Alexandria Engineering Journal 82 (2023) 358–376

Contents lists available at ScienceDirect

Alexandria Engineering Journal

journal homepage: www.elsevier.com/locate/aej

Original Article

A novel binary Kepler optimization algorithm for 0–1 knapsack problems: Methods and applications

Mohamed Abdel-Basset [a], Reda Mohamed [a], Ibrahim M. Hezam [b], Karam M. Sallam [c,d,*], Ahmad M. Alshamrani [b], Ibrahim A. Hameed [e,*]

[a] Faculty of Computers and Informatics, Zagazig University, Zagazig, Sharqiyah 44519, Egypt
[b] Department of Statistics & Operations Research, College of Sciences, King Saud University, Riyadh, Saudi Arabia
[c] Department of Computer Science, University of Sharjah, Sharjah, United Arab Emarites
[d] Faculty of Science and Technology, School of IT and Systems, University of Canberra, Canberra, ACT 2601, Australia
[e] Department of ICT and Natural Sciences, Norwegian University of Science and Technology (NTNU), Ålesund, Norway

ARTICLE INFO

Keywords:
Metaheuristics
Optimization
Knapsack problems
Combinatorial optimization
Approximated optimization methods
Real-world OR applications

ABSTRACT

The 0–1 Knapsack problem is a non-deterministic polynomial-time-hard combinatorial optimization problem that cannot be solved in reasonable time using traditional methods. Therefore, researchers have turned to metaheuristic algorithms for their ability to solve several combinatorial problems in a reasonable amount of time. This paper adapts the Kepler optimization algorithm using eight V-shaped and S-shaped transfer functions to create a binary variant called BKOA for solving the 0–1 Knapsack problem. Several experiments were conducted to compare the efficacy of the binary Kepler optimization algorithm to several competing optimizers when solving 20 well-known knapsack instances with dimensions ranging from 4 to 75. The experimental results demonstrate the superiority of this algorithm over other metaheuristic algorithms, except for the genetic algorithm, which is marginally superior. To further improve the binary Kepler optimization algorithm, it is combined with an enhanced improvement strategy to create a new hybrid variant. This hybrid variant, termed HBKOA, has superior exploration and exploitation capabilities that make it better than genetic algorithm and other optimizers for all performance metrics considered. The enhanced improvement strategy is also integrated with several competing optimizers, and the experimental results show that HBKOA, hybrid binary manta ray foraging optimization, and hybrid binary equilibrium optimizer are competitive for small and medium-dimensional instances and superior for the higher dimensions.

1. Introduction

Optimization problems in real-world applications are classified into two categories: Discrete and continuous. In continuous problems, the decision variables, also referred to as dimensions, include only decimal values; on the contrary, the decision variables in discrete problems include either integers or binary values. The metaheuristic algorithms could find acceptable solutions for the majority of those problems in an acceptable time. The continuous optimization problems are widely spread in several real-world applications, like parameter estimation of photovoltaic models and fuel cells, engineering design problems, and several else. The majority of discrete problems could not be solved in a reasonable amount of time by the traditional methods because they include a huge number of solutions that need to be checked to find the acceptable solutions. As aforementioned, the metaheuristics have been proposed for only solving the continuous optimization problems, and hence, to make their solutions suitable for the discrete problems, they have to be adapted using external methods, which might negatively affect their performance. In general, the solutions of the metaheuristic algorithms could not be applied directly to the discrete problems, but rather, some methods take their solutions and convert them into suitable solutions for the tackled problem. Hence, the performance of metaheuristics is more dependent on the integrated methods than the performance of the algorithm itself. There are several discrete optimization problems, like 0–1 knapsack problems (KP01), multidimensional knapsack problems, feature selection, task scheduling problems, and DNA fragment assembly problems [1,2].

* Corresponding authors at: Department of Computer Science, University of Sharjah, Sharjah, United Arab Emarites (K.M. Sallam).
E-mail addresses: mohamedbasset@ieee.org (M. Abdel-Basset), redamoh@zu.edu.eg (R. Mohamed), ialmishnanah@ksu.edu.sa (I.M. Hezam), karam.sallam@canberra.edu.au, KSallam@sharjah.ac.ae (K.M. Sallam), ialmishnanah@ksu.edu.sa (A.M. Alshamrani), ibib@ntnu.no (I.A. Hameed).

https://doi.org/10.1016/j.aej.2023.09.072
Received 18 July 2023; Received in revised form 5 September 2023; Accepted 27 September 2023
Available online 14 October 2023

In this paper, we will focus on the 0–1 knapsack problems (KP01) due to their significant importance in several fields, like real estate property maintenance optimization, adaptive multimedia systems, resource allocation, principal budgeting, and cargo loading [3]. In literature, there are two types of techniques for solving KP01, namely traditional techniques and approximation or metaheuristic techniques. The traditional techniques have failed to solve those problems due to their search methodology that are based on exploring all possible solutions for this problem. Hence, for the high dimensional problems, they could not explore all possible solutions. For instance, supposing that there is a KP01 problem with d dimensions, then the number of possible solutions that need to be explored are 2^d; if d is 20, the possible solutions are 1048576. In a nutshell, KP01 is classified as non-deterministic polynomial-time-hard combinatorial problems; hence the traditional techniques are not preferred for solving them [1]. Therefore, researchers, over the last few decades, have paid attention to metaheuristic algorithms due to their potential for finding acceptable solutions for several combinatorial problems [4]. in literature, the metaheuristic algorithms have been classified into four main categories: Evolution-based, physics-based, swarm-based, and human-based, each of which simulates specific behaviors for proposing a novel algorithm with different characteristics (Exploration and exploitation). The authors in the literature have adapted those novel algorithms for solving KP01 problems using various transfer functions; some of those algorithms are reviewed next.

In [1], a binary variant of the Aquila optimizer, namely BAO, has been proposed for solving KP01. In this variant, the performance of eight transfer functions was investigated to extract the best one that could maximize the performance of BAO. In addition, in the same study, an additional variant of BAO was proposed based on borrowing both the crossover operator and mutation operator from the genetic algorithms to further improve its exploration and exploitation operators. In the same context, Yildizdan proposed a binary variant of the artificial jellyfish search algorithm for tackling the same problem; this variant has been called Bin_AJS [5]. Several transfer functions were investigated to disclose their influence on the performance of this variant. According to [5], The experimental findings have exposed the Bin_AJS's superiority in comparison to several other optimizers. In [6], the reptile search algorithm has been adapted under several transfer functions for solving KP01; this binary variant was called BRSA. In addition, to further improve the performance of BRSA, it was integrated with a stochastic repair and improvement method. In [7], the original marine predators algorithm was adapted to deal with discrete problems like 0–1 KP. The V-shaped and S-shaped transfer functions were used to convert continuous values into binary, allowing for the development of the binary extension of this algorithm, entitled BMPA. The binary slime mould algorithm (BSMA) was improved in [8] for accurately solving KP01. Similarly, in [9], the standard binary equilibrium has been converted into a binary variant under V-shaped and S-shaped transfer functions for solving KP01.

Li [10] proposed a binary variant of the particle swarm optimization algorithm (PSO), namely binary quantum-behaved PSO (BQPSO). However, the authors found that this variant suffers from slow convergence and local optima problems. Therefore, it was enhanced by presenting a new mapping strategy based on all particles' average positions, a new repair method for handling the solutions that could not achieve the knapsack capacity, and a diversity improvement mechanism to preserve the population diversity to avoid getting stuck into local minima. This improved variant was compared to several optimizers on different KP01 instances to reveal its effectiveness. Büyüköz [11] investigated the performance of several transfer functions, including Tangent Sigmoid, S-Shaped, O-Shaped, T-Shaped, V-Shaped, Z-Shaped, and U-Shaped for adapting the honey badger algorithm to solve 25 KP01 instances. Du et al [12] solved the multidimensional knapsack problems using a novel binary variant of the fruit fly optimization algorithm. This variant was improved using several mechanisms, such as a multi-swarm cooperation strategy, DROP and ADD operators, and an item frequency tree (IFT); all of those were used to improve the convergence speed and prevent falling into local minima.

Ballinas [13] proposed a new variant of the genetic algorithm (GA) for solving the KP01. This variant was based on integrating GA with two mechanisms, namely Deutsch-Jozsa quantum circuit, and adaptive rotation angle. This variant was compared to several techniques for solving several KP01 instances. There are several other metaheuristic algorithms proposed for solving the knapsack problems, some of which are the Giza pyramids construction [14], simulated annealing [15], modified artificial bee colony [16], monarch butterfly optimization [17], differential evolution integrated with grey wolf optimizer [18], bat algorithm [19], gradient-based optimizer [20], harmony search algorithm [21], rice optimization algorithm [22], Archimedes optimization algorithm [23], imperialist competitive algorithm [24], nutcracker optimization algorithm [25], synergistic fibroblast optimizer [26], sine cosine algorithm [27–29], coyote optimization algorithm [30], grey wolf optimizer [4], differential evolution [31], moth search algorithm [32], and whale optimization algorithm [33].

