0	0	0	0	0
u1	0	0	0	0	0	0	0	0	0	0
u2	0	0	0	0	0	0	0	0	0	0
u3	0	0	0	0	0	0	0	0	0	0
u4	0	0	0	0	0	0	0	0	0	0
z1	0	0	0	0	0	0	0	0	0	0
z2	0	0	0	0	0	0	0	0	0	0
z3	0	0	0	0	0	0	0	0	0	0
z4	0	0	0	0	0	0	0	0	0	0
t1	0	0	0	0	0	0	0	0	0	0
t2	0	0	0	0	0	0	0	0	0	0
t3	0	0	0	0	0	0	0	0.002	0	0
t4	0	0	0	0	0	0	0	0	0	0

The gap values produced by the binHOA for the medium-scale dataset are presented in Tables 10 and 11.

Table 10. Gap values produced by the binHOA for the medium-scale dataset (ks_8a-ks_16c)

TF	ks_8a	ks_8b	ks_8c	ks_8d	ks_8e	ks_12a	ks_12b	ks_12c	ks_12d	ks_12e	ks_16a	ks_16b	ks_16c
s1	0	0	0	0	0	0	0	0	0	0	0.157	0	0
s2	0	0	0	0	0	0	0	0	0	0	0.1691	0	0
s3	0	0	0	0	0	0	0	0	0	0	0.1691	0	0
s4	0	0	0	0	0	0	0	0	0	0	0.2053	0	0
u1	0	0	0	0	0	0	0	0	0	0	0.1691	0	0
u2	0	0	0	0	0	0	0	0	0	0	0.157	0	0
u3	0	0	0	0	0	0	0	0	0	0	0.1691	0	0
u4	0	0	0	0	0	0	0	0	0	0	0.1691	0	0
z1	0	0	0	0	0	0	0	0	0	0	0.1449	0	0
z2	0	0	0	0	0	0	0	0	0	0	0.1207	0	0
z3	0	0	0	0	0	0	0	0	0	0	0.1328	0	0
z4	0	0	0	0	0	0	0	0	0	0	0.0845	0	0
t1	0	0	0	0	0	0	0	0	0	0	0.157	0	0
t2	0	0	0	0	0	0	0	0	0	0	0.1328	0	0
t3	0	0	0	0	0	0.0057	0	0	0	0	0.1811	0	0
t4	0	0	0	0	0	0	0	0	0	0	0.157	0	0

Table 11. Gap values produced by the binHOA for the medium-scale dataset (ks_16d-ks_24e)

TF	ks_16d	ks_16e	ks_20a	ks_20b	ks_20c	ks_20d	ks_20e	ks_24a	ks_24b	ks_24c	ks_24d	ks_24e
s1	0.1044	0.0693	0.084	0.1568	0.0122	0.1061	0	0	0.2183	0	0	0.033
s2	0.0979	0.1328	0.084	0.0715	0.0122	0.0595	0	0	0.1709	0	0	0.0252
s3	0.0913	0.082	0	0.0449	0.0115	0.0672	0	0	0.1804	0	0	0.0291
s4	0.0913	0.1066	0	0.0449	0.0122	0.0672	0	0	0.1614	0	0	0.0103
u1	0.0979	0.0406	0.084	0.0449	0.0129	0.0397	0	0	0.1575	0	0	0.0108
u2	0.0848	0.0382	0.0751	0.0449	0.0108	0.0154	0	0	0.1519	0	0	0.0103
u3	0.0783	0.1058	0	0.0673	0.0115	0.0267	0	0	0.1601	0	0	0.0173
u4	0.1044	0.0867	0	0.0715	0.0088	0.0198	0	0	0.1424	0	0	0.0213
z1	0.0783	0.0421	0	0	0.0115	0.0275	0	0	0.1519	0	0	0.0037
z2	0.0652	0.0502	0	0	0.0108	0	0	0	0.1424	0	0	0.0037
z3	0.0457	0.027	0	0	0	0	0	0	0.1329	0	0	0
z4	0.0652	0.0461	0.0795	0	0.0095	0.0198	0	0	0.1139	0	0	0.0103
t1	0.1044	0.0556	0.0795	0.1651	0.0095	0.0793	0	0	0.1562	0	0	0.0288
t2	0.0979	0.0862	0.084	0.0224	0.0122	0.0793	0	0	0.1506	0	0	0.0139
t3	0.0913	0.1401	0.0795	0.161	0.0115	0.1182	0	0	0.1939	0	0	0.0364
t4	0.1044	0.1041	0.084	0.0939	0.0095	0.1259	0	0	0.1959	0	0	0.0474

