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 415-424

ICBAST 2026: International Conference on Basic Sciences and Technology

Solution of the Binary Knapsack Problem Using the Secretary Bird Optimization Algorithm

Vahit Tongur
Konya Technical University

Aysen Kucukyaglioglu
Konya Technical University

Abstract: Many nature-inspired metaheuristic algorithms proposed in the literature are initially designed to operate in continuous search spaces. However, real-world problems are not limited to continuous domains; they also include discrete, combinatorial, and binary problem types.In this study, the performance of the Secretary Bird Optimization Algorithm (SBOA), a recently introduced metaheuristic method, is investigated on the knapsack problem, which belongs to the class of binary optimization problems. Numerous transfer functions have been proposed in the literature to transform continuous search spaces into binary ones. In this work, S-shaped transfer functions are employed for this purpose.Through this transformation, SBOA—originally designed for continuous domains—is adapted to the binary search space and referred to as the Binary SBOA (BSBOA). The performance of BSBOA is evaluated on well-known benchmark knapsack problem instances. While BSBOA achieves optimal solutions on 10 small-scale benchmark problems, it produces near-optimal results for 18 large-scale binary knapsack instances.This study aims to address the SBOA binary optimization problem and evaluate its performance in such an environment.

Keywords: Knapsack problem, Secretary bird optimization algorithm, Transfer functions

Introduction

The knapsack problem represents a classical discrete optimization framework frequently used to model various real-life decision-making scenarios. The goal is to identify a subset of items that yields the best possible objective value without violating predefined constraints. There are both single-dimensional and multi-dimensional variants of this problem. In the single-dimensional case, there is a single knapsack with a capacity limit into which the items are placed. In contrast, multi-dimensional variants may involve multiple knapsacks or multiple constraints associated with a single knapsack. In addition, some variants allow items to be selected more than once and included in the knapsack.

In this study, the single-dimensional version is considered. Each item is assumed to have a value and a weight, and the problem can be defined as selecting items to be placed into a single knapsack with a limited total weight capacity. Each item is associated with a weight (w) and a profit (p), which represents its level of utility. Let (C) denote the total weight capacity of the knapsack. In this case, the objective function of the problem can be formulated as given in Equation (1).

$$P_{max} = \sum_{i=1}^{n} p_i x_i \tag{1}$$

Subject to;

$$\sum_{i=1}^{n} w_i x_i \leq C \tag{2}$$

Here, n denotes the number of items, p_i represents the profit of the *i-th* item, w_i denotes the weight of the *i-th* item, and x_i indicates whether the *i-th* item is included in the knapsack. If the *i-th* item is selected, then $x_i = 1$; otherwise, it takes the value 0. Accordingly, the problem considered in this study is referred to as the binary knapsack problem. Although the problem may appear simple, its complexity increases exponentially as the problem size grows. For this reason, it is classified as an NP-hard problem (Feng et al., 2015). For this reason, metaheuristic methods are more commonly preferred than exact methods, particularly for large-scale instances of the knapsack problem. The binary knapsack problem has been widely addressed in literature through various optimization techniques.

Wang et al. proposed an effective method for solving the multiple knapsack problem (MKP) by employing a discrete variant of differential evolution. In their approach, the MKP was formulated as an integer programming model compatible with discrete evolutionary strategies. Furthermore, a novel discretization mechanism was introduced, enabling the transformation of a continuous algorithm into a discrete one through the combined use of transfer functions and modular operations. To handle infeasible solutions, the GROA algorithm was incorporated. The proposed framework, which integrates multiple types of transfer functions such as S, V, U, and taper-shaped functions, was tested on 30 benchmark datasets (Wang et al., 2024).

Feng and Wang adapted the moth search algorithm to the binary domain and introduced a self-learning-based binary moth search method. This approach enhances population diversity and strengthens global search capability. A key component of the method is the self-learning straight-flight operator, which allows individuals to learn not only from the global best solution but also from better-performing individuals within the population. The performance of the proposed method was evaluated using 89 benchmark problems, and the impact of transfer functions on the algorithm's effectiveness was also analyzed. (Feng & Wang, 2022).