Recently, a new metaheuristic optimization algorithm called the Kepler optimization algorithm (KOA), inspired by Kepler's laws on the motion of the planets was proposed for tackling continuous optimization problems [34]. The experimental findings show the high effectiveness of this algorithm for solving various continuous problems. This motivates us to observe its performance for KP01. To make it applicable to binary space, two different families of transfer functions, namely V-shaped and S-shaped, are employed to identify the best TF that could present a strong binary variant of KOA for accurately solving KP01. This binary variant of KOA is called BKOA. First, several experiments have been conducted to compare the performance of BKOA against several rival optimizers for solving 20 well-known knapsack instances with a number of dimensions ranging between 4 and 75. This comparison is based on several performance metrics, such as best fitness, average fitness, worst fitness, standard deviation, computational cost, and Friedman mean rank. This comparison shows the effectiveness of BKOA over several metaheuristic algorithms, with the exception of the genetic algorithm (GA), which is slightly better than BKOA. To improve BKOA over GA, it is effectively integrated with a novel strategy, called enhanced improvement strategy (EIS), to propose a new variant, namely HBKOA. This variant has higher exploration and exploitation capabilities than BKOA. These capabilities make it better than GA and the others for all considered performance metrics. To reveal the influence of EIS on the performance of binary metaheuristic algorithms, it is integrated with some rival optimizers such as the binary equilibrium optimizer (BEO), the binary marine predators algorithm (BMPA), the binary manta ray foraging optimization (BMRFO), the binary teaching–learning-based optimization (BTLBO), the binary Jaya (BJA), and the binary Young's double-slit experiment optimizer (BYDSE). The experimental findings show the competitiveness between HBKOA, HBMRFO, and HBEO for small- and medium-dimensional KP01 instances. Meanwhile, HBMRFO could perform better than all for high-dimensional KP01 instances. Briefly, the main contributions of this study are listed below:

- Presenting a binary variant of KOA, namely BKOA, for solving the KP01 using two different families of transfer functions, namely S-shaped and V-shaped transfer functions.
- Integrating BKOA with an EIS to propose a new variant called HBKOA with better exploration and exploitation operators for effectively solving KP01.
- Integrating EIS with several other metaheuristics to disclose its influence on their performance
- Conducting several experiments to reveal the performance of BKOA and HBKOA against several other competitors for solving 20 small- and medium-dimensional KP01 instances. Those experiments proved the effectiveness of BKOA over the compared algorithms, except for GA, which is slightly better than HBKOA.

- Conducting other experiments to disclose the performance of some metaheuristic algorithms improved by the EIS strategy. The experimental results demonstrate the competition between HBKOA, HBMRFO, and HBEO for tiny and medium-sized KP01 instances. Meanwhile, HBMRFO can outperform all others for high-dimensional KP01 instances.

The remainder of this study is organized as follows: Section 2 provides an overview of the Kepler optimization algorithm, Section 3 describes the proposed algorithm, Section 4 contains a discussion and comparison, and Section 5 provides a conclusion regarding our work and future work.

2. Kepler optimization algorithm (KOA)

In [34], a new optimization technique called the Kepler optimization algorithm (KOA) inspired by Kepler's laws of planetary motion has been recently proposed for tackling optimization problems. In a nutshell, KOA is built on the same two operators as the other metaheuristic algorithms, which are exploitation and exploration. When the planet is relatively distant from the sun, the exploratory operator takes place, and when it is relatively close to the sun, the other operator takes place. This section continues with a detailed discussion of the steps of KOA.

Step 1: Initialization step

Like the other population-based metaheuristic algorithms, KOA first creates a two-dimensional matrix of $N\times d$, where N represents the population size and d represents the number of decision variables or dimensions of the tackled optimization problems. Then, this matrix is randomly initialized within the lower and upper bounds of each dimension, as formulated in the following formula:

$$\overrightarrow{X_i} = \overrightarrow{X_{lb}} + \overrightarrow{r} \times \left(\overrightarrow{X_{up}} - \overrightarrow{X_{lb}}\right), \begin{cases} i = 1, 2, \cdots, N. \\ j = 1, 2, \cdots, d. \end{cases} \tag{1}$$

where $\overrightarrow{X_i}$ represents the ith solution, $\overrightarrow{X_{up}}$ is the upper bound vector of the dimensions in the tackled problem, $\overrightarrow{X_{lb}}$ represents the lower bound vector of the same dimensions, $\overrightarrow{r}$ is a vector including numerical values generated randomly between 0 and 1 based on the normal distribution. Before starting the optimization process, KOA also needs to initialize some other controlling parameters, such as the orbital eccentricity (e) and orbital period (T).e and T are randomly initialized for each planet/solution in the population, as formulated, respectively, in the following two formulas:

$$e_i = r,\ i = 1, \cdots, N \tag{2}$$

$$T_i = |rn|,\ i = 1, \cdots, N \tag{3}$$

where rn is a variable including a numerical value generated randomly according to the normal distribution.

Step 2: Defining the gravitational force (.**F**)

In KOA, the universal law of gravity is employed to determine the gravity strength of the Sun $\overrightarrow{X_S}$ to each planet $\overrightarrow{X_i}$, as formulated in the following formula:

$$\boldsymbol{F}_{g_i}(t) = e_i \times \mu(t) \times \frac{\overline{M}_s \times \overline{m}_i}{\overline{R}^2_i + \varepsilon} + r_1 \tag{4}$$

where ε is a tiny number that is used to prevent division by zero, r_1 is a number chosen at random between 1 and 0. $\overline{M}_s$ and$\overline{m}_i$ stand for the normalized values of both M_s and m_i, respectively, where M_s and m_i represent the mass of both $X_S$andX_i, respectively, and are computed according to Eqs. (7) and (8); μ is a constant value to determine the universal gravitational constant; and $\overline{R}_i$ is the normalized value of R_i that represents the Euclidian distance between $\overrightarrow{X_S}$ and $\overrightarrow{X_i}$, and computed according to the next equation:

$$R_i(t) = \|X_S(t) - X_i(t)\|_2 = \sqrt{\sum_{j=1}^{d} \left(X_{Sj}(t) - X_{ij}(t)\right)^2} \tag{5}$$

$$\overline{R}_i = \frac{R_i(t) - \min(R(t))}{\max(\mathrm{R}(t)) - \min(R(t))} \tag{6}$$

$$M_s = r_2 \frac{fit_s(t) - worst(t)}{\sum_{k=1}^{N}(fit_k(t) - worst(t))} \tag{7}$$

$$m_i = \frac{fit_i(t) - worst(t)}{\sum_{k=1}^{N}(fit_k(t) - worst(t))} \tag{8}$$

$$fit_s(t) = best(t) = \min_{k\in\{1,2,\cdots,N\}} fit_k(t) \tag{9}$$

$$worst(t) = \max_{k\in\{1,2,\cdots,N\}} fit_k(t) \tag{10}$$

where r_2 is a numerical value chosen at random in the interval (0, 1). $\mu(t)$ is computed using the following formula:

$$\mu(t) = \mu_0 \times \exp(-\gamma \frac{t}{T_{max}}) \tag{11}$$

where γ is a constant value; μ_0 refers to the initial value; T_{max} and t indicate the maximum number of function evaluations and current function evaluations, respectively.

Step 3: Calculating an object's velocity

The distance of a planet from the Sun determines its velocity. In other words, a planet's speed increases as it approaches the Sun and decreases as it recedes from it. Planets and other objects try to increase their velocity as they approach the Sun because the Sun's gravity is much stronger, so they attempt to accelerate up to avoid being drawn in the Sun. According to KOA [34], the velocity of each planet in accordance with its distance from the Sun is computed according to the following equation:

$$V_i(t) = \begin{cases} l\left(2r_4\overrightarrow{X_i} - \overrightarrow{X_b}\right) + \ddot{\ell}\left(\overrightarrow{X_a} - \overrightarrow{X_b}\right) + (1 - R_{i-norm}(t))F\overrightarrow{U}_1\overrightarrow{r_5}\left(\overrightarrow{X_{up}} - \overrightarrow{X_{lb}}\right), & if\ R_{i-norm}(t) \le 0.5 \\ r_4 L\left(\overrightarrow{X}_a - \overrightarrow{X}_i\right) + (1 - R_{i-norm}(t))FU_2\overrightarrow{r_5}\left(r_3\overrightarrow{X_{up}} - \overrightarrow{X_{lb}}\right), & Else \end{cases} \tag{12}$$

$$\ell = \overrightarrow{U}\mathscr{M}\mathscr{L} \tag{13}$$

$$\mathscr{L} = \left[\mu(t)(M_S + m_i)\left(\frac{2}{R_i(t) + \varepsilon} - \frac{1}{a_{i(t)} + \varepsilon}\right)\right]^{\frac{1}{2}} \tag{14}$$

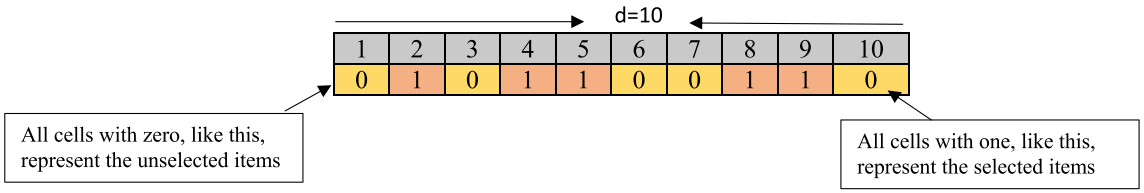


Fig. 1. A solution encoding for KP01.

$$\mathscr{M} = (r_3(1 - r_4) + r_4) \tag{15}$$

$$\overrightarrow{U} = \begin{cases} 0, \ \overrightarrow{r_5} \leq \overrightarrow{r_6} 1, & Else \end{cases} \tag{16}$$

$$\mathscr{F} = \begin{cases} 1, \ if \ r_4 \leq 0.5 \\ -1, \quad Else \end{cases} \tag{17}$$

$$\ddot{\mathscr{C}} = \left(1 - \overrightarrow{U}\right) \times \overrightarrow{\mathscr{M}} \times \mathscr{L}, \tag{18}$$

$$\overrightarrow{\mathscr{M}} = (r_3 \times (1 - \overrightarrow{r_5}) + \overrightarrow{r_5}) \tag{19}$$

$$\overrightarrow{U}_1 = \begin{cases} 0, \ \overrightarrow{r_5} \leq r_4 1, \ Else \end{cases} \tag{20}$$

$$U_2 = \begin{cases} 0, \ if \ r_3 \leq r_4 \\ 1, \ Else \end{cases}. \tag{21}$$

$$\overrightarrow{X}_i(t+1) = \overrightarrow{X}_i(t) + \mathscr{F} \times \overrightarrow{V}_i(t) + \left(\boldsymbol{F}_{\boldsymbol{g}_i}(t) + |r|\right) \times \overrightarrow{U} \times \left(\overrightarrow{X_S} - \overrightarrow{X_i}(t)\right). \tag{23}$$

