BINARY HONEY BADGER ALGORITHM ENHANCED WITH TIME-VARYING SIGMOID TRANSFER FUNCTION AND CROSSOVER STRATEGY

Gülnur YILDIZDAN[1*], Emine BAŞ[2]
[1] Selcuk University, Kulu Vocational School, Department of Computer Tecnologies, Konya
ORCID No: https://orcid.org/0000-0001-6252-9012
[2] Konya Technical University, Faculty of Engineering and Nature Sciences, Department of Software Engineering, Konya
ORCID No: https://orcid.org/0000-0003-4322-6010

Keywords
Binary optimization, Crossover strategy, Honey badger algorithm, Knapsack problems, Time-varying transfer function

Abstract
Modeling the foraging behavior of honey badgers, the Honey Badger Algorithm (HBA) is a recently proposed metaheuristic algorithm. In this study, a binary version of this algorithm that was proposed for solving continuous optimization problems was developed. The S-shaped transfer function and crossover strategy were used to transform the continuous algorithm into a binary algorithm. Eight S-shaped transfer functions with constant and time-varying features were used, and the most successful function was determined. Additionally, the effect of time-varying transfer functions was examined. Three strategies, single-point, two-point, and uniform, were applied as crossover strategies, and the uniform strategy, which was more successful than others, was integrated into the algorithm. The binary algorithm (BinHBA) developed in this way was tested on a total of twenty-seven knapsack problems, fifteen small-scale and twelve large-scale. Statistical tests were employed to analyze the results and compare them with methods found in the existing literature. The results showed that the proposed BinHBA for binary optimization problems is effective and preferable.

ZAMANLA DEĞİŞEN SİGMOİD TRANSFER FONKSİYONU VE ÇAPRAZLAMA STRATEJİSİ İLE GELİŞTİRİLMİŞ İKİLİ BAL PORSUĞU ALGORİTMASI

Anahtar Kelimeler
İkili optimization, Çaprazlama stratejisi, Bal porsuğu algoritması, Sırt çantası problemleri, Zamanla-değişen transfer fonksiyonu

Öz
Bal porsuklarının yiyecek arama davranışını modelleyen Bal Porsuğu Algoritması (HBA), yakın zamanda önerilen bir meta-sezgisel algoritmadır. Bu çalışmada, sürekli optimizasyon problemlerinin çözümü için önerilen bu algoritmanın ikili versiyonu geliştirildi. Sürekli algoritmayı ikili bir algoritmaya dönüştürmek için S-şekilli transfer fonksiyonu ve çaprazlama stratejisi kullanıldı. Sabit ve zamanla değişen özelliklere sahip sekiz adet S-şekilli transfer fonksiyonu kullanıldı ve en başarılı fonksiyon belirlendi. Ayrıca zamanla değişen transfer fonksiyonlarının etkisi de incelendi. Çaprazlama stratejisi olarak tek nokta, iki nokta ve tekdüze olmak üzere üç strateji uygulandı ve diğerlerinden daha başarılı olan tekdüze stratejisi algoritmaya entegre edildi. Bu şekilde geliştirilen ikili algoritma (BinHBA), on beşi küçük ölçekli ve on ikisi büyük ölçekli olmak üzere toplam yirmi yedi sırt çantası problemi üzerinde test edildi. Sonuçları analiz etmek ve mevcut literatürde bulunan yöntemlerle karşılaştırmak için istatistiksel testler kullanıldı. Sonuçlar, ikili optimizasyon problemleri için önerilen BinHBA'nın etkili ve tercih edilebilir olduğunu gösterdi.

Araştırma Makalesi		Research Article	
Başvuru Tarihi	: 02.05.2024	Submission Date	: 02.05.2024
Kabul Tarihi	: 23.01.2025	Accepted Date	: 23.01.2025

* Sorumlu yazar: gavsar@selcuk.edu.tr
https://doi.org/10.31796/ogummf.1477088

1. Introduction

The 0-1 knapsack problem optimization problems are binary optimization problems (BOPs), which are an important class of combinatorial optimization problems. BOPs take the value 0 or 1 as the decision variable. The algorithms used to solve these problems, where the solution space is expressed as $\{0,1\}_n$, are inadequate, especially for large-scale problems. For this reason, researchers in this field have turned to non-deterministic algorithms in recent years, and evolutionary algorithms have become the most prominent in this category (He, Zhang, Mirjalili, and Zhang, 2022).

The number of evolutionary algorithms has been increasing rapidly in recent years. Most algorithms are initially proposed for continuous problems and then binarized to solve binary optimization problems. One of these algorithms is the Honey Badger Algorithm (Hashim, Houssein, Hussain, Mabrouk, and Al-Atabany, 2022). The algorithm has advantages such as exhibiting a dynamic search behavior in searching for a food source, thus maintaining the balance between exploration and exploitation, and having few parameters that need to be adjusted (Yasear and Ghanimi, 2022). On the other hand, the local improvement ability of the algorithm is weak, and it is insufficient to escape from the local optimum in solving complex problems (Jin, Li, Zhang, and Zhang, 2023). Despite these drawbacks encountered in most metaheuristic algorithms, it has been used to solve many problems in the literature because it is simple and easy to implement.

Hu et al. proposed a modified HBA algorithm based on the Bernoulli shift map, piecewise optimal decreasing neighborhood, and strategy-adaptive horizontal migration to solve the unmanned aerial vehicle path planning problem (Hu, Zhong and Wei, 2023). Abasi, Aloqaily, and Guizani proposed a modified HBA to optimize the hyperparameters of a convolutional neural network for sleep apnea detection (Abasi, Aloqaily, and Guizani, 2023). A novel honey badger optimization approach utilizing ensemble learning-based vehicle detection and classification methods was presented by Aljebreen et al. (Aljebreen et al., 2023). Majumdar, Mitra, and Bhattacharya proposed a modified HBA using lens opposition-based learning in the initial stage to improve population quality and diversity and achieve better discovery, and they proved the success of the proposed algorithm on CEC functions (Majumdar, Mitra, and Bhattacharya, 2023). Jain, Ding, and Kotecha developed a new fuzzy deep neural network for cloud environments with HBA for privacy-preserving intrusion detection techniques (Jain, Ding, and Kotecha, 2023). Altuwairiqi proposed a model to solve security and energy problems (Altuwairiqi, 2024). He also created enhanced HBA-based multi-hop routing optimized for wireless sensor networks. In order to effectively forecast COVID-19 situations, Qasem proposed a hybrid intelligent model that combined an artificial neural network with HBA (Qasem, 2024). Huang et al. proposed an advanced HBA that utilizes orthogonal opposition-based learning, a balance pooling strategy, and differential evolution to improve the HBA's performance. The method was tested for CEC 2022 and engineering problems (Huang et al., 2024). Wang et al. developed a novel wrapper feature selection approach based on the binary honey badger algorithm, integrating chaotic sub-swarm and Lévy flight approaches, to identify a traffic-based Internet of Things device (Wang, Kang, Sun, and Li, 2024). Büyüköz and Haklı discretized HBA using transfer functions and used it to solve 0-1 knapsack problems (Büyüköz and Haklı, 2023). To address the portfolio selection problem, Ni et al. created a novel hybrid method that combines quadratic programming and the binary honey-badger algorithm (Ni, Wang, Huang, and Li, 2022).

As can be seen above, although HBA is preferred in many areas for solving continuous optimization problems, its use in solving binary optimization problems is less common. The scarcity of proposed binary HBA versions and the desire to contribute to the literature on the analysis of HBA's performance on binary problems were the motivations for this study.

The following is a list of the study's contributions to the literature:

- The effect of using time-varying transfer functions for the binarization of HBA was investigated for the first time.
- The effect of constant and time-varying S-shaped transfer functions on the algorithm performance was examined.
- The effect of integrating crossover strategies into HBA on performance was evaluated.
- A new binary HBA algorithm was presented to the literature by integrating the most successful transfer function and crossover strategy determined into the algorithm.
- Both small- and large-scale knapsack problems were used to test the proposed algorithm. As a result, it was possible to see how the dimension increase affected the proposed algorithm.
- The results obtained were compared with the algorithms in the literature to demonstrate the success of the algorithm.

The following study sections are arranged as follows: In Section 2, the HBA algorithm, the proposed binary HBA algorithm, and the knapsack problem are explained. Section 3 presents the experimental test

results performed to demonstrate the performance of BinHBA. Section 4 delineates conclusions and prospective works for further research.

2. Material and Methods

2.1. Honey Badger Algorithm

The Honey Badger Algorithm (HBA) is a metaheuristic algorithm proposed by Hashim et al. (Hashim et al., 2022) in 2022. The algorithm is inspired by the behavior of the honey badger when searching for food, such as smelling, digging, and following the honeyguide bird. HBA is divided into two phases, the "digging phase" and the "honey phase", and is mathematically modeled as given below.