Li et al. proposed a binary version of the quantum-behaved particle swarm optimization algorithm to solve knapsack problems. Although the original method is known for its simplicity and strong global search ability, it suffers from premature convergence caused by particle clustering. To overcome this limitation, an improved variant was developed. In this enhanced version, a mapping mechanism based on the mean best position was introduced to address discretization challenges. Additionally, transfer functions were used to convert local attractors into binary representations, and crossover operations were applied to guide the search process. A repair mechanism and a diversity preservation strategy were also incorporated to prevent convergence to local optima and improve overall performance (Li et al., 2023).

Ali et al. introduced a modified differential evolution algorithm tailored for binary optimization problems. Their method enhances the classical differential evolution approach by incorporating new components such as solution encoding schemes, mapping strategies, and mechanisms to maintain population diversity. Moreover, an efficient fitness evaluation procedure was designed to both assess and repair candidate solutions. The effectiveness of the proposed method was validated on four datasets containing a total of 44 binary knapsack instances (Ali et al., 2021).

Abdel-Basset et al. developed a binary version of the slime mould algorithm for solving multidimensional knapsack problems. To enable the algorithm to operate in a binary search space, different categories of transfer functions, including S-shaped, V-shaped, and U-shaped functions, were investigated. However, the algorithm was observed to suffer from premature convergence. To address this issue, two improvement strategies were introduced. The first strategy involves modifying the current best solution by replacing an unselected item and evaluating the resulting solution. The second strategy focuses on restarting the population after a predefined number of iterations. These enhancements led to the development of an improved version of the algorithm. In addition, a repair mechanism was implemented to handle infeasible solutions (Abdel-Basset et al., 2021).

Gupta et al. proposed an improved sine–cosine algorithm by integrating it with differential evolution. In this hybrid approach, the search process dynamically switches between the modified sine–cosine mechanism and differential evolution based on the evolutionary state of the population. While the modified sine–cosine component improves exploitation and accelerates convergence, differential evolution contributes to maintaining population diversity and avoiding local optima. The parameters of the algorithm are adaptively controlled to achieve a balance between exploration and exploitation. A binary version of this method was also developed and applied to multidimensional knapsack problems, demonstrating competitive performance (Gupta et al., 2022).

Granmo et al. explored the application of the knapsack problem in real-world scenarios, particularly in the context of web resource allocation, and reported successful outcomes (Granmo et al., 2007).

Kong et al. introduced a binary and colony optimization approach for solving the multidimensional knapsack problem. Their method employs a pheromone update mechanism specifically designed for binary representations and allows the generation of infeasible solutions during the construction phase. To address this issue, a problem-specific repair operator was used to correct infeasible solutions at each iteration. Additionally, strategies such as pheromone reinitialization and alternative intensification mechanisms were implemented to reduce the risk of premature convergence (Kong et al., 2008). Tongur and Ülker adapted the migrating birds optimization algorithm to the knapsack problem by modifying its neighborhood structure to suit discrete search spaces. Their approach demonstrated promising performance on benchmark problems (Tongur & Ülker, 2017).

A general observation from the literature is that algorithms originally designed for continuous search spaces are commonly adapted to binary domains through the use of transfer functions. However, the effectiveness of these functions varies depending on the algorithm, yielding different levels of performance across studies. In this study, the performance of the SBOA, a recently proposed algorithm designed for continuous search spaces, is investigated in a binary setting by employing an S-shaped transfer function. The evaluation is conducted on the single-dimensional binary knapsack problem.

Method

Secretary Bird Optimization Algorithm (SBOA)

The Secretary Bird Optimization Algorithm (SBOA) is a recently developed metaheuristic approach inspired by the hunting behavior and survival strategies of secretary birds (Fu et al., 2024). The algorithm models both the preparation and hunting processes of the bird, mapping these natural behaviors into exploration and exploitation mechanisms within the optimization framework. The initial preparation phase of the bird corresponds to the initialization stage of the algorithm, while the subsequent hunting process is divided into multiple exploration phases. In contrast, the bird's predator avoidance strategies are associated with the exploitation phase of the algorithm.