Step 6: Updating distance with the Sun

Simulating the naturally occurring change in the distance between the Sun and the planets over time is another attempt to enhance the KOA's exploration and exploitation capabilities. Whenever a planet is in close proximity to the Sun, KOA will give that planet preference for exploitation, but when the Sun is far away, KOA will give preference to exploration. This concept is mimicked by using a time-varying value for the controlling parameter h. If it is high, the exploration operator is utilized to broaden the search space for a better solution, while if it is low, the exploitation operator is employed to focus further on the solutions around the best-so-far solutions. The mathematical model for this principle can be expressed in terms of the following equations:

$$\overrightarrow{X}_i(t+1) = \overrightarrow{X}_i(t) \times \overrightarrow{U}_1 + \left(1 - \overrightarrow{U}_1\right) \times \left(\overrightarrow{X}_i(t) + \frac{\overrightarrow{X_S} + \overrightarrow{X_a}(t)}{3.0} + h \times \left(\overrightarrow{X}_i(t) + \frac{\overrightarrow{X_S} + \overrightarrow{X_a}(t)}{3.0} - \overrightarrow{X_b}(t)\right)\right), \tag{24}$$

where $\overrightarrow{V_i}(t)$ represents the velocity of the *ith* planet, r_3 and r_4 are two numerical values chosen at random between 0 and 1 according to the uniform distribution; and $\overrightarrow{r_6}$ and $\overrightarrow{r_5}$ stands for two vectors of numerical values that are generated arbitrarily in the range (0, 1). $\overrightarrow{X_a}$ and $\overrightarrow{X_b}$ are two objects chosen arbitrarily from the individuals in the current population; $\mathscr{F}$ is a variable including randomly one value of [1, −1] to alter the search direction; and a_i could be estimated according to the following equation:

$$a_i(t) = r_3 \times \left[T_i^2 \times \frac{\mu(t) \times (M_s + m_i)}{4\pi^2}\right]^{\frac{1}{3}}. \tag{22}$$

Step 4: Escaping from the local optimum

The majority of planets in the solar system revolve around their axes and orbit in an anticlockwise direction relative to the Sun, while some others revolve clockwise around the sun. This fact is simulated in KOA to vary the exploration capabilities of KOA to avoid getting stuck in local minima and achieve better outcomes. This fact has been mimicked based on the factor $\mathscr{F}$, which only includes two integers, 1 and −1, where −1 is used to make some planets revolve clockwise relative to the sun.

Step 5: Updating objects' positions

After estimating the velocity of each planet, its new position could be computed using the next mathematical formula:

$$h = \frac{1}{e^{\eta r}}, \tag{25}$$

where r is a numerical value chosen arbitrarily based on the normal distribution, and η is computed as:

$$\eta = (a_2 - 1) \times r_4 + 1 \tag{26}$$

where a_2 stands for a cyclic controlling parameter, which is computed using the following equation:

$$a_2 = -1 - \left(\frac{t\%\frac{T_{max}}{TC}}{\frac{T_{max}}{TC}}\right) \tag{27}$$

where *TC* represents the number of cycles in the optimization process, and % denotes the remainder operator.

Step 7: Elitism

This step establishes an elitist strategy to guarantee that the Sun and planets are always in the best local positions, as defined by the next mathematical formula:

Table 1
An illustrative example of KP01.

Index	1	2	3	4	5	6	7	8	9	10	
$\overrightarrow{X_t}$	0	1	0	1	1	0	0	1	1	0	
w	20	18	17	15	15	10	5	3	1	1	
p	30	25	20	18	17	11	5	2	1	1	
p_jX_j	0	25	0	18	17	0	0	2	1	0	$\sum_{j=1}^{d} p_j X_j = 63$
w_jX_j	0	18	0	15	15	0	0	3	1	0	$\sum_{j=1}^{d} w_j X_j = 52$

$$\overrightarrow{X_{i,new}}(t+1) = \begin{cases} \overrightarrow{X_i}(t+1), & if\ f(\overrightarrow{X_i}(t+1) \le f\left(\overrightarrow{X_i}(t)\right)\overrightarrow{X_i}(t), \quad Else \end{cases} \tag{28}$$

Algorithm 1 Steps of KOA.

Start
Set $N, T_{max}, \mu_0, \gamma, \overline{T}$.
Initialization
Evaluation.
Determine the best-so-far solution ($\overrightarrow{X_S}$).
While ($t < T_{max}$)
Update $e_i i = 1, 2, \cdots, N, best(t), worst(t),\ and\ \mu(t)$.
For $i = 1{:}N$
Calculate $\overline{R}_i$ between $\overrightarrow{X_S}$ and $\overrightarrow{X_i}$ using Eq. (6).
Calculate $\boldsymbol{F}_{g_i}$ between $\overrightarrow{X_S}$ and $\overrightarrow{X_i}$ using Eq. (4).
Calculate the velocity of $\overrightarrow{X_i}$ using Eq. (12).
Generate two random numbers r, r_1 between 0 and 1.
If $r > r_1$ /* Update position of the planet*/
Update $\overrightarrow{X_i}$ using Eq. (23).
Else /***update the distance between the planet and the sun***/
Update $\overrightarrow{X_i}$ using Eq. (24).
End if
Apply Eq.(28)
$t = t+1$ ***End for***
End while
Stop

3. The proposed binary algorithms

This section presents the main steps used to make the continuous KOA applicable to the binary space. Those steps are initialization, normalization under various V-shaped and S-shaped transfer functions, extracting binary solutions, and evaluation. In addition, the enhanced improvement strategy is presented to improve the quality of the obtained binary solutions. This strategy is integrated with the binary KOA (BKOA) to present a new strongly-performing variant called hybrid BKOA (HBKOA).

3.1. Initialization

Before starting the optimization process, the metaheuristic algorithms randomly distribute N individuals with d dimensions within the search space of the tackled optimization problems. The solutions suitable for the KP01 must include binary values to determine whether the item is selected or not. Therefore, in the initialization step of KOA and the other optimizers, all solutions are randomly distributed with 0 and 1, as defined in the following formula:

$$\overrightarrow{X_i} = \begin{cases} 1, & if\ \overrightarrow{r} > 0.5 \\ 0, & otherwise \end{cases}. \tag{29}$$

where $\overrightarrow{r}$ is a vector including numerical values generated randomly between 0 and 1 according to the uniform distribution. In Fig. 1, we encode a suitable solution for KP01 to illustrate how to select the items that are added to the knapsack. For instance, in this figure, items 2, 4, 5, 8, and 9 are added to the knapsack because the binary solution includes 1 in the corresponding cells.

Table 2
S-shaped and V-shaped TF's Mathematical model.

S-Shaped	V-Shaped
S1 $F(\overrightarrow{X}) = \frac{1}{1+e^{-\overrightarrow{X}}}$	V1 $F(\overrightarrow{X}) = \left\lvert \frac{2}{\pi} \text{arcTan}\left(\frac{\pi}{2}\overrightarrow{X}\right) \right\rvert$
S2 $F(\overrightarrow{X}) = \frac{1}{1+e^{-2\overrightarrow{X}}}$	V2 $F(\overrightarrow{X}) = \left\lvert \tanh\left(\overrightarrow{X}\right) \right\rvert$
S3 $F(\overrightarrow{X}) = \frac{1}{1+e^{-\frac{\overrightarrow{X}}{2}}}$	V3 $F(\overrightarrow{X}) = \left\lvert \frac{a}{\sqrt{1+\overrightarrow{X}^2}} \right\rvert$
S4 $F(\overrightarrow{X}) = \frac{1}{1+e^{-\frac{\overrightarrow{X}}{3}}}$	V4 $F(\overrightarrow{X}) = \left\lvert \text{erf}\left(\frac{\sqrt{\pi}}{2}\overrightarrow{X}\right) \right\rvert$

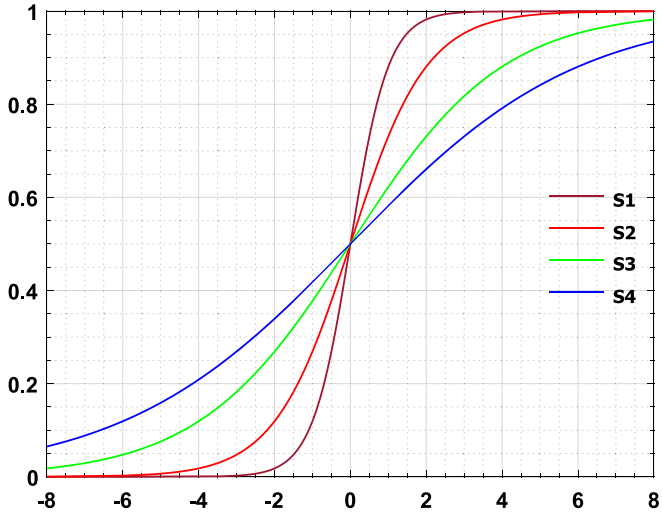


Fig. 2. S-shaped TFs.

3.2. Evaluation: Objective function

The objective function of KP01 is based on finding a set of items that maximizes the total profits and is subject to the knapsack capacity. This objective function is formulated as follows:

$$Maximize\ \ \mathrm{f}\left(\overrightarrow{X_i}\right) = \sum_{j=1}^{d} p_j X_j. \tag{30}$$

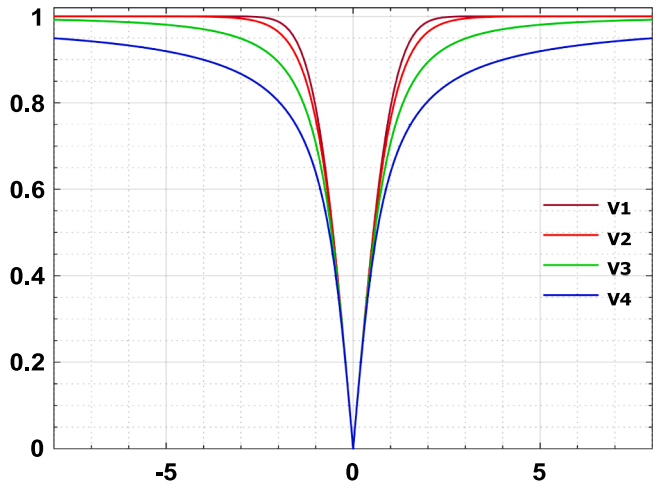


Fig. 3. V-shaped TFs.