The best gap values for each type of transfer function are highlighted with different colors (S-shaped: green, U-shaped: blue, Z-shaped: pink, and T-shaped: orange) in Tables 10 and 11. As seen in Tables 10 and 11, all transfer functions except for t3 achieve the optimum value in all runs for lower-dimensional problem instances (ks_8a–ks_12e). For higher-dimensional problem instances ks_16b–16c, ks_20e–24a, and ks_24c–24d, the

optimum values are achieved with all transfer functions in all runs. Among the S-shaped, U-shaped, and T-shaped transfer functions, s4, u2, and t2 achieved the lowest gap values in 21 out of 25 problems, respectively, while z3 outperformed all other Z-shaped transfer functions by achieving the lowest gap value in 23 out of 25 problems. Overall, z3 demonstrated the best performance among all transfer functions; therefore, the gap values obtained using the binHOA with the z3 transfer function are used for comparison with other algorithms.

In order to evaluate the performance of the algorithms across multiple problems and rank them accordingly, the Friedman test is employed. The Friedman test is a non-parametric statistical method that compares and ranks the performance of more than two algorithms across multiple datasets (Demšar, 2006). The Friedman test results are reported at the bottom of the tables. The gap values produced by the binHOA and the compared algorithms, namely PSO, DE, ABC, GWO, WOA, and SSA, for the small-scale dataset are presented in Table 12.

Table 12. Gap values obtained by the algorithms on the small-scale dataset

	binHOA	PSO	DE	ABC	GWO	WOA	SSA
f1	0	0	0	0	0	0	6.1186
f2	0	0.083	0	0	0	0	1.8311
f3	0	0	0	0	0	0	0
f4	0	0	0	0	0	0	0.2174
f5	0	0	0	0	0	0	8.2732
f6	0	0	0	0	0	0	2.0192
f7	0	0	0	0	0	0	2.4766
f8	0	0.0041	0	0.0036	0.001	0.0015	0.1797
f9	0	0	0	0	0	0	0.4615
f10	0	0.1805	0	0	0	0	1.3951
MeanRank	3.25	4.3	3.25	3.6	3.4	3.5	6.7
Rank	1	5	1	4	2	3	6

Table 13. Gap values obtained by the algorithms on the medium-scale dataset

Gap	binHOA	PSO	DE	ABC	GWO	WOA	SSA
ks_8a	0	0	0	0	0	0	0.97
ks_8b	0	0	0.0406	0	0	0	1.0999
ks_8c	0	0.4894	0.0334	0	0	0	2.1931
ks_8d	0	0	0	0	0	0	1.3531
ks_8e	0	0.1984	0.0992	0	0	0	1.9231
ks_12a	0	0.0255	0.0057	0	0.0057	0	1.2751
ks_12b	0	0.1978	0.0394	0	0	0	1.9365
ks_12c	0	0.0779	0	0	0	0	2.9817
ks_12d	0	0.0455	0	0	0	0	1.8567
ks_12e	0	0.2297	0.1033	0	0	0	3.5616
ks_16a	0.1328	0.1996	0.0242	0.0901	0.0845	0.1449	2.2403
ks_16b	0	0.1416	0.1136	0	0	0	3.0207
ks_16c	0	0.2888	0.0381	0	0	0	3.5639
ks_16d	0.0457	0.2926	0.0867	0.0391	0.0587	0.0783	2.1494
ks_16e	0.027	0.168	0.0096	0.0764	0.121	0.1041	2.6041
ks_20a	0	0.2903	0	0.3015	0.0707	0.084	2.2978
ks_20b	0	0.341	0.0355	0.0266	0.0939	0.0715	2.4728
ks_20c	0	0.4089	0.0076	0.0343	0.0081	0.0115	2.6748
ks_20d	0	0.3837	0.0154	0	0.0154	0.0785	3.0631
ks_20e	0	0.2359	0.0008	0	0	0	2.0923
ks_24a	0	0.3751	0.0995	0	0	0	2.5354
ks_24b	0.1329	0.3182	0.0106	0.4648	0.1253	0.148	2.1391
ks_24c	0	0.3928	0.0405	0	0	0	2.4621
ks_24d	0	0.329	0.0105	0.1931	0	0.0423	2.4899
ks_24e	0	0.2339	0.0068	0.1257	0.0142	0.0215	2.108
MeanRank	2.36	5.6	3.66	3.12	2.96	3.3	7
Rank	1	6	5	3	2	4	7