Initial Phase

SBOA is a population-based optimization technique in which each individual represents a candidate solution within the search space. The position of each individual corresponds to the values of the decision variables of the problem. At the beginning of the algorithm, the positions of individuals are randomly generated according to Equation (3). Following initialization, the entire population is represented as given in Equation (4), where each row corresponds to a candidate solution.

$$X_{i,j} = lwb_j + r \times \left(upb_j - lwb_j\right), i = 1,2,\ldots,N, j = 1,2,\ldots,Dim \tag{3}$$

where, $X_{i,j}$ denotes the position of the bird, while lwb_j and upb_j represent the lower and upper bounds of the search space, respectively. The parameter r is a random number in the interval [0,1].

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} & \cdots & x_{1,Dim} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,j} & \cdots & x_{2,Dim} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & x_{i,2} & \cdots & x_{i,j} & \cdots & x_{i,Dim} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,j} & \cdots & x_{N,Dim} \end{bmatrix}_{N \times Dim} \tag{4}$$

where, X represents the entire population, while X_i denotes the *i-th* individual in the population. The term $X_{i,j}$ corresponds to the value of the *j-th* decision variable of the *i-th* individual. N, denotes the population size, and Dim represents the problem dimension.

Each individual in the population is evaluated using the objective function, and the corresponding fitness values are stored in a vector as described in Equation (5). These values are used to guide the search process throughout the optimization.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1} \tag{5}$$

Where, F represents the objective function vector, while F_i denotes the objective function value of the *i-th* individual.

The SBOA updates individuals in the population based on two fundamental behavioral patterns observed in secretary birds:

- Hunting behavior
- Escape behavior

During each iteration, all individuals are updated according to these two mechanisms.

Hunting-Based Exploration Mechanism

The hunting behavior of secretary birds can be characterized by a three-stage process: detecting the prey, exhausting it, and finally attacking. In SBOA, these stages are modeled as three distinct phases of exploration. These stages are presented in Figure 1.

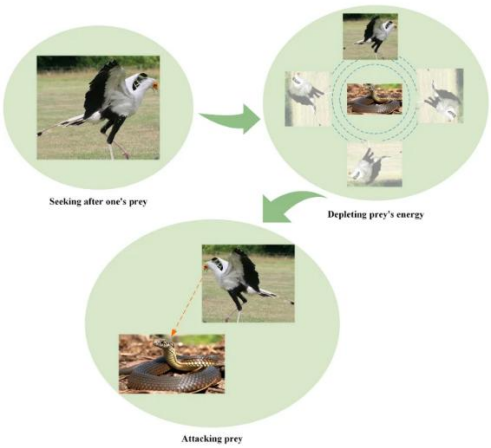

Figure 1. Hunting behaviors of secretary birds (Fu et al., 2024)

The entire search process is divided into three-time intervals, allowing the algorithm to gradually transition from exploration to exploitation.

Prey Searching (Phase 1): In the early stages of the optimization process, exploration is emphasized to ensure a broad search of the solution space. To achieve this, a differential evolution-based mechanism is incorporated. This mechanism generates new candidate solutions by exploiting the differences between randomly selected individuals. As a result, population diversity is increased, and the algorithm is encouraged to explore different regions of the search space. This enhances the probability of discovering high-quality solutions. The position update in this phase is performed according to Equations (6) and (7).

$$t < \frac{1}{3}T, x_{i,j}^{newP_1} = x_{i,j} + (x_{random1} - x_{random2}) \times R_1 \tag{6}$$

$$X_i = \begin{cases} X_i^{new,P1}, if\ F_i^{new,P1} < F_i \\ X_i, otherwise \end{cases} \tag{7}$$

where, t refers to the current iteration, while T denotes the maximum number of iterations. The term X_i^{new,P_1} represents the updated position of the *i-th* individual during the first phase. In this stage, $x_{random1}$ and $x_{random2}$ correspond to randomly chosen candidate solutions. Additionally, R_1 is a randomly generated value within the interval [0,1], and *Dim* indicates the dimensionality of the search space.