- Initialization

The algorithm is initialized with a population of possible solutions. Where N is the number of honey badgers, each individual in the population is placed at their starting position according to Equation 1.

$$X_i = Lb_i + r^1 \times (Ub_i - Lb_i) \quad (1)$$

In the equation, r^1 is a random number between 0 and 1. Lb and Ub indicate the lower and upper bound values of the search space, respectively. X_i represents the i.honey badger's position.

- Intensity (I) definition

Density depends on the concentration power of the prey and the distance between it and the honey badger. The higher the scent, the faster the movement. The opposite situation is expressed in Equation 2 according to the Inverse Square Law (Kapner et al., 2007).

$$I_i = r^2 \times \frac{S}{4\pi D_i^2} \quad (2)$$

$$S = (X_i - X_{i+1})^2 \quad (3)$$

$$D_i = X_P - X_i \quad (4)$$

where S is the concentration power. In Equation 2, D_i represents the distance between the prey and the ith honey badger. r^2 is a random number in the range of 0 to 1. Also, X_P is the prey position.

- Update density factor

The density factor (α) controls the transition from the exploration to the exploitation phase and has a character that reduces randomness by decreasing over time. It is formulated according to Equation 5.

$$\alpha = C \times exp\left(\frac{-t}{t_{max}}\right) \quad (5)$$

where c is a constant value ($C \geq 1$ (default = 2)). While t is the iteration number, t_{max} indicates the maximum iteration number.

- Escaping from the local optimum

This and the next two steps are used to get out of the local optimum. In order to fully discover the search space, the algorithm makes use of an F flag that permits changing it.

- Updating the position of agents

HBA's position update process is divided into two phases: "digging phase" and "honey phase".

Digging phase: In this phase, a honey badger moves in a manner similar to a cardioid, and this movement is formulated as in Equation 6.

$$X_{new} = X_P + F \times \beta \times I \times X_P + F \times r^3 \times \alpha \times D_i \times |\cos(2\pi r^4) \times [1 - \cos(2\pi r^5)]| \quad (6)$$

In the equation, X_{new} is the new candidate position, and X_P is the best prey position found so far. The honey badger's ability to take in food is denoted by β ($\beta \geq 1$ (default = 6)). r^3, r^4, and r^5 are random numbers between 0 and 1. D_i represents the distance between the prey and the ith honey badger. F is the flag that changes the search direction and is formulated using Equation 7.

$$F = \begin{cases} 1 & r^6 \leq 0.5 \\ -1 & else \end{cases} \quad (7)$$

Honey phase: It is the phase in which a honey badger follows the honeyguide bird to reach the beehive and is formulated according to Equation 8.

$$X_{new} = X_P + F \times r^7 \times \alpha \times D_i \quad (8)$$

In the equation, r^7 is a random number between 0 and 1. At this phase, the search is affected by time-varying search behavior (α). In addition, a honey badger might discover disturbance F.

In Figure 1, the HBA pseudocode is presented.

2.2. Binary Honey Badger Algorithm (BinHBA)

The standard HBA is proposed for continuous optimization problems and needs to be redesigned to solve binary optimization problems. In binary optimization, decision variables have a search space

where they can take the value "0" or "1". In these problems, each solution is represented by a series of decision variables that take "0" or "1", where the value 0 represents absence and the value 1 represents presence (Karakoyun and Ozkis, 2022).

1. Determine parameters t_{max}, N, β,C
2. Initialize population of N individuals
3. Evaluate the fitness (f_i) of each honey badger position (X_i).
4. Find best position (X_P) and assign fitness to f_P.
5. **While** $t \leq t_{max}$ **do**
6. Utilize Equation 5 to update the decreasing factor α.
7. **For** i=1 **to** N **do**
8. Utilize Equation 2 to calculate the intensity I_i.
9. **If** $r < 0.5$ **then**
10. Use Equation 6 to update the position X_{new}.
11. **Else**
12. Use Equation 8 to update the position X_{new}.
13. **End if**
14. Examine new position (X_{new}) and assign to f_{new}.
15. **If** $f_{new} \leq f_i$ **then**
16. Assign $X_i = X_{new}$ and $f_i = f_{new}$
17. **End if**
18. **If** $f_{new} \leq f_P$ **then**
19. Assign $X_P = X_{new}$ and $f_P = f_{new}$
20. **End if**
21. **End for**
22. **End while**
23. Display X_P

Figure 1. HBA's Pseudocode

In this section, a binary HBA algorithm is proposed to solve binary optimization problems, and the steps followed in creating the proposed algorithm are listed below.

- Transfer function

The 0-1 knapsack problems chosen for this study are combinatorial problems, and for HBA to solve these problems, it must be transformed to work in the binary search space. The most common way to perform this transformation is to use the transfer function. Transfer functions are responsible for ensuring that if the candidate solution defined by 0 and 1 turns into a continuous value after the algorithm steps, this value is converted back to 0 or 1 according to Equation 9. In the equation, ∂ represents the value in a certain dimension of the solution, while F(∂) indicates the value obtained from the transfer function for the value ∂. Additionally, R is a random number in the range of 0 to 1 (Yildizdan and Bas, 2024; Yildizdan and Baş, 2023).

$$F(\partial) = \begin{cases} 1 & if\ F(\partial) > \mathcal{R} \\ 0 & else \end{cases} \quad (9)$$

More generally, transfer functions are used to determine the probability that a position value will change to 0 or 1 based on the value of the step vector (velocity) for a given dimension in the current iteration. The step vector is defined as the movement magnitude for the current individual. In the exploitation phase of the algorithm, the step vector value is small, and a detailed search is performed in small steps. On the contrary, in the exploration phase, the step vector value is large, and it explores different regions of the search space in large steps. In an algorithm where the probability of changing location is calculated using the transfer function, transfer functions significantly affect the balance between exploration and exploitation. If the transfer function does not change, the probability of changing positions remains the same throughout the optimization process. On the other hand, if the transfer function is changed, the effect of the step vector on position updating is utilized to explore and exploit the search space (Mafarja et al., 2018).

To better balance exploration and exploitation, Mafarja et al. showed that using time-varying transfer functions is an effective strategy (Mafarja et al., 2018). One of the transfer functions frequently used in the literature is the S-shaped transfer function (Al-Betar, Hammouri, Awadallah, and Abu Doush, 2021; Feda et al., 2024). Based on what was mentioned above, in this study, the effect of using constant S-shaped and time-varying S-shaped transfer functions on HBA was examined. Table 1 lists the transfer functions that were employed in the study. Figures 2 and 3 depict these transfer functions graphically. In time-varying transfer functions, the value of τ is formulated as shown in Equation 10 and starts with an initial value and progressively decreases over the iterations, meaning it is a time-varying variable. t_{max} is the maximum number of iterations, t is the current iteration, and τ_{min} and τ_{max} are the min and max values of the control parameter τ in the equation. In this study, τ_{min}= 0.01 and τ_{max} = 4 were used, as in Mafarja et al.'s study (Mafarja et al., 2018).

$$\tau = \left(1 - \frac{t}{t_{max}}\right)\tau_{max} + \tau_{min}\left(\frac{t}{t_{max}}\right) \quad (10)$$

Table 1. S-Shaped Transfer Functions

Constant	Time-varying
S1 $F(c) = \frac{1}{1+e^{-\partial}}$	S1(tv) $F(\partial,\tau) = \frac{1}{1+e^{\frac{-\partial}{\tau}}}$
S2 $F(\partial) = \frac{1}{1+e^{-2*\partial}}$	S2(tv) $F(\partial,\tau) = \frac{1}{1+e^{\frac{-2*\partial}{\tau}}}$
S3 $F(\partial) = \frac{1}{1+e^{-\frac{\partial}{2}}}$	S3(tv) $F(\partial,\tau) = \frac{1}{1+e^{-\frac{\partial}{2\tau}}}$
S4 $F(\partial) = \frac{1}{1+e^{-\frac{\partial}{3}}}$	S4(tv) $F(\partial,\tau) = \frac{1}{1+e^{-\frac{\partial}{3\tau}}}$