Table 12 shows that the binHOA and DE algorithms found the optimum values in all runs for all problem instances. The MeanRank and Rank rows denote the Friedman test results and the corresponding rank values, respectively. According to the rank values, the binHOA and DE algorithms outperform the other algorithms. In

contrast, SSA ranked last. The second-ranked algorithm is GWO, with a mean rank value of 3.4. The gap values produced by the binHOA and the compared algorithms, namely PSO, DE, ABC, GWO, WOA, and SSA, for the medium-scale dataset are presented in Table 13.

Table 13 shows that the binHOA found the optimum values in all runs for 21 of the 25 problem instances, thus producing better gap values than the other algorithms for most of the problems. According to the rank values, the binHOA, with a mean rank value of 2.36, outperforms the other algorithms. In contrast, SSA ranked last among the compared algorithms. The second-ranked algorithm is GWO, with a mean rank value of 2.96.

Conclusion

In this study, the recently proposed hiking optimization algorithm is binarized using four different types of transfer functions and the binHOA is proposed. The proposed binHOA is applied to the 0/1 knapsack problem. In addition, to improve the performance of the proposed algorithm, a stochastic repair function is adopted. Initially, the performance of 16 different transfer functions is assessed on small-scale and medium-scale benchmark datasets. The experimental results indicate that the z3 transfer function outperforms the other functions; thus, the results obtained using the z3 transfer function are compared against the binary versions of six well-known algorithms. According to the experimental results, the binHOA outperforms the other algorithms for most of the problem instances in the datasets. The overall experimental results demonstrate that the proposed algorithm serves as a competitive and effective approach for solving the 0/1 knapsack problem.

Recommendations

For future studies, the HOA can be evaluated on large-scale datasets to observe its limitations, and further modifications can be applied to enhance the overall performance of the algorithm.

Scientific Ethics Declaration

* The author declares that the scientific ethical and legal responsibility of this article published in EPSTEM journal belongs to the author.

Conflict of Interest

* The author declares that there is no conflict of interest.

Funding

* No funding was received for this study.

Acknowledgements or Notes

* This article was presented as an oral presentation at the International Conference on Basic Sciences and Technology (www.icbast.net) held in Konya/Türkiye on April 02-05, 2026.

References

Abdel-Basset, M., Mohamed, R., Chakrabortty, R. K., Ryan, M., & Mirjalili, S. (2021). New binary marine predators optimization algorithms for 0–1 knapsack problems. *Computers & Industrial Engineering, 151*, 106949.

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

Büyüköz, G. O., & Haklı, H. (2023). Binary honey badger algorithm for 0–1 knapsack problem. *Journal of Intelligent Systems: Theory and Applications, 6*(2), 108–118.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Erdoğan, F., Karakoyun, M., & Gülcü, Ş. (2025). An effective binary dynamic grey wolf optimization algorithm for the 0–1 knapsack problem. *Multimedia Tools and Applications, 84*(21), 23279–23311.

Ervural, B., & Haklı, H. (2023). A binary reptile search algorithm based on transfer functions with a new stochastic repair method for 0–1 knapsack problems. *Computers & Industrial Engineering, 178*, 109080.

Guo, S.-S., Wang, J.-S., & Guo, M.-W. (2020). Z-shaped transfer functions for binary particle swarm optimization algorithm. *Computational Intelligence and Neuroscience, 2020*(1), 6502807.

He, Y., Zhang, F., Mirjalili, S., & Zhang, T. (2022). Novel binary differential evolution algorithm based on taper-shaped transfer functions for binary optimization problems. *Swarm and Evolutionary Computation, 69*, 101022.