Furthermore, $x_{i,j}^{newP_1}$ corresponds to the value of the *j-th* decision variable of the *i-th* bird in the first phase, and $F_{i,j}^{newP_1}$ represents the corresponding fitness value of the objective function.

Prey Exhaustion (Phase 2): After detecting the prey, the secretary bird attempts to weaken it through continuous movement and interaction. This behavior is modeled using stochastic movements based on Brownian motion. The incorporation of Brownian motion introduces randomness into the search process, which helps the algorithm avoid premature convergence and improves its ability to explore complex search spaces. The position update mechanism for this phase is defined in Equations (8)–(10).

$$RB = randn(1, Dim) \tag{8}$$

$$\frac{1}{3}T < t < \frac{2}{3}T, x_{i,j}^{new,P1} = x_{best} + exp\left(\left(\frac{t}{T}\right)\text{^}4\right) \times (RB - 0.5) \times \left(x_{best} - x_{i,j}\right) \tag{9}$$

$$X_i = \begin{cases} X_i^{new,P1}, if\ F_i^{new,P1} < F_i \\ X_i, otherwise \end{cases} \tag{10}$$

where, *randn(1,D)* generates a *1xDim* vector of random values drawn from a normal distribution. The term x_{best} represents the current best solution.

Attacking the Prey (Phase 3): Once the prey is sufficiently weakened, the secretary bird performs a decisive attack. In SBOA, this behavior is modeled by introducing a nonlinear perturbation factor. This factor enhances the adaptability of the algorithm and allows it to fine-tune candidate solutions as the search progresses. Additionally, a weighted Lévy flight mechanism is incorporated to further improve search accuracy and provide a balance between exploration and exploitation. The mathematical formulation of this phase is given in Equations (11)–(13).

$$t > \frac{2}{3}T, x_{i,j}^{newP_1} = x_{best} + \left(\left(1 - \frac{t}{T}\right)\text{^}\left(2 \times \frac{t}{T}\right)\right) \times x_{i,j} \times RL \tag{11}$$

$$X_i = \begin{cases} X_i^{new,P1}, if\ F_i^{new,P1} < F_i \\ X_i, otherwise \end{cases} \tag{12}$$

To enhance the algorithm's accuracy, a weighted *Lévy* flight mechanism, referred to as RL, is incorporated.

$$RL = 0.5 \times Levy(Dim) \tag{13}$$

where, *Levy(Dim)* corresponds to the Lévy flight distribution function and is calculated as described in Equation (14).

$$Levy(D) = s \times \frac{(u \times \sigma)}{|v|^{\frac{1}{\eta}}} \tag{14}$$

where, s and η are constants with values of 0.01 and 1.5, respectively. The variables u and v are random values within the interval [0,1]. The parameter σ is formulated as given in Equation (15).

$$\sigma = \left(\frac{\Gamma(1+\eta) \times sin\left(\frac{\pi\eta}{2}\right)}{\Gamma\left(\frac{1+\eta}{2}\right) \times \eta \times 2\left(\frac{\eta-1}{2}\right)}\right)^{\frac{1}{\eta}} \tag{15}$$

where, Γ denotes the gamma function, while η takes the value of 1.5.

Escape-Based Exploitation Mechanism

In addition to hunting, secretary birds employ strategies to escape from predators. These behaviors are incorporated into SBOA to enhance exploitation capability. Two main escape strategies are considered:

- C_1: Camouflage within the environment
- C_2: Escaping through rapid movement (running or flying)

In the first case, the individual attempts to move toward a safer region by adjusting its position relative to the best solution. If this is not effective, the second strategy is applied, where the individual performs a more dynamic movement to escape. To maintain a balance between exploration and exploitation, a dynamic control parameter is introduced. This parameter adapts throughout the iterations, allowing the algorithm to emphasize exploration in early stages and exploitation in later stages. The mathematical representation of these strategies is given in Equations (16) and (17).