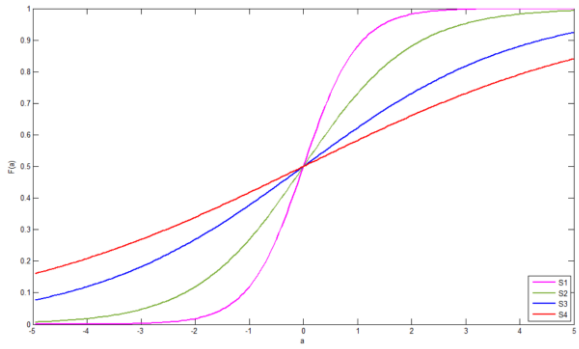
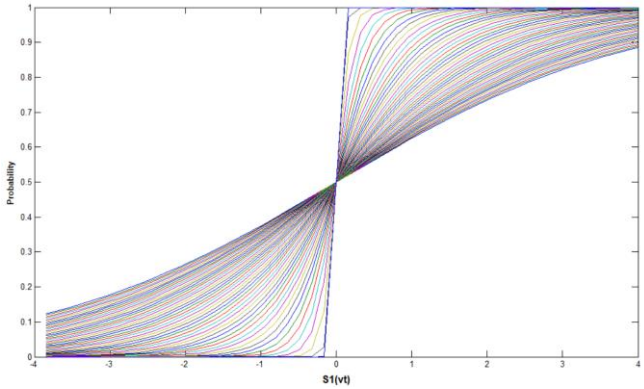
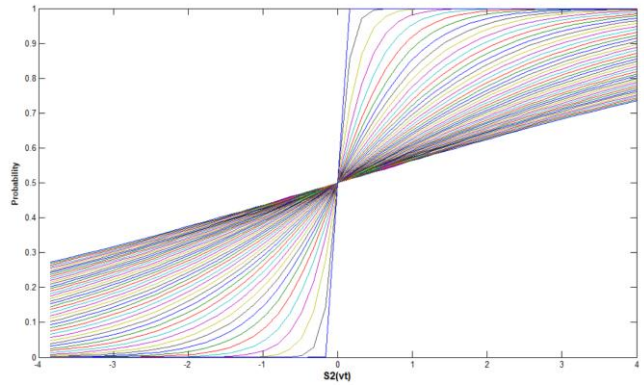
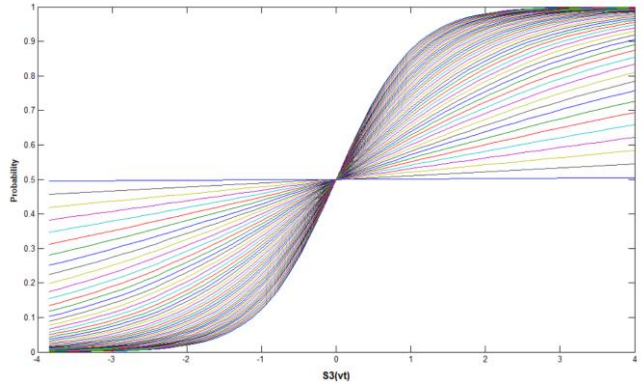


Figure 2. S-Shaped Transfer Functions (Constant)







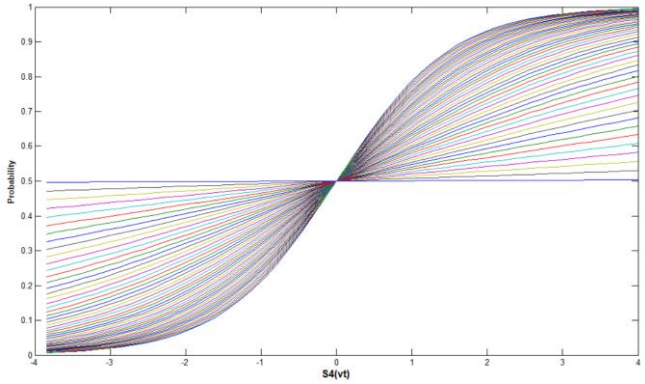


Figure 3. S-Shaped Transfer Functions (Time-varying (t_{max} = 50, τ_{min}= 0.01 and τ_{max} = 4))

- Crossover strategy

The crossover strategy was incorporated into the proposed BinHBA to strengthen its exploitation performance. In the crossover strategy, every solution in the population is combined with a different solution that has been chosen from the population based on the crossover method that has been specified (Awadallah, Hammouri, Al-Betar, Braik, and Elaziz, 2022). In this study, three different crossover strategies, which are frequently used in the literature, were integrated into the algorithm and their effects on performance were examined. The crossover operation is formulated as in Equation 11. In the equation, $\otimes$ denotes the crossover operator between two solutions. $X(t,i)$ is the existing solution at the ith position, and $X(t,b)$ is also the best solution in the population (for this study).

$$X(t,i) = \otimes\,(X(t,i)\,, X(t,b)) \qquad (11)$$

The following is a list of the three kinds of crossover operators that BinHBA uses (Awadallah et al., 2022):

- One-point crossover: In this crossover strategy, a random dimension is selected in the existing solution. All dimensions after this point are then swapped between the two solutions (Figure 4(a)).
- Two-point crossover: In this crossover strategy, two dimensions are determined on the existing solution. Then the dimensions between these two points are swapped between the two solutions (Figure 4(b)).
- Uniform crossover: The dimensions of the existing solution and the dimensions of the best solution are randomly selected in a determined ratio and swapped. For instance, if the ratio is 0.5, the elements of the optimal solution are swapped for half of the existing solution (Figure 4(c)).

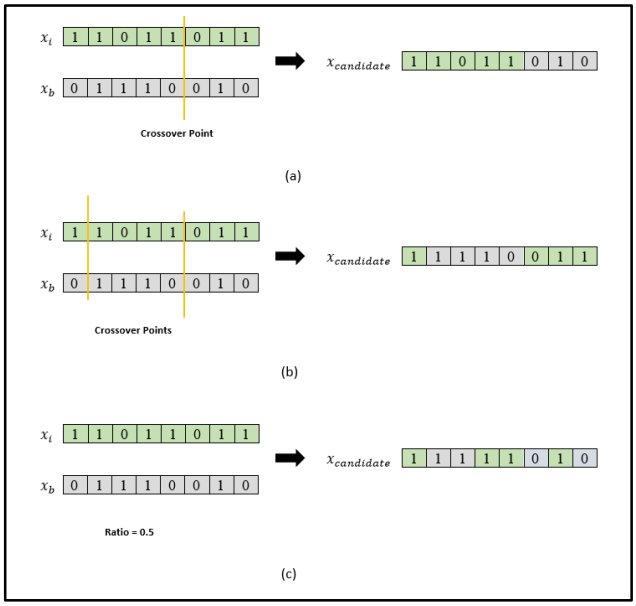


Figure 4. Crossover Strategies

The BinHBA algorithm proposed in this study was created using the transfer function and crossover strategy. In Figure 5, the BinHBA pseudocode is presented.

1. Determine parameters t_{max}, N, β,C
2. Initialize population of N individuals for binary search space (using "0" or "1")
3. Evaluate the fitness (f_i) of each honey badger position (X_i).
4. Find best position (X_P) and assign fitness to f_P.
5. **While** $t \leq t_{max}$ **do**
6. Utilize Equation 5 to update the decreasing factor α.
7. **For** i=1 **to** N **do**
8. Utilize Equation 2 to calculate the intensity I_i.
9. **If** $r < 0.5$ **then**
10. Use Equation 6 to update the position X_{new}.
11. **Else**
12. Use Equation 8 to update the position X_{new}.
13. **End if**
14. Apply the selected transfer function to X_{new}
15. Apply the determined crossover strategy to X_{new}
16. Examine new candidate position (X_{new}) and assign to f_{new}
17. **If** $f_{new} \leq f_i$ **then**
18. Assign $X_i = X_{new}$ and $f_i = f_{new}$
19. **End if**
20. **If** $f_{new} \leq f_P$ **then**
21. Assign $X_P = X_{new}$ and $f_P = f_{new}$
22. **End if**
23. **End for**
24. **End while**
25. Display X_P

Figure 5. BinHBA's Pseudocode

2.3. 0-1 Knapsack Problems

The 0-1 knapsack problem (KP) is an NP-hard optimization problem consisting of a knapsack and a set of items, each with its weight and profit, the aim of which is to place the items in the knapsack to maximize the total profit without exceeding the capacity of the knapsack. The following is a brief definition of the general 0-1 KP (Pisinger, 2005). Let n be the item with weights $w_1, w_2, \dots, w_n$ and profits $p_1, p_2, \dots, p_n$, and let C denote the maximum capacity of the knapsack. In such a case, the items have two states: the jth item is put in the knapsack ($x_j = 1$) and it is not put ($x_j = 0$). The mathematical formulation of 0-1 KP is given in Equation 12 (Li, Fang, and Zhu, 2023).

$$maximize \sum_{j=1}^{n} p_j x_j$$
$$subject\ to \quad \sum_{j=1}^{n} w_j x_j \leq C,\ x_j \epsilon \{0,1\},\ j = 1, \dots, n \quad (12)$$

3. Experimental Results

In this section, the performance of the proposed BinHBA algorithm is tested on small and large scale 0-1 KPs. The results were evaluated using statistical tests and compared with methods found in the literature. In this study, the KPs used to analyze the performance of BinHBA are considered two different datasets, as follows:
Dataset1: Includes fifteen small-scale 0–1 KP samples. The following link provides examples: https://pages.mtu.edu/~kreher/cages/Data.html. The dataset has between 16 and 24 items.
Dataset2: Includes twelve large-scale 0–1 KP samples. The following link provides examples: http://artemisa.unicauca.edu.co/~johnyortega/instances_01_KP. The dataset has between 100 and 1000 items.

Tables 2 and 3 include the tables for Datasets 1 and 2, respectively.