Subject to $\sum_{j=1}^{d} w_j X_j \leq C.$

where w_j represents the *jth* item's weight, X_j is the *jth* cell in the *ith* binary solution to determine whether the *jth* item is selected or not, p_j refers to the profit of the *jth* item, and C represents the knapsack capacity. For example, assume that we have a knapsack problem with the following characteristics:

- $d = 10$
- $C = 60$
- $\vec{w} = [20, 18, 17, 15, 15, 10, 5, 3, 1, 1]$
- $\vec{p} = [30, 25, 20, 18, 17, 11, 5, 2, 1, 1]$

This problem is solved by finding a set of items that could satisfy eq. (30). This set is selected based on creating a binary solution with d dimensions, whereby each dimension determines if the current item is selected or not. For example, suppose that BKOA could generate the

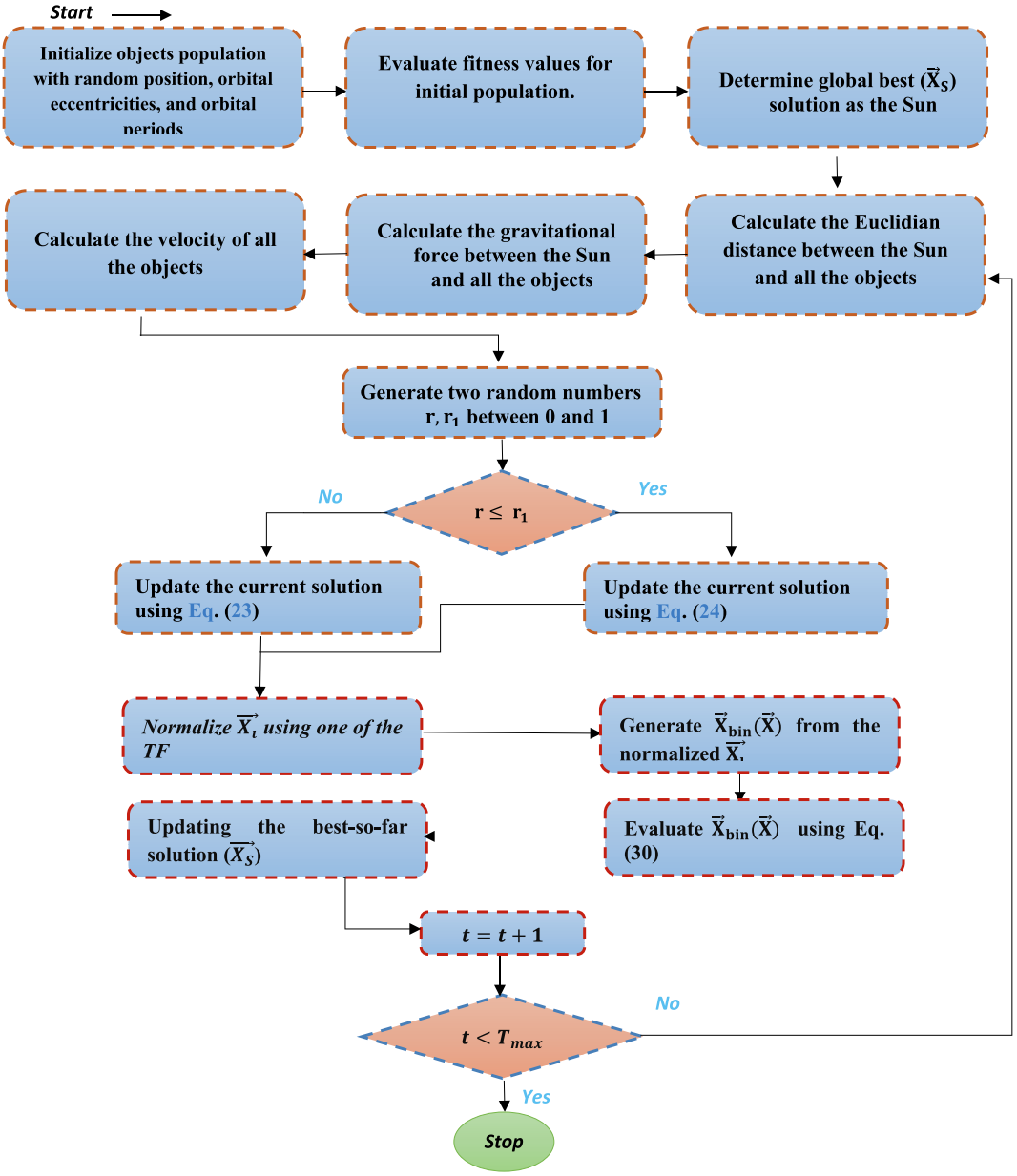


Fig. 4. Flowchart of BKOA.

Table 3
An example to illustrate a disadvantage of IA.

Selecting the item 1 since it has the highest ratio. **Discarded**

Index	1	2	3	4	5	6	7	8	9	10	
$\vec{X_i}$	0	1	0	1	1	0	1	1	1	1	
w	20	18	17	15	15	3	5	3	1	1	
p	30	25	20	18	17	2	5	2	1	1	
$\frac{p_j}{w_j}$	1.5	1.39	1.18	1.2	1.13	0.66	1	0.6	1	1	
p_jX_j	30	25	0	18	17	0	0	2	1	0	$\sum_{j=1}^{d} p_j X_j = 93$
w_jX_j	20	18	0	15	15	0	0	3	1	0	$\sum_{j=1}^{d} w_j X_j = 83 > 60$
Step 2: Selecting the item 3 since it has the second highest ratio. Discarded											
p_jX_j	0	25	17	18	17	0	0	2	1	0	$\sum_{j=1}^{d} p_j X_j = 80$
w_jX_j	0	18	20	15	15	0	0	3	1	0	$\sum_{j=1}^{d} w_j X_j = 72 > 60$
Step 3: Selecting the item 7 since it has the third highest ratio. Considered											
p_jX_j	0	25	0	18	17	0	5	2	1	0	$\sum_{j=1}^{d} p_j X_j = 68$
w_jX_j	0	18	0	15	15	0	5	3	1	0	$\sum_{j=1}^{d} w_j X_j = 57 < 60$
Step 4: Selecting the item 10 since it has the fourth highest ratio. Considered											
p_jX_j	0	25	0	18	17	0	5	2	1	1	$\sum_{j=1}^{d} p_j X_j = 69$
w_jX_j	0	18	0	15	15	0	5	3	1	1	$\sum_{j=1}^{d} w_j X_j = 58 < 60$

*__Red cells__ represent the discarded items due to destroying the knapsack capacity constraint.
*__Blue cells__ represent the considered items due to satisfying the knapsack capacity constraint.
*__Green cell__ represents the item that could achieve better profit if it has been considered first.

Table 4
Characteristics of the KP01 instances.

IN	Capacity	D	Opt	IN	Capacity	D	Opt	IN	Capacity	D	Opt
KP1	269	10	295	KP12	655	35	1689.0	KP22	1008	200	11,238
KP2	878	20	1024	KP13	819	40	1821	KP23	2543	500	28,857
KP3	20	4	35	KP14	907	45	2033	KP24	5002	1000	54,503
KP4	11	4	23	KP15	882	50	2440	KP25	995	100	1514
KP5	375	15	481.069368	KP16	1050	55	2651	KP26	1008	200	1634
KP6	60	10	52	KP17	1006	60	2917	KP27	2543	500	4566
KP7	50	7	107	KP18	1319	65	2817	KP28	5002	1000	9052
KP8	10,000	23	9767	KP19	1426	70	3223	KP39	997	100	2397
KP9	80	5	130	KP20	1433	75	3614	KP30	997	200	2697
KP10	879	20	1025	KP21	995	100	9147	KP31	2517	500	7117
KP11	577	30	1437					KP32	4990	1000	14,390

solution $\vec{X}_i$ depicted in Fig. 1. The quality of this solution is determined by passing it to Eq. (30) to compute the total profit based on summing the results of multiplying the binary vector $\vec{X}_i$ by the profit vector $\vec{p}$, as described in Table 1. The total profit based on this calculation is 63. But, the total weight of this solution must be subject to the knapsack capacity to be considered as a feasible solution. To check this constraint, the total weight is also computed based on summing the results of multiplying the binary vector $\vec{X}_i$ by the weight vector $\vec{w}$, as described in the last column in Table 1. This solution is considered feasible for solving this knapsack problem because its total profit is smaller than the permitted knapsack capacity.

3.3. S-shaped and V-shaped transfer functions

Typically, metaheuristic algorithms are presented for solving continuous optimization problems. Hence, they could not be directly applied to tackle KP01. To adapt the continuous metaheuristic algorithms for KP01, various V-shaped and S-shaped transfer functions (TF) are employed to normalize the continuous values between 0 and 1, and then those normalized values are randomly converted into binary values, as modeled in Eq. (31). The mathematical models of various V-

and S-shaped TFs are presented in Table 2. In Addition, the curves of V-shaped and S-shaped TFs are depicted in Figs. 2 and 3.

$$\overrightarrow{X}_{bin}\left(\overrightarrow{X}\right) = \begin{cases} 1, & \text{if } F\left(\overrightarrow{X}\right) \geq rand \\ 0, & otherwise \end{cases}. \tag{31}$$

3.4. Binary KOA (BKOA)

As stated before, KOA has been proposed for tackling continuous optimization problems, and hence it could not be directly applied to tackling binary optimization problems. To make it applicable to tackle this problem, it first initializes N solutions randomly with binary values that are then evaluated to determine their quality. The initial solution with the highest objective value and satisfying the knapsack capacity is considered the best-so-far solution. Afterward, the optimization process is started to update those initial solutions and generate new continuous solutions. Those continuous solutions are normalized using one of the V- or S-shaped TFs and converted under the randomization process into binary solutions $\overrightarrow{X_{bin}}(\overrightarrow{X})$, as depicted in Eq. (31). Those binary solutions are evaluated using Eq. (30) to determine the quality of each solution, and the solution that has the highest objective function and could satisfy the knapsack capacity is considered the best-so-far solution ($\overrightarrow{X_S}$). This process is continued until the maximum number of function evaluations are satisfied. Briefly, the steps of the binary variant of KOA (BKOA) are depicted in Fig. 4.