$$x_{i,j}^{new,P2} = \begin{cases} C_1: x_{best} + (2 \times RB - 1) \times \left(1 - \frac{t}{T}\right)^2 \times x_{i,j}, \text{if } rand < r_i \\ C_2: x_{i,j} + R_2 \times \left(x_{random} - K \times x_{i,j}\right), otherwise \end{cases} \quad (16)$$

$$X_i = \begin{cases} X_i^{new,P2}, if\ F_i^{new,P2} < F_i \\ X_i, otherwise \end{cases} \quad (17)$$

where, $r = 0.5$, and R_2 generates a *1xDim* vector of random values drawn from a normal distribution. The term x_{random} denotes a candidate solution randomly selected from the current iteration. The parameter K, which is an integer value computed according to Equation (18), determines the random selection of the strategies.

$$K = round\big(1 + rand(1,1)\big) \quad (18)$$

where, *rand*(1,1) denotes the generation of a random number within the interval (0,1).

Binary Secretary Bird Optimization Algorithm (BSBOA)

SBOA was initially developed for continuous optimization tasks. Nevertheless, many real-world applications involve discrete or binary search spaces. A method developed for continuous domains cannot be directly applied to discrete spaces without modification. To address discrete problems, the decision variables of the method must be adapted to store and process data in a discrete form.

The binary search space is a specific type of discrete search space in which decision variables can take only two possible values, namely “0” and “1”. A wide range of methods has been introduced in the literature to convert continuous search spaces into binary domains, with transfer functions being among the most widely used and established approaches. These functions map continuous values into the interval [0,1]. The resulting values are then compared with a predefined threshold and converted into binary values, either 0 or 1.

In the literature, various methods have been proposed to transform continuous search spaces into binary ones, including S, U, V, Z, and taper-shaped transfer functions, as well as mod-based, XOR-based, and angle modulation techniques. Each of these methods has its own advantages and disadvantages.

In this study, the widely used S-shaped transfer function is employed for this transformation. The S-shaped transfer function is given in Equation (19).

$$T(a) = \frac{1}{1 + e^{-x}} \quad (19)$$

where, x represents the decision variables in SBOA; in other words, it denotes the values in the position vector of each individual. The position vector of each individual, consisting of real-valued elements, is provided as input to the corresponding transfer function. Equation (19) produces an output for each decision variable in the solution vector, yielding real values within the interval [0,1]. These values are then converted into binary values (0 or 1) using Equation (20).

$$X_{i,j}^{bin} = \begin{cases} 0, if rand < T(a) \\ 1, if rand \geq T(a) \end{cases} \quad (20)$$

where, *T(a)* represents the value returned by the transfer function. This function is applied to each position of every individual, and the resulting values are converted into binary form according to Equation (20).

In this study, the original structure of SBOA is preserved, and only a transfer function is applied after determining the new position vectors of individuals to convert them into the binary search space. No additional modifications are introduced. In this way, the performance of the original SBOA is analyzed on the knapsack problem, which is one of the binary optimization problems.

Results and Discussion

This study focuses on the one-dimensional binary knapsack problem. The format of the dataset used in the problem is presented in Figure 2. The first row of the dataset contains the number of items and the capacity of the knapsack, respectively. Each subsequent row corresponds to an item, where the first and second columns represent the profit and weight of the item, respectively.

In the solution vector, an item is represented by 1 if it is included in the knapsack and by 0 otherwise. Accordingly, the size of the solution vector of individuals varies depending on the problem dimension. In the experimental studies, a total of 28 datasets were used, consisting of 10 small-scale and 18 large-scale instances from (Pisinger, 2005). The details of the datasets are presented in Table 1.