Table 2. The Small-scale Knapsack Problems Dataset

ID	Capacity	Size	Optimum
Kp1	3,780,355	16	7,850,983
Kp2	4,426,945	16	9,352,998
Kp3	4,323,280	16	9,151,147
Kp4	4,550,938	16	9,348,889
Kp5	3,760,429	16	7,769,117
Kp6	5,169,647	20	10,727,049
Kp7	4,681,373	20	9,818,261
Kp8	5,063,791	20	10,714,023
Kp9	4,286,641	20	8,929,156
Kp10	4,476,000	20	9,357,969
Kp11	6,404,180	24	13,549,094
Kp12	5,971,071	24	12,233,713
Kp13	5,870,470	24	12,448,780
Kp14	5,762,284	24	11,815,315
Kp15	6,654,569	24	13,940,099

Table 3. The Large-scale Knapsack Problems Dataset

Characteristic	ID	Capacity	Size	Optimum
Uncorrelated	**Kp16**	995	100	9147
	Kp17	1008	200	11,238
	Kp18	2543	500	28,857
	Kp19	5002	1000	54,503
Weakly correlated	**Kp20**	995	100	1514
	Kp21	1008	200	1634
	Kp22	2543	500	4566
	Kp23	5002	1000	9052
Strongly correlated	**Kp24**	997	100	2397
	Kp25	997	200	2697
	Kp26	2517	500	7117
	Kp27	4990	1000	14,390

The gathered results were compared using time and gap criteria, and the Wilcoxon signed-rank test (Woolson, 2007) was used for statistical evaluation. Equation 13 provides the formula for the gap, which establishes the percentage error between the optimum value and the obtained mean value.

$$Gap = \frac{Optimum\ value - obtained\ mean\ value}{Optimum\ value} * 100 \qquad (13)$$

In all KPs, the population size was determined as 20 and the maximum number of iterations was defined as 5000. Additionally, each problem was run independently 20 times. These parameter values were selected to align with those of the algorithms being compared to guarantee a fair literature comparison.

First, this section evaluated the results obtained by integrating eight different transfer functions, including constant S-shaped and time-varying S-shaped, into the HBA algorithm. The comparative results of the evaluations made according to gap and time are given in Tables 4 and 5, respectively. Based on the total gap values shown in Table 4, the smallest values were obtained from the S1 and S1(vt) transfer functions, respectively, and they were the most successful versions. Moreover, Table 5's total time values indicated that the versions using S4 and S2(vt) transfer functions execute all problems in the shortest time, respectively. In addition to the evaluation in Table 5, the run times are considered as two groups of constant and time-varying S-shaped transfer functions, and the total times obtained are given comparatively in Figure 6. According to Figure 6, time-varying transfer functions solved problems in less time than constant ones.

According to the results above, S1 was more successful in both constant and time-varying transfer functions. Although the versions using this transfer function lag behind other transfer functions in terms of run time, it can be said that the increase in time is tolerable and the difference between times is not too much. Upon analyzing the gap values, it was observed that the value 0 could not yet be obtained for all problems. Accordingly, it was decided that the algorithm needed further improvement. To achieve this, solution diversity was increased by applying a crossover between the candidate solution and the best individual in the population after the transfer function. The crossover strategy was applied to the most successful versions, namely BinHBA_S1 and BinHBA_S1(vt). As mentioned in the sections above, three crossover strategies—one point, two point, and uniform—were selected, and their contribution to performance was examined. The gap and time results of the versions obtained by separately integrating three different crossover strategies into BinHBA_S1 are given in Table 6. According to the table, the version in which the uniform crossover strategy was integrated obtained a gap value of 0 in all problems except KP10. Accordingly, it became the most successful version. When evaluated in terms of time, it was seen that the version in which the two-point crossover strategy was integrated found solutions faster in thirteen out of fifteen problems. A more detailed evaluation of the results is given in Figure 7. In the graphic, the total gap and run-time values of the versions obtained with different crossover strategies were compared. According to Figure 7, in terms of the total gap, the most successful crossover strategy was uniform, while the least successful strategy was two-point. In terms of total time, the crossover strategy that took the shortest time was two-point, while the strategy that required the longest time was one-point. Despite this, the total time values are quite close to each other and at a tolerable level.

The gap and time results of the versions obtained by separately integrating three different crossover strategies into BinHBA_S1(vt) are given in Table 7. According to the table, the version integrated with the uniform crossover strategy was able to obtain 0 gap values in all problems, and it became the most successful version. When evaluated in terms of time, it was seen that the version with an integrated two-point crossover strategy was faster in eleven out of fifteen problems. A more detailed evaluation of the results is given in Figure 8. According to Figure 8, in terms of total gap, the most successful crossover strategy was uniform, while the least successful strategy was two-point. In terms of total time, the crossover strategy that took the shortest time was two-point, while the strategy that required the longest time was uniform. Despite this, the total time values are quite close to each other and at a tolerable level.

Table 4. The Gap results of BinHBA for Transfer Functions on Small-scale KPs

Gap	S1	S2	S3	S4	S1 (tv)	S2 (tv)	S3 (tv)	S4 (tv)
Kp1	0	0	0	0	0	0	0	0
Kp2	0	0	0	0	0	0	0	0
Kp3	0	0	0	0	0	0	0	0
Kp4	0	0	0	0	0	0	0	0
Kp5	0	0	0	0	0	0	0	0
Kp6	0	0	0.087087	0.100851	0	0	0	0.038023
Kp7	0	0.021248	0.088065	0.198782	0	0.063744	0	0.063744
Kp8	0	0.001354	0.012924	0.018281	0	0.002709	0.012496	0.001354
Kp9	0	0.01541	0.068902	0.099723	0	0	0	0.01541
Kp10	0	0	0.00083	0.016758	0.00083	0	0.00904	0.002491
Kp11	0.059247	0.119325	0.283389	0.459879	0.155572	0.209475	0.15182	0.223807
Kp12	0.010593	0.117434	0.339589	0.374163	0.016356	0.031779	0.083946	0.262456
Kp13	0.010928	0.10126	0.28258	0.318272	0.136092	0.147541	0.089782	0.116231
Kp14	0.017821	0.191986	0.338926	0.279588	0.075097	0.09552	0.134309	0.216103
Kp15	0.0108	0.103534	0.177068	0.388096	0.02835	0.145135	0.084037	0.108163
Total	**0.109389**	0.671551	1.67936	2.254393	0.412297	0.695903	0.56543	1.047782
Rank	**1**	4	7	8	2	5	3	6

Table 5. The Time Results of BinHBA for Transfer Functions on Small-scale KPs

Time	S1	S2	S3	S4	S1 (tv)	S2 (tv)	S3 (tv)	S4 (tv)
Kp1	6.65069	5.892792	4.872831	4.507016	5.325669	4.439543	4.711883	5.009465
Kp2	6.629714	5.695539	4.969793	4.988711	5.406604	4.717382	4.504029	4.739923
Kp3	6.816617	5.546846	4.890418	4.650769	4.84883	4.345735	4.752857	4.765454
Kp4	6.473288	5.340374	5.351684	4.830218	5.092839	4.552444	4.545971	4.984927
Kp5	4.426726	5.125402	5.242635	4.663558	5.073189	3.367428	4.722582	4.650742
Kp6	6.769689	4.226241	5.458453	5.3677	5.85826	5.141575	4.849339	4.513609
Kp7	6.829454	5.92239	5.33276	5.237071	5.711099	4.980693	5.152497	5.227866
Kp8	5.370071	5.83809	5.747943	5.080702	6.837116	5.230434	3.731661	5.358518
Kp9	5.594199	5.479169	5.235293	3.617332	5.73581	4.990484	3.734037	5.08305
Kp10	4.867781	5.674898	5.373873	6.15694	5.507986	4.972808	5.199535	5.274221
Kp11	5.383256	6.051872	5.466676	3.781263	6.052793	5.397622	5.741501	5.141686
Kp12	7.892347	6.112406	5.473513	4.582532	6.19244	5.4342	5.995793	5.564137
Kp13	5.694487	6.549463	5.34437	5.647101	6.579839	4.273229	5.385772	5.932589
Kp14	6.157034	6.814908	5.487054	4.056518	4.467437	5.380495	4.934439	5.40607
Kp15	8.141763	5.789737	5.381136	3.767247	4.834168	5.440346	5.683747	5.451694
Total	93.69712	86.06012	79.62843	**70.93468**	83.52408	72.66442	73.64564	77.10395
Rank	8	7	5	**1**	6	2	3	4