3.5. Enhanced improvement strategy (EIS)

In [7], an improvement algorithm (IA) has been proposed to improve the quality of the binary solutions obtained by the binary metaheuristic algorithms. According to [7], the steps of IA are described as that:

1. The item i with the highest $\frac{p_i}{w_i}$ is selected in the binary solution $\overrightarrow{X}_{bin}$.
2. Evaluating the updated solution $\overrightarrow{X}_{bin}$
3. If the new solution satisfies the knapsack capacity, it is considered for the next update under the IA; otherwise, the last update is discarded, and the IA moves to the item with the second highest $\frac{p_i}{w_i}$.
4. Repeating the previous three steps for checking all unselected items in the binary solution $\overrightarrow{X}_{bin}$.

IA has some disadvantages, as described below:

- Selecting the items in the knapsack according to the highest ratio based on dividing profit by weight might result in discarding some items with a smaller ratio but could achieve better profit, as discussed in Table 3.
- Traversing all unselected items in each solution consumes a huge number of function evaluations, thereby increasing the computational cost consumed by each algorithm.

To overcome those disadvantages, a new enhanced variant of IA, namely enhanced improvement strategy (EIS), is proposed in this study to achieve the following two purposes:

- Taking into consideration all items, even those with a small ratio.
- Saving the computational cost as much as possible.

The first purpose of this algorithm is achieved by shuffling the indices of a percent β of the solutions that have the highest ratio. Hence, selecting the item is not dependent on the highest ratio but is based on its order in the shuffled indices (I). β is a predefined controlling parameter discussed in the experiments section. The second purpose is achieved by using a counter to determine the number of times the objective function is called. If this counter exceeds a specific ratio (γ), the EIS algorithm is stopped, and the control is returned to the binary metaheuristic algorithm. Finally, the steps of EIS are described in Algorithm 3. Finally, this algorithm is integrated with BKOA to present a better binary variant called HBKOA. In this variant, EIS is employed to improve the quality of the feasible solutions obtained by the classical BKOA, as described in Algorithm 3.

Algorithm 2 Steps of EIS

Input: $\overrightarrow{X}_{bin}$
1. $C = 0$, A counter to determine the number of function evaluations
2. $f =$ compute the fitness value of $\overrightarrow{X}_{bin}$
3. **for** $j = 1{:}\ d$ %% ***Ratio between weights and profits***
4. $R_j = \frac{p_j}{w_j}$
5. **end**
6. Sort $\boldsymbol{R}$ in descending order and return the sorted indices **I**
7. **I(1:**$\beta \times d$**) = shuffled(I(1:**$\beta \times d$**))**
8. **for** $j = 1{:}\ d$
9. **if** $\overrightarrow{X}_{bin,I(j)} == 0$
10. $\overrightarrow{X}_{bin,I(j)} = 1$
11. $f =$ compute the fitness value of $\overrightarrow{X}_{bin}$
12. **if** $f == 0$ %% ***infeasible solution***
13. $\overrightarrow{X}_{bin,I(j)} = 0$ %% Discard the above update
14. **end**
15. $C++$
16. **if** $C > \gamma \times d$
17. **Break;**
18. **end**
19. **end**
20. **end**

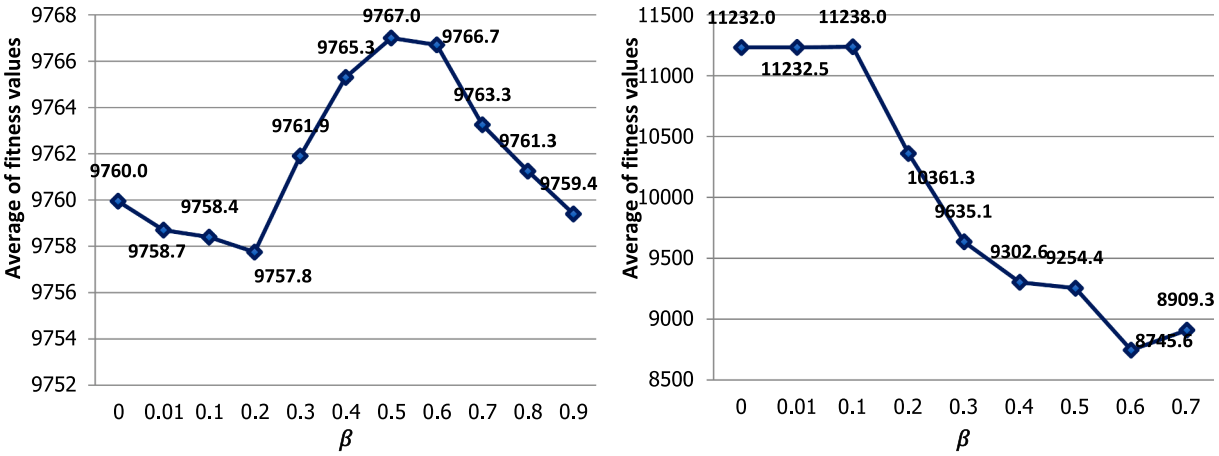
Return $\overrightarrow{X}_{bin}$, C

Algorithm 3 Steps of BKOA.

Start
Set $N, T_{max}, \mu_0, \gamma, \overline{T}$.
Initialize N solutions using Eq. (29)
Evaluate each solution using Eq. (30)
Determine the best-so-far solution ($\overrightarrow{X_S}$).
While $(t < T_{max})$
Update $e_i i = 1, 2, \cdots, N, best(t),\ worst(t),\ and\ \mu(t)$.
For $i = 1{:}N$
Calculate $\overline{R}_i$ between $\overrightarrow{X_S}$ and $\overrightarrow{X_i}$ using Eq. (6).
Calculate $\boldsymbol{F}_{g_i}$ between $\overrightarrow{X_S}$ and $\overrightarrow{X_i}$ using Eq. (4).
Calculate the velocity of $\overrightarrow{X_i}$ using Eq. (12).
Generate two random numbers r, r_1 between 0 and 1.
If $r > r_1$ /* Update position of the planet*/
Update $\overrightarrow{X_i}$ using Eq. (23).
Else /***update the distance between the planet and the sun***/
Update $\overrightarrow{X_i}$ using Eq. (24).
End if
Applying one of the TFs modeled in table 1 to normalize $\overrightarrow{X_i}$
Generate $\overrightarrow{X}_{bin}(\overrightarrow{X})$ from the normalized $\overrightarrow{X_i}$ using Eq. (31)
if $f\left(\overrightarrow{X}_{bin}\right) > 0$
$\overrightarrow{X}_{bin}, C =$ Applying Algorithm 2 to further improve $\overrightarrow{X}_{bin}$
$t = t + C$ %% Increase the current function evaluation
end
Evaluate $\overrightarrow{X}_{bin}(\overrightarrow{X})$ using Eq. (30)
Updating the best-so-far solution ($\overrightarrow{X_S}$) if there is better
Apply Eq.(28)
$t = t + 1$ %% Increase the current function evaluation
End for
End while
Stop

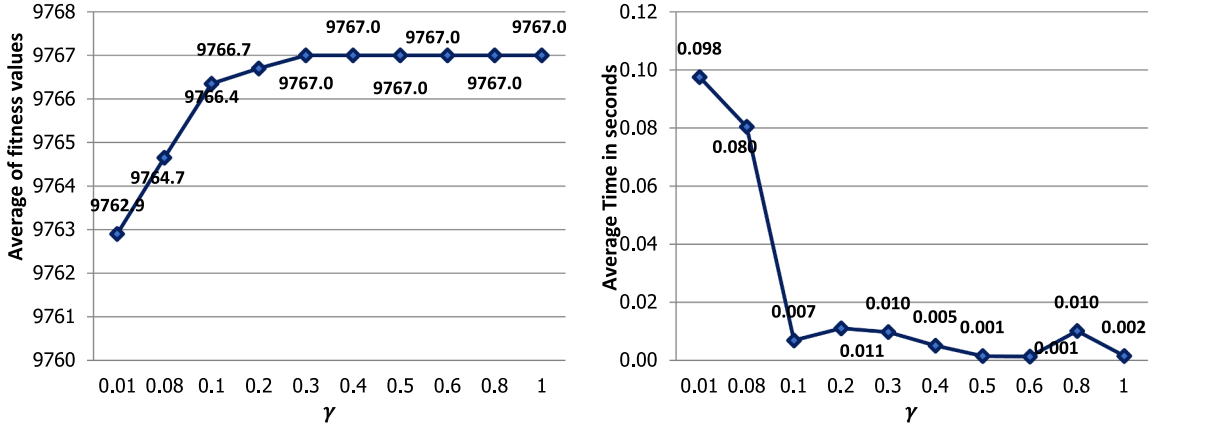
4. Results and discussion

In this section, the proposed BKOA is first assessed over various S-shaped and V-shaped TFs to pick the most effective one. Then, BKOA with the most effective TF is compared to several classical metaheuristic

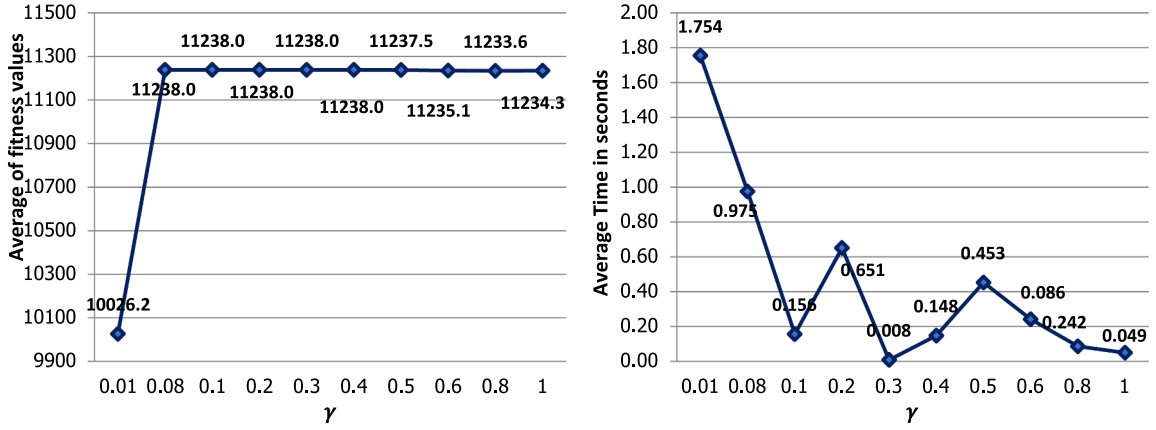


a) medium-dimensional instance: KP8 b) high-dimensional instance: KP21

Fig. 5. Tuning the parameter β.