Table 1. Characteristics of the knapsack problem instances used

Instance	Type	Dimension	Optimum Value
Ins1	small-scale	10	295
Ins2		20	1024
Ins3		4	35
Ins4		4	23
Ins5		15	481.0694
Ins6		10	52
Ins7		7	107
Ins8		23	9767
Ins9		5	130
Ins10		20	1025
Ins11	large-scale	100	9147
Ins12		200	11238
Ins13		500	28857
Ins14		1000	54503
Ins15		2000	110625
Ins16		5000	276457
Ins17		100	1514
Ins18		200	1634
Ins19		500	4566
Ins20		1000	9052
Ins21		2000	18051
Ins22		5000	44356
Ins23		100	2397
Ins24		200	2697
Ins25		500	7117
Ins26		1000	14390
Ins27		2000	28919
Ins28		5000	72505

All problem instances are one-dimensional. In addition, each item can be used only once. The best, average, worst, and standard deviation values were calculated of obtained results. Furthermore, the GAP of each result was computed according to Equation (21). In this equation, the average value represents the average of the best results obtained from 30 independent runs. All results are presented in Table 2.

$$GAP = \frac{opt - avg}{opt} \times 100 \quad (21)$$

where, *opt* denotes the optimal value for the corresponding problem (i.e., the known best solution), while *avg* represents the average of the best values obtained from 30 independent runs for that problem.

```
94 485
506 326
416 248
992 421
649 322
237 795
457 43
815 845
446 955
422 252
791 9
359 901
667 122
598 94
```

Figure 2. Dataset format

Table 2. Results obtained from BSBOA

Instance	Optimum	Best	Avg.	Worst	Std.	GAP
Ins1	295	295	295	295	0	0
Ins2	1024	1024	1004,70	978	15,96	1,88
Ins3	35	35	35	35	0	0
Ins4	23	22	22	22	0	0
Ins5	481.0694	481.0694	481.0694	481.0694	0	0
Ins6	52	52	52	52	0	0
Ins7	107	107	107	107	0	0
Ins8	9767	9767	9767	9767	0	0
Ins9	130	130	130	130	0	0
Ins10	1025	1025	1002,1	973	18,36	2,23
Ins11	9147	9147	9093,8	8929	87,25	0,58
Ins12	11238	11227	11142,5	11007	100,13	0,85
Ins13	28857	26670	26087,5	25496	324,14	9,60
Ins14	54503	47503	46672,90	46211	374,48	14,37
Ins15	110625	91776	91124,10	90259	572,97	17,63
Ins16	276457	216893	215401,00	213872	914,99	22,09
Ins17	1514	1512	1512	1512	0	0,13
Ins18	1634	1634	1634,00	1634	0	0
Ins19	4566	4453	4432,60	4404	14,56	2,92
Ins20	9052	8595	8545,40	8480	36,45	5,60
Ins21	18051	16877	16744,80	16623	79,78	7,24
Ins22	44356	40620	40465,90	40328	85,29	8,77
Ins23	2397	2397	2392,60	2390	3,37	0,18
Ins24	2697	2697	2694,60	2691	2,32	0,09
Ins25	7117	6707	6620,40	6595	31,10	6,98
Ins26	14390	13077	12985,20	12904	43,73	9,76
Ins27	28919	25512	25295,40	25178	108,61	12,53
Ins28	72505	62543	61723,70	61397	324,96	14,87

As shown in Table 2, successful results have been obtained in small-scale problems using BSBOA. Specifically, it is observed that the optimum value is reached in all 30 independent tests for all eight low-dimensional problems except Ins2 and Ins10. As expected, the quality of the results decreases as the problem size increases, which is also reflected in the GAP ratios. However, two exceptions were observed in problems Ins18 and Ins24. Apart from these instances, it is observed that in all datasets, the deviation from the optimum solution increases proportionally with the problem size.

In addition, the iteration value of the method has been kept independent of the problem size. The iteration value is given as the same for every problem. Therefore, the computation times of the method have changed proportionally with the problem size. For 4-dimensional problems, this time is approximately 9 seconds in a

system with the hardware specifications mentioned earlier, while for a 5000-dimensional problem with the largest dimensions, this time is approximately 3.5 hours.

Conclusion

In this study, the performance of the recently introduced Secretary Bird Optimization Algorithm (SBOA) is investigated on binary optimization problems. Since SBOA is originally designed for continuous search spaces, it cannot be directly applied to binary problems. Therefore, the algorithm is adapted to the binary search space using an S-shaped transfer function.