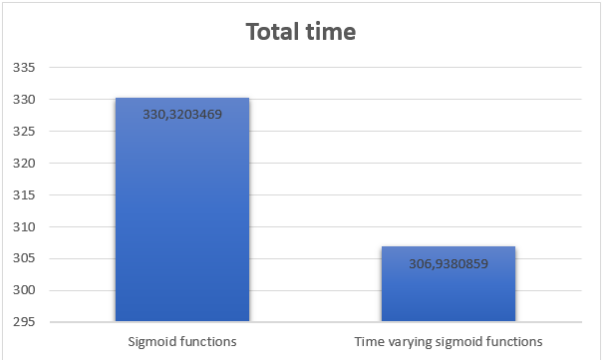


Figure 6. Total Times for S-shaped and Time-varying S-shaped

Table 6. Comparison Results of BinHBA_S1 With Crossover Strategy

	BinHBA_S1					
Crossover	**One-point**		**Two-point**		**Uniform**	
	Gap	**Time**	**Gap**	**Time**	**Gap**	**Time**
Kp1	0	4.38319	0.412793	**4.03508**	0	4.109804
Kp2	0.033756	4.242462	1.056551	**3.893201**	0	4.080178
Kp3	0.501881	4.175273	0.736159	**3.89766**	0	4.273908
Kp4	0.130475	4.781753	0.374374	4.113133	0	**4.076676**
Kp5	0	4.366115	0.680832	**3.775772**	0	4.078058
Kp6	0	4.415716	0.088375	**4.122253**	0	4.608072
Kp7	0.377662	4.837716	1.134329	**4.277449**	0	4.441237
Kp8	0.071377	4.678485	0.187971	**4.245522**	0	4.478014
Kp9	0	**4.12813**	0	4.179539	0	4.652959
Kp10	0	4.495505	0.122016	**4.293849**	0.00083	4.51974
Kp11	0.36519	4.729588	0.700267	**4.580302**	0	4.818838
Kp12	0.2942	4.489277	0.555171	**4.389614**	0	4.824257
Kp13	0.402994	4.891271	0.992989	**4.696395**	0	4.841823
Kp14	0.839502	5.297324	0.275875	**4.556759**	0	4.822083
Kp15	0.101273	4.712998	0.068385	**4.306862**	0	4.800877

Table 7. Comparison Results of BinHBA_S1(vt) With Crossover Strategy

	BinHBA_S1(vt)					
Crossover	**One-point**		**Two-point**		**Uniform**	
	Gap	**Time**	**Gap**	**Time**	**Gap**	**Time**
Kp1	0	3.790519	0.414203	**3.358378**	0	3.64086
Kp2	0.033756	3.773799	0.776256	**3.505204**	0	3.768489
Kp3	0.390352	3.609998	0.751007	**3.554214**	0	3.763693
Kp4	0.117428	4.038579	0.38316	**3.523191**	0	3.800585
Kp5	0	3.86185	0.680832	**3.175116**	0	3.657917
Kp6	0.061043	3.786714	0.079537	**3.583831**	0	4.115571
Kp7	0.188831	4.385487	0.446363	**3.637944**	0	4.060471
Kp8	0.060578	5.494723	0.163389	4.383684	0	**3.840681**
Kp9	0	4.661513	0.039671	4.290042	0	**3.919826**
Kp10	0	4.743587	0.140522	4.344328	0	**4.19987**
Kp11	0.47381	5.57457	0.490837	4.659348	0	**4.20154**
Kp12	0.343233	4.892168	0.566566	**4.09707**	0	6.714814
Kp13	0.301536	4.421041	0.831091	**4.10336**	0	4.204954
Kp14	0.222046	4.381411	1.005966	**4.309546**	0	5.829603
Kp15	0.121527	4.196752	0.068124	**3.916412**	0	5.96857

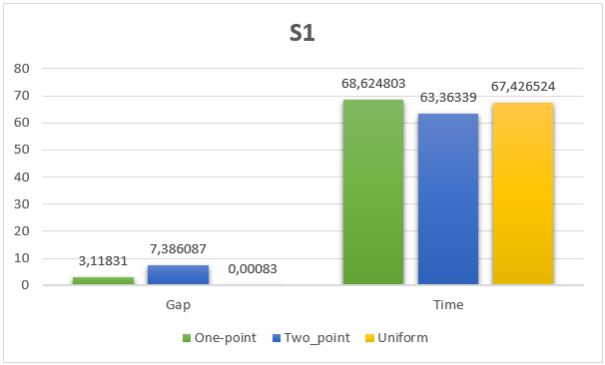


Figure 7. Total Gap and Time Graphics for BinHBA_S1 Version

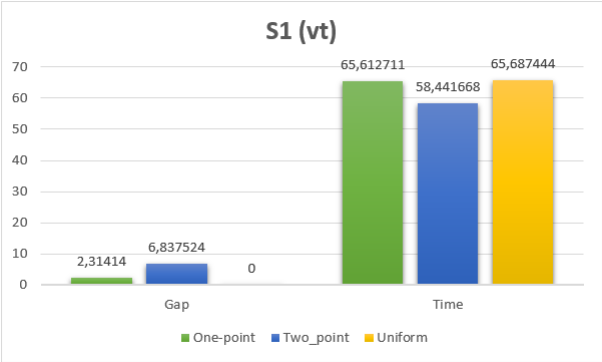


Figure 8. Total Gap and Time graphics for BinHBA_S1(vt) Version

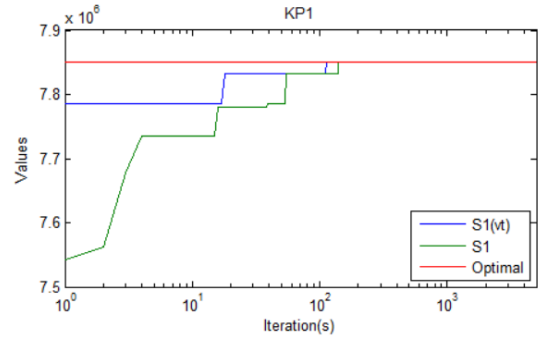
Figure 9 shows the comparative convergence graphics of the best versions of BinHBA_S1 and BinHBA_S1(vt) algorithms using the uniform crossover strategy in small-scale KPs. Graphics are drawn according to the best value obtained. When the graphics were examined, it was determined that BinHBA_S1(vt) converged to the optimal value faster in KP1, KP3, KP4, KP10, KP11, KP12, KP13, and KP15. In six of the remaining problems, BinHBA_S1 converged to the optimal faster, while in KP8, the two algorithms converged to the optimal in similar iterations.

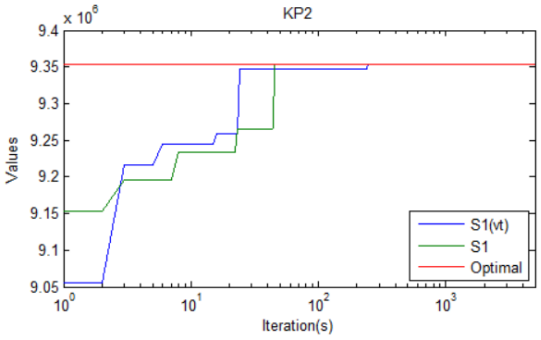
The results obtained for the best BinHBA_S1 and BinHBA_S1(vt) versions were evaluated with the help of statistical testing. Wilcoxon signed rank test (García, Molina, Lozano, and Herrera, 2009) was used to determine whether the algorithms had a significant performance difference at the 5% confidence level when compared to each other and algorithms in other literature. In this study, the Wilcoxon signed-rank test was applied to gap values. In Table 8, the p and h values obtained as a result of the pairwise Wilcoxon signed rank test for the BinHBA_S1(vt) and BinHBA_S1 algorithms were presented. An h value of 0 in the table indicates that the results of the two algorithms are similar (p-value >0.05), in contrast to an h value of 1 in the table, which indicates that there is a significant difference between the two algorithms and that their results are not similar (p-value <0.05). The results in Table 8 demonstrated that the algorithms performed similarly in all problems, with no significant differences between them. Because both algorithms found the optimum values in other problems except KP10. However, it can be concluded that the BinHBA_S1(vt) algorithm is more successful due to reasons such as finding the optimum values in all problems, converging faster in most of the problems, and taking a shorter total time.

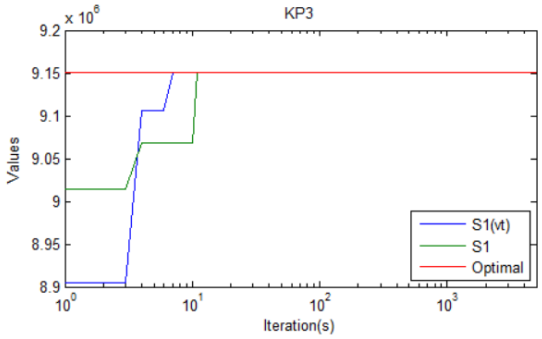
BinHBA_S1 and BinHBA_S1(vt) algorithms were also compared with algorithms in the literature. For this purpose, Binary Aquila Optimizer with Crossover and Mutation methods (BAO-CM) (Baş, 2023), Binary Artificial Jellyfish Search algorithm (Bin_AJS) (Yildizdan and Baş, 2023), Binary Equilibrium Optimization Algorithm with V3-shaped transfer function (BEOV3) (Abdel-Basset, Mohamed, and Mirjalili, 2021), Binary Slime Mould Algorithm (BSMA) (Abdollahzadeh, Barshandeh, Javadi, and Epicoco, 2022), and Enhanced Binary Coati Optimization Algorithm (EBinCOA) (Yildizdan and Bas, 2024) algorithms, which were run under the same conditions, were selected from the literature. Based on Table 9, which presents the comparison results according to the gap values, the Bin_AJS, EBinCOA, and BinHBA_S1(vt) algorithms, which found the value 0 in all problems, were the most successful and ranked first in the ranking. According to this result, it was revealed that the proposed time-varying BinHBA version was a successful algorithm that competes with the algorithms in the literature. Table 10 shows the pairwise Wilcoxon-signed rank test results of the proposed BinHBA_S1(vt) with other algorithms. The '+', '-', and '=' lines indicate how many problems BinHBA_S1(vt) found better, worse, and equal gap values than the other algorithm, respectively. After analyzing the results, it was found that there was no significant difference between the algorithms' performance in most problems.