Ihsan, A., & Sag, T. (2025). Binary puma optimizer: A novel approach for solving 0–1 knapsack problems and the uncapacitated facility location problem. *Applied Sciences, 15*(18), 9955. Retrieved from https://www.mdpi.com/2076-3417/15/18/9955

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*(3), 459–471.

Kennedy, J., & Eberhart, R. (1995, November 27–December 1). Particle swarm optimization. *Proceedings of the International Conference on Neural Networks (ICNN'95)*.

Limane, A., Zitouni, F., Ferhat, A., & Lakbichi, R. (2024, April 24–25). Binary electric eel foraging optimization algorithm for solving 0–1 knapsack problems. *2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*.

Mahafzah, B. A., Al-Tawil, M., & Krunz, M. (2025). 0/1 knapsack problem on hyper hexa-cell interconnection network. *Cluster Computing, 29*(1), 2.

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

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.

Mirjalili, S., Zhang, H., Mirjalili, S., Chalup, S., & Noman, N. (2020). A novel U-shaped transfer function for binary particle swarm optimisation. In A. K. Nagar, K. Deep, J. C. Bansal, & K. N. Das (Eds.), *Soft computing for problem solving 2019*. Springer.

Oladejo, S. O., Ekwe, S. O., & Mirjalili, S. (2024). The hiking optimization algorithm: A novel human-based metaheuristic approach. *Knowledge-Based Systems, 296*, 111880.

Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research, 141*(2), 382–392.

Pisinger, D. (2005). Where are the hard knapsack problems? *Computers & Operations Research, 32*(9), 2271–2284.

Reniers, G. L. L., & Sörensen, K. (2013). An approach for optimal allocation of safety resources: Using the knapsack problem to take aggregated cost-efficient preventive measures. *Risk Analysis, 33*(11), 2056–2067.

Sakib, F. F., Joy, J. S., Juel, S. I., & Hasan, M. M. (2025, January 11–12). Applying heuristic algorithms for solving the 0–1 knapsack problem: An experimental analysis. *2025 4th International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*.

Shu, Z., Ye, Z., Zong, X., Liu, S., Zhang, D., Wang, C., & Wang, M. (2022). A modified hybrid rice optimization algorithm for solving 0–1 knapsack problem. *Applied Intelligence, 52*(5), 5751–5769.

Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.

Sun, W.-Z., Zhang, M., Wang, J.-S., Guo, S.-S., Wang, M., & Hao, W.-K. (2021). Binary particle swarm optimization algorithm based on z-shaped probability transfer function to solve 0–1 knapsack problem. *IAENG International Journal of Computer Science, 48*(2), 294.

Taillandier, F., Fernandez, C., & Ndiaye, A. (2017). Real estate property maintenance optimization based on multiobjective multidimensional knapsack problem. *Computer-Aided Civil and Infrastructure Engineering, 32*(3), 227–251.

Xue, J., & Shen, B. (2020). A novel swarm intelligence optimization approach: Sparrow search algorithm. *Systems Science & Control Engineering, 8*(1), 22–34.

Yildizdan, G., & Baş, E. (2023). A novel binary artificial jellyfish search algorithm for solving 0–1 knapsack problems. *Neural Processing Letters, 55*(7), 8605–8671.

Zhou, Y., Shi, Y., Wei, Y., Luo, Q., & Tang, Z. (2023). Nature-inspired algorithms for 0–1 knapsack problem: A survey. *Neurocomputing, 554*, 126630.

Zou, D., Gao, L., Li, S., & Wu, J. (2011). Solving 0–1 knapsack problem by a novel global harmony search algorithm. *Applied Soft Computing, 11*(2), 1556–1564.

Author(s) Information

Aybuke Babadag
Konya Technical University
Ardıçlı Mahallesi Rauf Orbay Cd. 42250 Selçuklu/Konya,
Turkiye
ORCID iD: https://orcid.org/0000-0002-4091-6684
*Contact e-mail: *ababadag@ktun.edu.tr*

*Corresponding author(s) email

To cite this article:

Babadag, A. (2026). A binary hiking optimization algorithm for 0/1 knapsack problem. *The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 40,* 425-438.