Subsequently, the adapted version, referred to as BSBOA, is applied to the well-known binary knapsack problem. The dataset consists of a total of 28 problem instances, including 10 small-scale and 18 large-scale cases. The results show that BSBOA achieves optimal solutions for all small-scale problems. However, for large-scale problems, the solutions tend to deviate from the optimal values. In addition, the computational time of the method increases with the problem size. Overall, the findings indicate that BSBOA, in its current form, is capable of producing high-quality results for knapsack problems with dimensions up to 100.

Recommendations

The experimental results indicate that improvements are needed for large-scale problems in terms of both solution quality and computational time. To enhance solution quality, alternative transfer functions can be explored, or hybrid approaches can be developed. Additionally, to reduce computational time, improvements may be made to the repair mechanism used for handling infeasible solutions or to the initialization of the population.

Scientific Ethics Declaration

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

Conflict of Interest

* The authors declare that they have no conflicts of interest.

Funding

* No financial support was received for the conduct of 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., Sallam, K. M., Chakrabortty, R. K., & Ryan, M. J. (2021). BSMA: A novel metaheuristic algorithm for multi-dimensional knapsack problems: Method and comprehensive analysis. *Computers & Industrial Engineering*, *159*(3), 107469.

Ali, I. M., Essam, D., & Kasmarik, K. (2021). Novel binary differential evolution algorithm for knapsack problems. *Information Sciences*, *542*(2), 177–194.

Feng, Y., & Wang, G. G. (2022). A binary moth search algorithm based on self-learning for multidimensional knapsack problems. *Future Generation Computer Systems*, *126*(6), 48–64.

Feng, Y., Wang, G. G., Deb, S., Lu, M., & Zhao, X. J. (2015). Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. *Neural Computing and Applications 2015 28:7*, *28*(7), 1619–1634.

Fu, Y., Liu, D., Chen, J., & He, L. (2024). Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems. *Artificial Intelligence Review 2024, 57*(5), 123-.

Granmo, O. C., Oommen, B. J., Myrer, S. A., & Olsen, M. G. (2007). Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *37*(1), 166–175.

Gupta, S., Su, R., & Singh, S. (2022). Diversified sine–cosine algorithm based on differential evolution for multidimensional knapsack problem. *Applied Soft Computing*, *130*(2), 109682.

Kong, M., Tian, P., & Kao, Y. (2008). A new ant colony optimization algorithm for the multidimensional Knapsack problem. *Computers & Operations Research*, *35*(8), 2672–2683.

Li, X., Fang, W., & Zhu, S. (2023). An improved binary quantum-behaved particle swarm optimization algorithm for knapsack problems. *Information Sciences*, *648*, 119529.

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

Tongur, V., & Ülker, E. (2017). 0-1 çok boyutlu sirt çantasi probleminin göçmen kuşlar optimizasyon algoritmasi ile çözümü. *2nd International Conference on Computer Science and Engineering, UBMK 2017*, 786–789.

Wang, L., He, Y., Wang, X., Zhou, Z., Ouyang, H., & Mirjalili, S. (2024). A novel discrete differential evolution algorithm combining transfer function with modulo operation for solving the multiple knapsack problem. *Information Sciences*, *680*, 121170.

Author(s) Information

Vahit Tongur
Konya Technical University
Ardıçlı Mahallesi Rauf Orbay Cd. 42250 Selçuklu/Konya, Turkiye
ORCID iD: https://orcid.org/0000-0001-5419-7839
*Contact e-mail: *vtongur@ktun.edu.tr*

Aysen Kucukyaglıoglu
Konya Technical University
Ardıçlı Mahallesi Rauf Orbay Cd. 42250 Selçuklu/Konya, Turkiye
ORCID iD: https://orcid.org/0009-0006-2986-5034

*Corresponding author(s) contact email

To cite this article:

Tongur, V., & Kucukyaglioglu, A. (2026). Solution of the binary knapsack problem using the secretary bird optimization algorithm. *The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 40,* 415-424.