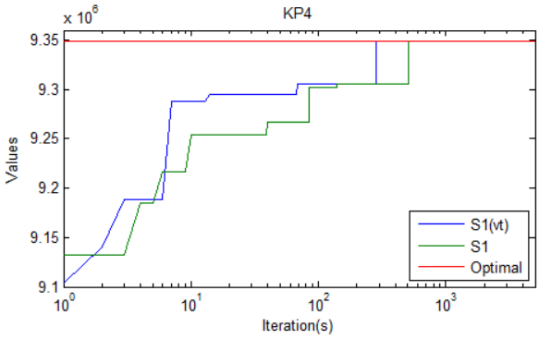
Table 8. The Results of the Wilcoxon Signed-rank Tests for BinHBA_S1 and BinHBA_S1(vt) on Small-scale KPs

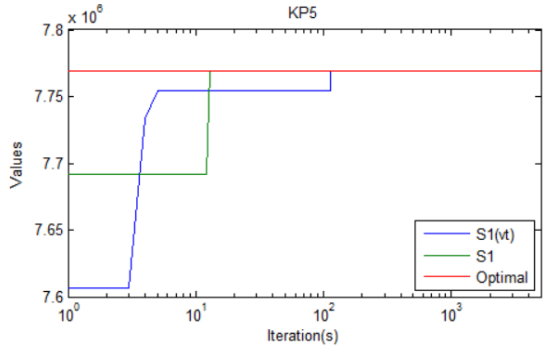
ID	BinHBA_S1(vt) – BinHBA_S1	
	p-value	h-value
Kp1	1.0000	0
Kp2	1.0000	0
Kp3	1.0000	0
Kp4	1.0000	0
Kp5	1.0000	0
Kp6	1.0000	0
Kp7	1.0000	0
Kp8	1.0000	0
Kp9	1.0000	0
Kp10	0.3173	0
Kp11	1.0000	0
Kp12	1.0000	0
Kp13	1.0000	0
Kp14	1.0000	0
Kp15	1.0000	0

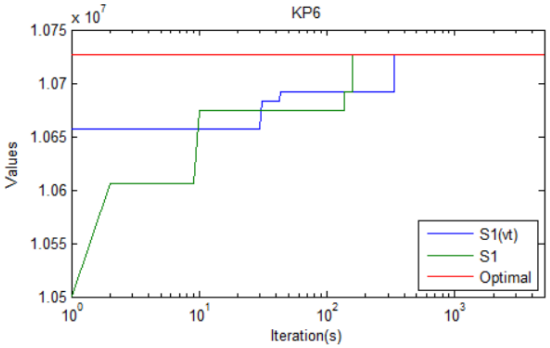

KP1
x 10^6
Values
Iteration(s)
S1(vt)
S1
Optimal

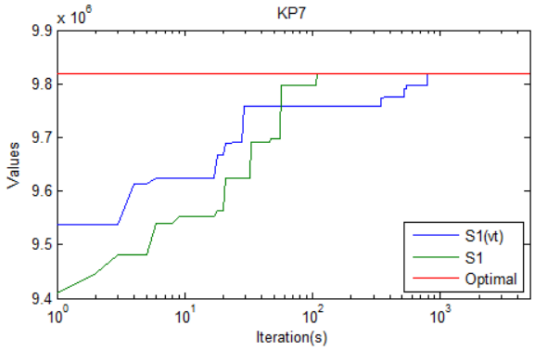

KP2
x 10^6
Values
Iteration(s)
S1(vt)
S1
Optimal

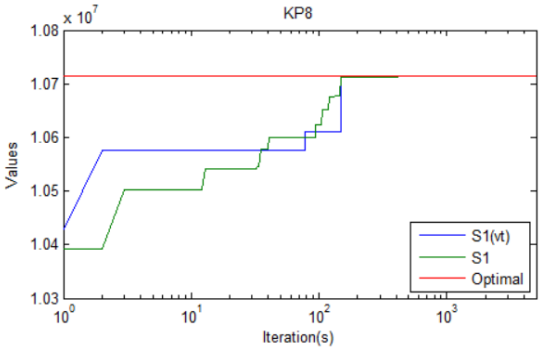

KP3
x 10^6
Values
Iteration(s)
S1(vt)
S1
Optimal


KP4
x 10^6
Values
Iteration(s)
S1(vt)
S1
Optimal


KP5
x 10^6
Values
Iteration(s)
S1(vt)
S1
Optimal


KP6
x 10^7
Values
Iteration(s)
S1(vt)
S1
Optimal


KP7
x 10^6
Values
Iteration(s)
S1(vt)
S1
Optimal


KP8
x 10^7
Values
Iteration(s)
S1(vt)
S1
Optimal

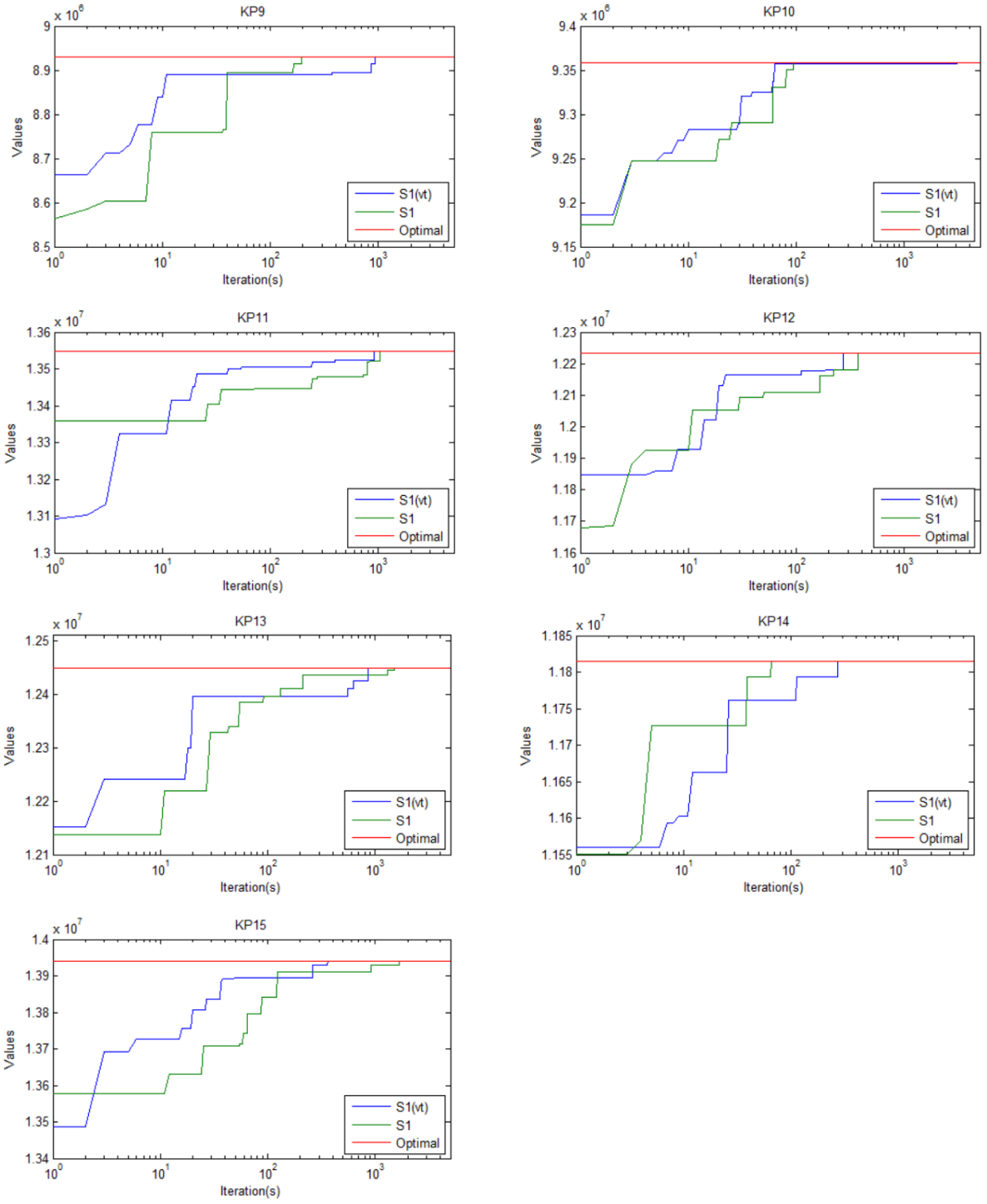


Figure 9. Convergence Graphics for BinHBA_S1 and BinHBA_S1 (vt) Versions

Table 9. Comparison of the Gap results of BinHBA_S1 and BinHBA_S1(vt) With Algorithms in the Literature