a) Average fitness under medium-dimensional instance: KP8 b) Average time under medium-dimensional instance: KP8



a) Average fitness under high-dimensional instance: KP8 b) Average time under high-dimensional instance: KP8

Fig. 6. Tuning the parameter γ.

algorithms, such as BEO [9], BMPA [7], GA [35], BMRFO [36], BTLBO [37], BJA [38], GA under the tournament selection (GAT) [35], and BYDSE [39], to evaluate its performance in terms of best, average (Ave), and worst fitness values, in addition to the standard deviation (SD), Friedman mean rank (F-rank), and computational cost (Time). Those metrics have been computed after executing all algorithms 20 independent times on a device equipped with the following capabilities:

- Intel(R) Core(TM) i7-4700MQ processor running at 2.40 GHz,
- 32 GB of RAM,
- a 64-bit version of Windows 10 Pro
- MATLAB R2019a,

The controlling parameters of the rival algorithms are assigned in our experiments as recommended in the cited reference. All algorithms are assessed using 32 well-known instances with a number of dimensions (d) ranging between 4 and 1000 to cover small-, medium-, and large-scale [16,40]. Table 4 describes these instances in terms of the number of

Table 5
Comparison of BKOA with various V-shaped and S-shaped TF over the instances—(KP1–KP20).

Name		S1	S2	S3	S4	V1	V2	V3	V4
KP1	Ave	**295.0000**	**295.0000**	**295.0000**	**295.0000**	**295.0000**	**295.0000**	**295.0000**	**295.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
KP2	Ave	**1024.0000**	**1024.0000**	**1024.0000**	**1024.0000**	**1023.7000**	**1024.0000**	**1024.0000**	**1023.7000**
	F-rank	**4.4500**	**4.4500**	**4.4500**	**4.4500**	**4.6500**	**4.4500**	**4.4500**	**4.6500**
KP3	Ave	**35.0000**	**35.0000**	**35.0000**	**35.0000**	**35.0000**	**35.0000**	**35.0000**	**35.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
KP4	Ave	**23.0000**	**23.0000**	**23.0000**	**23.0000**	**23.0000**	**23.0000**	**23.0000**	**23.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
KP5	Ave	**481.0694**	**481.0694**	**481.0694**	**481.0694**	**481.0694**	**481.0694**	**481.0694**	**481.0694**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
KP6	Ave	**52.0000**	**52.0000**	**52.0000**	**52.0000**	**52.0000**	**52.0000**	**52.0000**	**52.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
KP7	Ave	**107.0000**	**107.0000**	**107.0000**	**107.0000**	**107.0000**	**107.0000**	**107.0000**	**107.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
KP8	Ave	**9767.0000**	**9767.0000**	**9767.0000**	**9767.0000**	9760.9000	9762.7500	9762.7500	9764.3500
	F-rank	**2.9500**	**2.9500**	**2.9500**	**2.9500**	6.9500	5.7500	6.1250	5.3750
KP9	Ave	**130.0000**	**130.0000**	**130.0000**	**130.0000**	**130.0000**	**130.0000**	**130.0000**	**130.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
KP10	Ave	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
KP11	Ave	1436.3500	1436.0500	**1437.0000**	1436.3500	1425.0500	1424.6500	1416.9500	1423.1000
	F-rank	3.1000	3.2750	**2.9250**	3.1500	5.1500	5.7250	6.7250	5.9500
KP12	Ave	**1689.0000**	1688.6000	1688.7500	1688.6000	1679.4000	1679.5500	1682.5000	1683.8000
	F-rank	**3.2500**	3.5750	3.4500	3.5500	5.9750	5.7750	5.0250	5.4000
KP13	Ave	1819.0500	1820.2000	**1820.4000**	1820.3500	1788.4000	1794.1500	1791.8000	1799.2500
	F-rank	3.1000	2.4750	**2.4750**	2.6750	6.5500	6.6750	6.4750	5.5750
KP14	Ave	**2030.3000**	2030.1000	2029.4500	2028.6000	1979.3500	1998.0000	1976.9000	1985.7500
	F-rank	**2.6750**	2.4750	2.6500	2.7500	6.7000	5.7250	6.8250	6.2000
KP15	Ave	2440.6000	2440.4500	**2440.9500**	2440.8000	2386.4500	2394.7000	2393.0500	2409.1000
	F-rank	2.6500	2.7500	**2.3750**	2.7750	6.7250	6.4000	6.7500	5.5750
KP16	Ave	2639.8500	**2642.9000**	2639.4500	2638.8500	2550.5500	2570.9000	2557.8000	2565.9000
	F-rank	2.5500	**1.8500**	2.9750	2.6500	7.0250	6.3000	6.4000	6.2500
KP17	Ave	**2912.2000**	2910.6500	2911.3000	2911.3500	2821.1000	2841.7000	2815.0000	2831.2500
	F-rank	**2.3500**	2.8250	2.6750	2.3500	6.7500	6.0000	6.7000	6.3500
KP18	Ave	2811.0500	**2813.9000**	2813.2500	2811.8500	2705.7000	2724.5500	2716.9500	2752.2500
	F-rank	3.1250	**2.0250**	2.1750	2.7750	6.8750	6.4750	6.8000	5.7500
KP19	Ave	3215.6000	3215.8000	**3218.0500**	3216.5000	3128.2000	3151.9500	3128.0500	3151.4500
	F-rank	2.7750	2.5250	**2.2500**	2.4500	7.1000	5.9000	6.8500	6.1500
KP20	Ave	3600.7500	**3602.4500**	3600.5500	3599.6000	3424.2000	3469.9000	3442.1000	3465.8000
	F-rank	2.4750	**2.2750**	2.6750	2.5750	7.1500	6.1500	6.5500	6.1500

Bold value indicates the best result.

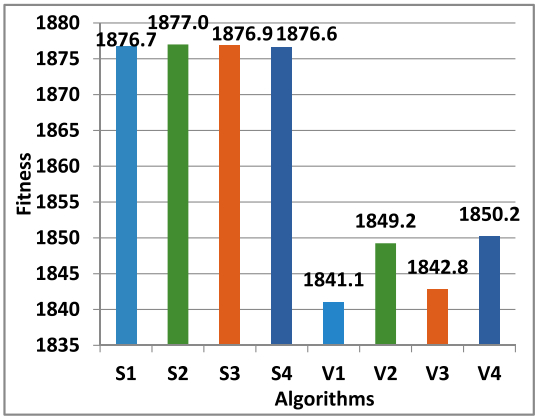


Fig. 7. Average fitness value for BKOA under each TF.

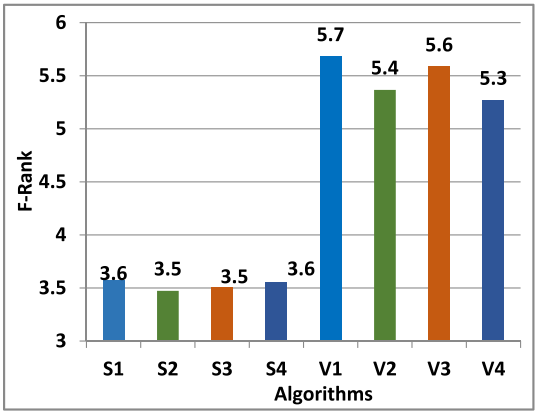


Fig. 8. Average F-rank for BKOA under each TF.

dimensions (d), the desired optimal solution (*Opt*), the instance name (*IN*), and the knapsack capacity (*Capacity*). Regarding the termination condition for each algorithm, it is unified by setting the maximum number of function evaluations to $5000 \times d$ if d is smaller than or equal to 100; otherwise, it is set to $500 \times d$ to make a fair comparison. The population size for all algorithms is set to 100.

The EIS has two additional controlling parameters (γ and β) that have to be accurately estimated to maximize its performance when integrating with HBKOA. Therefore, several experiments have been conducted under different values for each parameter to pick the most effective value. Those experiments include testing the influence of those values on medium and high-dimensional datasets. Fig. 5 illustrates the average fitness value obtained by HBKOA under various values for β. This figure discloses that HBKOA could achieve better outcomes for medium-dimensional instances and high-dimensional instances when β is set to 0.5 and 0.1, respectively. Fig. 6 presents the average fitness

Table 6
Comparison of HBKOA with various V-shaped and S-shaped TF—(KP1–KP20).