	BAO-CM	Bin_AJS	BEOV3	BSMA	EBinCOA	BinHBA_S1	BinHBA_S1(vt)
Kp1	0	0	0	0	0	0	0
Kp2	0	0	0	0	0	0	0
Kp3	0	0	0	0	0	0	0
Kp4	0	0	0	0	0	0	0
Kp5	0	0	0	0	0	0	0
Kp6	0	0	0	0	0	0	0
Kp7	0	0	0	0	0	0	0
Kp8	0	0	0	0	0	0	0
Kp9	0	0	0	0	0	0	0
Kp10	0	0	0	0	0	0.00083	0
Kp11	0	0	0	0	0	0	0
Kp12	0	0	0	0	0	0	0
Kp13	0	0	0	0	0	0	0
Kp14	0.0017	0	0.0036	0.004	0	0	0
Kp15	0	0	0	0	0	0	0
Total	0.0017	0	0.0036	0.004	0	0.00083	0
Rank	3	1	4	5	1	2	1

Table 10. The pairwise Wilcoxon signed-rank Test Results of BinHBA_S1(vt) and Other Algorithms

	BinHBA_S1(vt) BAO-CM	BinHBA_S1(vt) Bin_AJS	BinHBA_S1(vt) BEOV3	BinHBA_S1(vt) BSMA	BinHBA_S1(vt) EBinCOA	BinHBA_S1(vt) BinHBA_S1
+	1	0	1	1	0	1
-	0	0	0	0	0	0
=	14	15	14	14	15	14
p-value	0.317	1.000	0.317	0.317	1.000	0.317
h-value	0	0	0	0	0	0

Secondly, in this section, the performance of the proposed algorithms was tested on the large-scale KPs listed in Table 3. In Table 11, the gap and time values obtained as a result of these tests are presented for BinHBA_S1 and BinHBA_S1(vt). According to Table 11, BinHBA_S1(vt) was more successful by obtaining smaller gap values in eight of the twelve problems. In terms of time, BinHBA_S1 solved six problems more quickly, whereas BinHBA_S1(vt) resolved the other six in less time. The total gap and time graphics obtained according to the results in Table 11 are given in Figures 10 and 11. According to the graphic in Figure 10, the total gap value obtained for BinHBA_S1(vt) was smaller. According to the time graphic in Figure 11, BinHBA_S1(vt) completed the problems in a shorter time. Accordingly, in large-scale KPs, BinHBA_S1(vt) was more successful by obtaining smaller gap values in a shorter time. The proposed algorithms and the algorithms found in the literature were compared as well. For this purpose, Binary Evolutionary Mating Algorithm (BinEMA) (Yildizdan and Bas, 2024), Binary Fire Hawk Optimizer (BinFHO) (Yildizdan and Bas, 2024), and Binary Mountain Gazelle Optimizer (BinMGO) (Yildizdan and Bas, 2024) algorithms, which were run under the same conditions, were chosen. Comparison results are given in Table 12. In this comparison made according to gap values, the smallest values were mostly obtained from BinMGO. In the ranking, BinMGO was in first place, while BinHBA_S1(vt) was in second place. Based on the results of the pairwise Wilcoxon signed-rank test provided in Table 13, BinHBA_S1(vt) showed better performance by obtaining better gap values than BinEMA and BinFHO in all problems. When evaluated by p-values, it was seen that there was a significant difference between them. BinHBA_S1(vt) found better gap values than BinHBA_S1 in eight problems and showed better performance. There was a significant difference between the algorithms. On the other hand, BinMGO found a better gap value in ten of the problems and performed better than BinHBA_S1(vt). There was also a significant difference between the algorithms.

When all results are evaluated in general, the proposed BinHBA_S1 and BinHBA_S1(vt) algorithms achieve successful results in a reasonable time. BinHBA_S1(vt) achieved more successful results compared to BinHBA_S1 in both small and large-scale KPs. In comparisons made with the literature, BinHBA_S1(vt) generally showed better or similar performances than the compared algorithms. As encountered in most metaheuristics, the proposed BinHBA versions were also affected by the dimension increase. Despite this, BinHBA_S1(vt) ranked second in the literature comparison. This result revealed that new modifications may be needed to address the performance decrease caused by the dimension increase.

Table 11. Comparison Results of BinHBA_S1 and BinHBA_S1(vt) on Large-scale KPs

	Gap		Time	
ID	**BinHBA_S1**	**BinHBA_S1(vt)**	**BinHBA_S1**	**BinHBA_S1(vt)**
Kp16	**0.90**	1.59	361.02	**360.56**
Kp17	**1.67**	2.22	1221.15	**982.88**
Kp18	6.77	**5.17**	7142.54	**2966.64**
Kp19	11.94	**7.68**	**15539.14**	17127.80
Kp20	3.15	**1.54**	**175.38**	218.02
Kp21	14.89	**12.28**	**520.28**	1440.49
Kp22	22.47	**12.64**	**3144.10**	4588.00
Kp23	23.75	**20.08**	17941.32	**14898.59**
Kp24	**4.23**	4.27	239.53	**170.82**
Kp25	**8.69**	9.89	**523.32**	1003.96
Kp26	13.59	**9.84**	**3491.35**	6947.49
Kp27	18.08	**14.45**	17215.86	**12804.83**

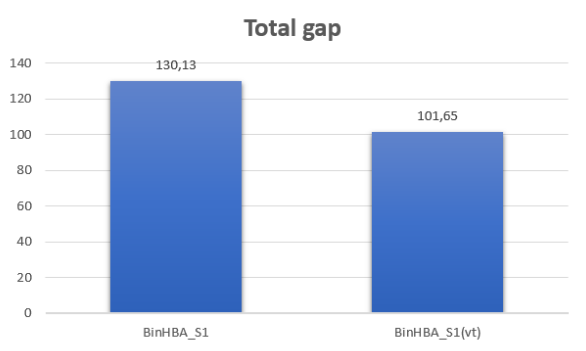


Figure 10. Total Gap for BinHBA_S1 and BinHBA_S1(vt) Versions

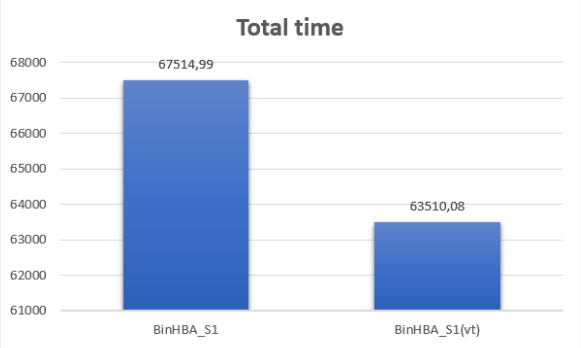


Figure 11. Total Time for BinHBA_S1 and BinHBA_S1(vt) Versions

Table 12. Comparison of the Gap results of BinHBA_S1 and BinHBA_S1(vt) With Algorithms in the Literature

ID	BinEMA	BinFHO	BinMGO	BinHBA_S1	BinHBA_S1(vt)
Kp16	2.79	2.98	**0.00**	0.90	1.59
Kp17	5.18	20.03	2.29	**1.67**	2.22
Kp18	13.59	38.95	**2.94**	6.77	5.17
Kp19	26.09	49.80	**5.88**	11.94	7.68
Kp20	6.26	15.85	**1.47**	3.15	1.54
Kp21	19.47	19.23	**11.13**	14.89	12.28
Kp22	24.59	33.53	13.54	22.47	**12.64**
Kp23	29.50	37.04	**14.62**	23.75	20.08
Kp24	7.63	10.96	**3.40**	4.23	4.27
Kp25	11.49	24.90	**5.24**	8.69	9.89
Kp26	18.62	36.31	**8.74**	13.59	9.84
Kp27	25.48	40.88	**12.51**	18.08	14.45
Total	190.70	330.46	**81.75**	130.13	101.65
Rank	4	5	1	3	2

Table 13. The Pairwise Wilcoxon Signed-rank Test Results of BinHBA_S1(vt) and Other Algorithms

	BinHBA_S1(vt) BinEMA	**BinHBA_S1(vt) BinFHO**	**BinHBA_S1(vt) BinMGO**	**BinHBA_S1(vt) BinHBA_S1**
+	12	12	2	8
-	0	0	10	4
=	0	0	0	0
p-value	0.002	0.002	0.009	0.023
h-value	1	1	1	1

4. Conclusion and future works

The Honey Badger Algorithm (HBA) is a metaheuristic algorithm proposed in 2022, inspired by the honey badger's foraging behaviors such as sniffing, digging, and following the honeyguide bird. In this study, a new binary version of the Honey Badger algorithm (BinHBA) was proposed. Transfer functions were used for the binarization of HBA, which was proposed for continuous optimization problems. S-shaped transfer functions, which are frequently used in the literature, were chosen as the transfer function. At the same time, time-varying S-shaped transfer functions were used for the HBA in this study to provide a better balance between exploration and exploitation. Binary HBA versions created with eight different transfer functions obtained this way were first tested on small-scale KPs. After this test, it was determined that the most successful transfer functions were S1 and time-varying S1. The results obtained revealed that using only the transfer function was insufficient in terms of performance. That's why the crossover strategy that supports diversity was integrated into the algorithm. The performance was examined by applying three different crossover strategies: one-point, two-point, and uniform. It was seen that the strategy that contributed the most was uniform. The BinHBA algorithm, which was finalized with the time-varying S1 transfer function and uniform crossover strategy, found the optimum value for all problems in small-scale KPs. It has reached the performance of successful algorithms in the literature. The algorithm was secondary tested on large-scale KPs. The results obtained from the test revealed that the version of BinHBA using the time-varying S1 transfer function is more successful. In literature comparisons, the BinMGO algorithm ranked first, while the BinHBA_S1(vt) ranked second. The BinHBA_S1(vt) performed better than all the algorithms except BinMGO, and there was a statistically significant difference between them.