Name		S1	S2	S3	S4	V1	V2	V3	V4
KP1	Ave	**295.0000**	**295.0000**	**295.0000**	**295.0000**	**295.0000**	**295.0000**	**295.0000**	**295.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	0.0043	0.0056	0.0019	**0.0002**	0.0004	0.0023	0.0026	0.0023
KP2	Ave	**1024.0000**	**1024.0000**	**1024.0000**	**1024.0000**	**1024.0000**	**1024.0000**	**1024.0000**	**1024.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**0.0018**	0.0057	0.0223	0.0261	0.0271	0.0042	0.0038	0.0188
KP3	Ave	**35.0000**	**35.0000**	**35.0000**	**35.0000**	**35.0000**	**35.0000**	**35.0000**	**35.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**1.003E−04**	1.283E−04	1.377E−04	1.546E−04	1.116E−04	1.557E−04	1.162E−04	1.649E−04
KP4	Ave	**23.0000**	**23.0000**	**23.0000**	**23.0000**	**23.0000**	**23.0000**	**23.0000**	**23.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	1.550E−04	1.053E−04	**1.024E−04**	1.438E−04	1.863E−04	1.082E−04	1.253E−04	1.322E−04
KP5	Ave	**481.0694**	**481.0694**	**481.0694**	**481.0694**	**481.0694**	**481.0694**	**481.0694**	**481.0694**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**9.425E−04**	3.009E−03	2.252E−03	2.971E−03	3.169E−03	2.456E−03	3.643E−03	3.031E−03
KP6	Ave	**52.0000**	**52.0000**	**52.0000**	**52.0000**	**52.0000**	**52.0000**	**52.0000**	**52.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**1.250E−04**	2.854E−04	2.015E−03	5.617E−03	1.428E−04	2.587E−04	1.976E−03	2.159E−03
KP7	Ave	**107.0000**	**107.0000**	**107.0000**	**107.0000**	**107.0000**	**107.0000**	**107.0000**	**107.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	7.832E−03	3.415E−02	1.967E−04	**1.263E−04**	5.299E−03	2.945E−03	2.513E−03	1.347E−04
KP8	Ave	**9767.0000**	**9767.0000**	**9767.0000**	**9766.9500**	**9765.7500**	**9767.0000**	**9767.0000**	**9766.7500**
	F-rank	**4.2500**	**4.2500**	**4.2500**	**4.4250**	**5.6750**	**4.2500**	**4.2500**	**4.6500**
	Time	0.0306	0.0353	0.2009	0.2672	0.1278	0.3558	1.6087	1.6899
KP9	Ave	**130.0000**	**130.0000**	**130.0000**	**130.0000**	**130.0000**	**130.0000**	**130.0000**	**130.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	1.227E−04	3.200E−04	1.379E−04	1.088E−04	1.589E−04	**1.174E−04**	2.152E−03	1.241E−04
KP10	Ave	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**	**1025.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	0.0075	**0.0033**	0.0091	0.0169	0.0160	0.0154	0.0747	0.0093
KP11	Ave	**1437.0000**	**1437.0000**	**1437.0000**	**1437.0000**	**1437.0000**	**1437.0000**	**1437.0000**	**1437.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**0.0032**	0.0056	0.0108	0.0121	0.0075	0.0068	0.0067	0.0072
KP12	Ave	**1689.0000**	**1689.0000**	**1689.0000**	**1689.0000**	**1689.0000**	**1689.0000**	**1689.0000**	**1689.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**0.0067**	0.0120	0.0128	0.0131	0.0408	0.0233	0.0096	0.0076
KP13	Ave	**1821.0000**	**1821.0000**	**1821.0000**	**1821.0000**	**1821.0000**	**1821.0000**	**1821.0000**	**1821.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**0.0026**	0.0334	0.0077	0.0293	0.0194	0.0278	0.0173	0.0288
KP14	Ave	**2033.0000**	**2033.0000**	**2033.0000**	**2033.0000**	**2033.0000**	**2033.0000**	**2033.0000**	**2033.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**0.0024**	0.0083	0.0158	0.0088	0.0106	0.0098	0.0090	0.0099
KP15	Ave	**2440.0000**	**2440.2000**	**2440.4000**	**2440.4000**	**2440.0000**	**2440.0000**	**2440.4000**	**2440.4000**
	F-rank	**4.7250**	**4.5250**	**4.3250**	**4.3250**	**4.7250**	**4.7250**	**4.3250**	**4.3250**
	Time	**0.0160**	0.0084	0.0329	0.1780	0.0114	0.0108	0.0779	0.0106
KP16	Ave	**2651.0000**	**2651.0000**	**2651.0000**	**2651.0000**	**2651.0000**	**2651.0000**	**2651.0000**	**2651.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**0.0018**	0.0091	0.0091	0.0189	0.0127	0.0117	0.0125	0.0120
KP17	Ave	**2917.0000**	**2917.0000**	**2917.0000**	**2917.0000**	**2917.0000**	**2917.0000**	**2917.0000**	**2917.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**0.0020**	0.0108	0.0099	0.0092	0.0148	0.0122	0.0135	0.0125
KP18	Ave	**2817.0000**	**2817.0000**	**2817.0000**	**2817.0000**	**2817.0000**	**2817.0000**	**2817.0000**	**2817.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**0.0023**	0.0112	0.0106	0.0116	0.0140	0.0135	0.0137	0.0142
KP19	Ave	**3223.0000**	**3223.0000**	**3223.0000**	**3223.0000**	**3223.0000**	**3223.0000**	**3223.0000**	**3223.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	0.0323	0.0209	0.0728	0.0684	0.5665	**0.0104**	0.0566	2.5894
KP20	Ave	**3614.0000**	**3614.0000**	**3614.0000**	**3614.0000**	**3614.0000**	**3614.0000**	**3614.0000**	**3614.0000**
	F-rank	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**	**4.5000**
	Time	**0.0017**	0.0093	0.0095	0.0125	0.0120	0.0116	0.0142	0.0124

Bold value indicates the best result.

value and computational cost in seconds obtained by HBKOA For tuning the parameter γ. This figure shows that HBKOA has the same fitness value under various values for this parameter. Therefore, the average computational cost is computed to reveal its convergence speed under these values. In a nutshell, this figure discloses that HBKOA performs better for medium-dimensional instances and high-dimensional instances when γ is set to 0.6 and 0.3, respectively.

4.1. Experiment 1: Comparison of V-shaped and S-shaped transfer functions

In this section, the performance of various TFs with BKOA is investigated to select the most effective one that could maximize the performance of BKOA in solving various KP01 instances. Various variants of BKOA under various TFs are executed 20 times separately, and the outcomes of Ave and F-rank metrics are computed and presented in Table 5. This table discloses that all TFs have the same performance for instances with a number of dimensions smaller than 15; otherwise, the S-

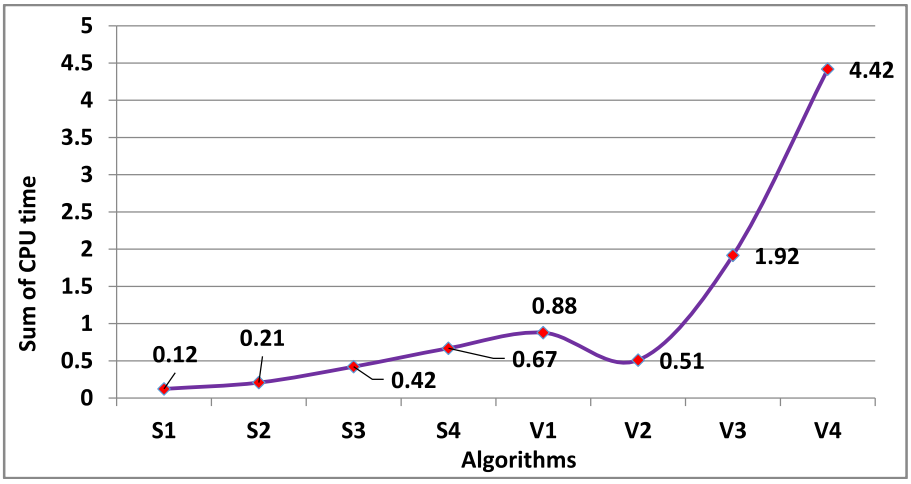


Fig. 9. Average computational cost for HBKOA under each TF.

Table 7
Comparison of integrating EIS with BKOA for feasible and infeasible solutions.

	KP11						KP12					
	Worst	Ave	Best	SD	Time	F-rank	Worst	Ave	Best	SD	Time	F-rank
HBKOA1	**1437.000**	**1437.000**	**1437.000**	**0.000**	2.38E−03	**1.500**	1684.000	1686.050	**1689.000**	2.328	8.42E−01	1.825
HBKOA	**1437.000**	**1437.000**	**1437.000**	**0.000**	**1.51E−03**	**1.500**	**1689.000**	**1689.000**	**1689.000**	**0.000**	**5.58E−03**	**1.175**
	KP13						**KP14**					
HBKOA1	1817.000	1817.400	1821.000	1.231	7.95E−01	1.950	**2033.000**	**2033.000**	**2033.000**	**0.000**	3.54E−03	1.500
HBKOA	**1821.000**	**1821.000**	**1821.000**	**0.000**	**4.44E−03**	**1.050**	**2033.000**	**2033.000**	**2033.000**	**0.000**	**3.40E−03**	**1.500**
	KP15						**KP16**					
HBKOA1	2438.000	2438.100	2440.000	4.47E−01	1.28E+00	1.975	**2651.000**	**2651.000**	**2651.000**	**0.00E+00**	**2.51E−03**	**1.500**
HBKOA	**2440.000**	**2440.200**	**2444.000**	**8.94E−01**	**2.45E−03**	**1.025**	**2651.000**	**2651.000**	**2651.000**	**0.00E+00**	2.83E−03	**1.500**
	KP17						**KP18**					
HBKOA1	**2917.000**	**2917.000**	**2917.000**	**0.00E+00**	2.69E−03	**1.500**	**2817.000**	**2817.000**	**2817.000**	**0.00E+00**	**3.19E−03**	**1.500**
HBKOA	**2917.000**	**2917.000**	**2917.000**	**0.00E+00**	**1.88E−03**	**1.500**	**2817.000**	**2817.000**	**2817.000**	**0.00E+00**	3.51E−03	**1.500**
	KP19						**KP20**					
HBKOA1	3220.000	3220.200	3223.000	6.96E−01	1.28E+00	1.975	**3614.000**	**3614.000**	**3614.000**	**0.00E+00**	**2.66E−03**	**1.500**
HBKOA	**3223.000**	**3223.000**	**3223.000**	**0.00E+00**	**1.07E−02**	**1.025**	**3614.000**	**3614.000**	**3614.000**	**0.00E+00**	4.06E−03	**1.500**
	KP1_200						**KP1_500**					
HBKOA1	11186.00	11224.40	**11238.00**	1.64E+01	1.60E−01	1.700	27353.00	27688.85	27977.00	**1.43E+02**	3.34E−01	2.000
HBKOA	**11186.00**	**11229.20**	**11238.00**	**1.60E+01**	**3.29E−02**	**1.300**	**27878.00**	**28234.15**	**28637.00**	1.73E+02	**1.44E+00**	**1.000**

Bold value indicates the best outcome.

shaped TFs (S1, S2, S3, and S4) could achieve better outcomes. Approximately four S-shaped TFs have the same performance as shown in Figs. 7 and 8. The little difference among them is due to the stochastic characteristic used to convert the normalized continuous values into 0 and 1. Since various S-shaped TFs are on par, S3 has been selected among them to adapt BKOA in the next experiments. Likewise, the best transfer function for HBKOA has been selected by running it with each transfer function 20 times separately, and the average fitness value and F-rank are computed and reported in Table 6. This table shows that HBKOA has the same performance as all TFs. Therefore, the computational cost under each TF is computed and presented in the same table to show which one has a lower cost. In addition, the average computational cost for each TF on all instances is reported in Fig. 9. This figure shows that S1 is the quickest, with a value of 0.12 s, while V4 is the slowest, with a value of 4.42 s. Therefore, S1 is employed with HBKOA in the next experiments.