Most algorithms in the literature find values that are very close to the optimum or optimal in small-scale KPs. Therefore, the performance differences between algorithms are revealed mostly by the tests performed in large-scale KPs. In this context, when the BinHBA_S1(vt) results are examined, the success of the algorithm in large-scale KPs compared to the literature proves that the algorithm is preferable and effective. In light of all these results, it can be said that the proposed algorithm is a promising algorithm for different binary optimization problems.

Future research can apply the BinHBA to a variety of binary optimization problems, including uncapacitated facility location and set-union knapsack problems.

Conflict of Interest

No conflict of interest was declared by the author.

References

Abasi, A. K., Aloqaily, M., and Guizani, M. (2023). Optimization of CNN using modified Honey Badger Algorithm for Sleep Apnea detection. Expert Systems with Applications, 229, 120484. doi: https://doi.org/10.1016/j.eswa.2023.120484

Abdel-Basset, M., Mohamed, R., and Mirjalili, S. (2021). A Binary Equilibrium Optimization Algorithm for 0–1 Knapsack Problems. Computers & Industrial Engineering, 151, 106946. doi: https://doi.org/10.1016/j.cie.2020.106946

Abdollahzadeh, B., Barshandeh, S., Javadi, H., and Epicoco, N. (2022). An enhanced binary slime mould algorithm for solving the 0–1 knapsack problem. Engineering with Computers, 1-22.

Al-Betar, M. A., Hammouri, A. I., Awadallah, M. A., and Abu Doush, I. (2021). Binary β-hill climbing optimizer with S-shape transfer function for feature selection. Journal of Ambient Intelligence and Humanized Computing, 12(7), 7637-7665.

Aljebreen, M., Alabduallah, B., Mahgoub, H., Allafi, R., Hamza, M. A., Ibrahim, S. S., . . . Alsaid, M. I. (2023). Integrating IoT and honey badger algorithm based ensemble learning for accurate vehicle detection and classification. Ain Shams Engineering Journal, 14(11), 102547. doi: https://doi.org/10.1016/j.asej.2023.102547

Altuwairiqi, M. (2024). An optimized multi-hop routing protocol for wireless sensor network using improved honey badger optimization algorithm for efficient and secure QoS. Computer Communications, 214, 244-259. doi: https://doi.org/10.1016/j.comcom.2023.08.011

Awadallah, M. A., Hammouri, A. I., Al-Betar, M. A., Braik, M. S., and Elaziz, M. A. (2022). Binary Horse herd optimization algorithm with crossover operators for feature selection. Computers in Biology and Medicine, 141, 105152. doi: https://doi.org/10.1016/j.compbiomed.2021.105152

Baş, E. (2023). Binary aquila optimizer for 0–1 knapsack problems. Engineering Applications of Artificial Intelligence, 118, 105592. doi: https://doi.org/10.1016/j.engappai.2022.105592

Büyüköz, G. O., and Haklı, H. (2023). Binary Honey Badger Algorithm for 0-1 Knapsack Problem. Journal of Intelligent Systems: Theory and Applications, 6(2), 108-118. doi: https://doi.org/10.38016/jista.1200225

Feda, A. K., Adegboye, M., Adegboye, O. R., Agyekum, E. B., Mbasso, W. F., and Kamel, S. (2024). S-shaped grey wolf optimizer-based FOX algorithm for feature selection. Heliyon, 10(2).

García, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. Journal of Heuristics, 15, 617-644.

Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S., and Al-Atabany, W. (2022). Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. Mathematics and Computers in Simulation, 192, 84-110. doi: https://doi.org/10.1016/j.matcom.2021.08.013

He, Y., Zhang, F., Mirjalili, S., and Zhang, T. (2022). Novel binary differential evolution algorithm based on Taper-shaped transfer functions for binary optimization problems. Swarm and evolutionary computation, 69, 101022. doi: https://doi.org/10.1016/j.swevo.2021.101022

Hu, G., Zhong, J., and Wei, G. (2023). SaCHBA_PDN: Modified honey badger algorithm with multi-strategy for UAV path planning. Expert Systems with Applications, 223, 119941. doi: https://doi.org/10.1016/j.eswa.2023.119941

Huang, P., Zhou, Y., Deng, W., Zhao, H., Luo, Q., and Wei, Y. (2024). Orthogonal opposition-based learning honey badger algorithm with differential evolution for global optimization and engineering design problems. Alexandria Engineering Journal, 91, 348-367. doi: https://doi.org/10.1016/j.aej.2024.02.024

Jain, D. K., Ding, W., and Kotecha, K. (2023). Training fuzzy deep neural network with honey badger algorithm for intrusion detection in cloud environment. International Journal of Machine Learning and Cybernetics, 14(6), 2221-2237. doi:10.1007/s13042-022-01758-6

Jin, C., Li, S., Zhang, L., and Zhang, D. (2023). The Improvement of the Honey Badger Algorithm and Its Application in the Location Problem of Logistics Centers. Applied Sciences, 13(11), 6805. doi: https://doi.org/10.3390/app13116805

Kapner, D. J., Cook, T. S., Adelberger, E. G., Gundlach, J. H., Heckel, B. R., Hoyle, C., and Swanson, H. E. (2007). Tests of the gravitational inverse-square law below the dark-energy length scale. Physical review letters, 98(2), 021101. doi: https://doi.org/10.1103/PhysRevLett.98.021101

Karakoyun, M., and Ozkis, A. (2022). A binary tree seed algorithm with selection-based local search mechanism for huge-sized optimization problems. Applied Soft Computing, 129, 109590. doi: https://doi.org/10.1016/j.asoc.2022.109590

Li, X., Fang, W., and Zhu, S. (2023). An improved binary quantum-behaved particle swarm optimization algorithm for knapsack problems. Information Sciences, 648, 119529. doi: https://doi.org/10.1016/j.ins.2023.119529

Mafarja, M., Aljarah, I., Heidari, A. A., Faris, H., Fournier-Viger, P., Li, X., and Mirjalili, S. (2018). Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowledge-Based Systems, 161, 185-204. doi: https://doi.org/10.1016/j.knosys.2018.08.003

Majumdar, P., Mitra, S., and Bhattacharya, D. (2023). Honey Badger algorithm using lens opposition based learning and local search algorithm. Evolving Systems, 1-26. doi: https://doi.org/10.1007/s12530-023-09495-z

Ni, B., Wang, Y., Huang, J., and Li, G. (2022). Hybrid Enhanced Binary Honey Badger Algorithm with Quadratic Programming for Cardinality Constrained Portfolio Optimization. International Journal of Foundations of Computer Science, 33(06n07), 787-803. doi: https://doi.org/10.1142/S0129054122420151

Pisinger, D. (2005). Where are the hard knapsack problems? Computers & Operations Research, 32(9), 2271-2284. doi: https://doi.org/10.1016/j.cor.2004.03.002

Qasem, S. N. (2024). A novel honey badger algorithm with multilayer perceptron for predicting COVID-19 time series data. The Journal of Supercomputing, 80(3), 3943-3969. doi: https://doi.org/10.1007/s11227-023-05560-1

Wang, B., Kang, H., Sun, G., and Li, J. (2024). Efficient traffic-based IoT device identification using a feature selection approach with Lévy flight-based sine chaotic sub-swarm binary honey badger algorithm. Applied Soft Computing, 155, 111455. doi: https://doi.org/10.1016/j.asoc.2024.111455

Woolson, R. F. (2007). Wilcoxon signed-rank test. Wiley encyclopedia of clinical trials, 1-3.

Yasear, S. A., and Ghanimi, H. M. (2022). A modified honey badger algorithm for solving optimal power flow optimization problem. International Journal of Intelligent Engineering and Systems, 15(4), 142-155.

Yildizdan, G., and Bas, E. (2024). A new binary coati optimization algorithm for binary optimization problems. Neural Computing and Applications, 36(6), 2797-2834.

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