4.2. Experiment 2: Performance evaluation of EIS with BKOA

In this section, EIS is employed with BKOA to improve the quality of the binary solutions. This strategy is integrated with the BKOA in two different ways to present two different variants: HBKOA1 and HBKOA. HBKOA1 uses EIS to improve the unfeasible solutions, while HBKOA uses this strategy to enhance the feasible solutions. To pick the best-performing variant, both of them are executed 20 independent times in some instances, and various performance indicators are computed and reported in Table 7. Inspecting this table shows that the second variant that applies EIS to improve the quality of feasible solutions is either better or on par for all validated instances. In the next experiments, to further observe the effectiveness of EIS, it is integrated with some other binary metaheuristic algorithms to show its influence on their performance.

4.3. Experiment 3: Comparison between BKOA and some classical metaheuristics (KP1-KP20)

In this section, BKOA and eight binary competitors are employed to solve the instances ranging between KP1 and KP20. All are independently executed 20 times, and the outcomes of various performance metrics are reported in Tables 8 and 9. Those tables show that HKOA is competitive with some algorithms, like GA, BMRFO, BEO, and GAT, for the instances ranging between KP1 and KP10. For the instances between KP11 and KP20, HKOA could come in the second rank after GA for 8 instances with $d \geq 40$, and in the third rank after GA and BMRFO for KP11, while it is competitive with GA and BMRFO for KP11. To better show the effectiveness of HKOA over the other optimizers, the average

Table 8
Comparison among algorithms over the instances from KP1 and KP10.

Name		BKOA	BEO	BMRFO	BTLBO	GAT	BMPA	GA	BJA	BYDSE
KP1	Worst	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	279.000	**295.000**
	Ave	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	293.300	**295.000**
	Best	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	4.156	**0.000**
	Time	1.71E−02	**5.59E−04**	1.43E−03	6.88E−03	4.30E−03	1.16E−04	6.99E−03	2.89E−02	4.74E−03
	F-rank	**4.875**	**4.875**	**4.875**	**4.875**	**4.875**	**4.875**	**4.875**	6.000	**4.875**
KP2	Worst	**1024.000**	**1024.000**	**1024.000**	1018.000	**1024.000**	993.000	**1024.000**	**1024.000**	985.000
	Ave	**1024.000**	**1024.000**	**1024.000**	1023.700	**1024.000**	1017.750	**1024.000**	**1024.000**	1006.500
	Best	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**
	SD	**0.000**	**0.000**	**0.000**	1.342	**0.000**	9.596	**0.000**	**0.000**	13.671
	Time	3.02E−02	6.49E−03	1.33E−02	1.59E−01	3.64E−02	1.27E+00	2.78E−02	**4.53E−04**	2.71E−01
	F−rank	**4.350**	**4.350**	**4.350**	4.550	**4.350**	6.400	**4.350**	**4.350**	7.950
KP3	Worst	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**
	Ave	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**
	Best	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	Time	1.45E−04	2.04E−04	1.61E−04	1.55E−04	1.89E−04	1.64E−04	8.64E−04	**1.05E−04**	1.60E−04
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
KP4	Worst	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**
	Ave	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**
	Best	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	Time	2.37E−04	4.09E−04	1.60E−04	1.19E−04	1.11E−04	1.47E−04	1.12E−03	1.10E−04	**9.47E−05**
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
KP5	Worst	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**	437.935	**481.069**	360.707	**481.069**
	Ave	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**	476.710	**481.069**	461.355	**481.069**
	Best	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**
	SD	**2.33E−13**	**2.33E−13**	**2.33E−13**	**2.33E−13**	**2.33E−13**	1.18E+01	**2.33E−13**	3.80E+01	**2.33E−13**
	Time	2.97E−02	6.91E−03	**7.67E−04**	1.11E−03	2.53E−03	1.07E+00	5.96E−03	9.84E−02	1.81E−02
	F-rank	**4.750**	**4.750**	**4.750**	**4.750**	**4.750**	5.600	**4.750**	6.150	**4.750**
KP6	Worst	**52.000**	**52.000**	**52.000**	**52.000**	51.000	**52.000**	**52.000**	**52.000**	**52.000**
	Ave	**52.000**	**52.000**	**52.000**	**52.000**	51.950	**52.000**	**52.000**	**52.000**	**52.000**
	Best	**52.000**	**52.000**	**52.000**	**52.000**	52.000	**52.000**	**52.000**	**52.000**	**52.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.24E−01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Time	2.16E−02	1.17E−03	5.89E−04	7.15E−04	9.86E−04	1.05E−03	1.60E−03	**1.23E−04**	7.89E−04
	F-rank	**4.975**	**4.975**	**4.975**	**4.975**	5.200	**4.975**	**4.975**	**4.975**	**4.975**
KP7	Worst	**107.000**	**107.000**	**107.000**	**107.000**	105.000	105.000	**107.000**	105.000	**107.000**
	Ave	**107.000**	**107.000**	**107.000**	**107.000**	106.400	106.900	**107.000**	106.400	**107.000**
	Best	**107.000**	**107.000**	**107.000**	**107.000**	**107.000**	**107.000**	**107.000**	**107.000**	**107.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.40E−01	4.47E−01	**0.00E+00**	9.40E−01	**0.00E+00**
	Time	6.09E−04	8.86E−04	5.11E−04	1.30E−03	1.86E−03	4.02E−01	9.82E−04	1.15E−04	**1.24E−04**
	F-rank	**4.675**	**4.675**	**4.675**	**4.675**	6.025	4.900	**4.675**	6.025	**4.675**
KP8	Worst	**9767.000**	**9767.000**	**9767.000**	9759.000	9762.000	9740.000	**9767.000**	9703.000	9762.000
	Ave	**9767.000**	**9767.000**	**9767.000**	9763.200	9766.150	9757.000	**9767.000**	9731.550	9764.400
	Best	**9767.000**	**9767.000**	**9767.000**	**9767.000**	**9767.000**	**9767.000**	**9767.000**	9751.000	**9767.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.24E+00	1.84E+00	6.62E+00	**0.00E+00**	1.20E+01	1.64E+00
	Time	6.63E−01	2.54E−02	**3.78E−03**	2.70E−01	7.47E−01	1.84E+00	2.74E−02	1.81E−01	4.87E−01
	F-rank	**3.075**	**3.075**	**3.075**	6.350	3.825	7.625	**3.075**	9.000	5.900
KP9	Worst	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**
	Ave	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**
	Best	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Time	1.70E−04	3.06E−04	2.29E−04	1.82E−04	1.86E−04	1.08E−04	1.20E−03	1.97E−04	**1.35E−04**
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
KP10	Worst	**1025.000**	**1025.000**	**1025.000**	1017.000	**1025.000**	1005.000	**1025.000**	**1025.000**	962.000
	Ave	**1025.000**	**1025.000**	**1025.000**	1023.700	**1025.000**	1021.550	**1025.000**	**1025.000**	1007.200
	Best	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.70E+00	**0.00E+00**	5.27E+00	**0.00E+00**	**0.00E+00**	1.57E+01
	Time	7.23E−02	4.61E−03	**2.72E−03**	1.07E−01	2.22E−03	1.57E−02	3.77E−03	1.31E−03	1.94E−01
	F-rank	**4.225**	**4.225**	**4.225**	5.100	**4.225**	5.900	**4.225**	**4.225**	8.650

Bold value indicates the best outcome.

Table 9
Comparison among algorithms over the instances from KP11 and KP20.

Name		BKOA	BEO	BMRFO	BTLBO	GAT	BMPA	GA	BJA	BYDSE
KP11	Worst	1424.000	1414.000	**1437.000**	1390.000	1415.000	1351.000	**1437.000**	1375.000	1304.000
	Ave	1436.350	1429.550	**1437.000**	1411.500	1433.000	1401.900	**1437.000**	1431.250	1349.200
	Best	**1437.000**	**1437.000**	**1437.000**	**1437.000**	**1437.000**	1431.000	**1437.000**	**1437.000**	1411.000
	SD	2.91E+00	7.11E+00	**0.00E+00**	1.65E+01	6.79E+00	2.05E+01	**0.00E+00**	1.55E+01	2.46E+01
	Time	7.90E−01	2.21E−01	1.47E−02	3.17E−01	4.06E−02	1.01E+00	**3.95E−03**	3.53E−02	2.12E−01
	F-rank	3.175	4.925	**3.000**	6.500	4.000	7.650	**3.000**	3.850	8.900
KP12	Worst	**1689.000**	1657.000	**1689.000**	1636.000	1679.000	1601.000	**1689.000**	1596.000	1497.000
	Ave	**1689.000**	1677.700	**1689.000**	1669.800	1688.500	1643.100	**1689.000**	1684.350	1535.450
	Best	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	1586.000
	SD	**0.00E+00**	8.49E+00	**0.00E+00**	1.88E+01	2.24E+00	2.30E+01	**0.00E+00**	2.08E+01	2.89E+01
	Time	2.56E−01	2.16E−01	**3.27E−03**	2.34E−01	1.92E−01	1.25E+00	4.45E−03	4.56E−02	2.87E−01
	F-rank	**3.175**	5.975	**3.175**	5.925	3.400	7.575	**3.175**	3.475	9.000
KP13	Worst	1817.000	1767.000	1816.000	1716.000	1816.000	1662.000	**1821.000**	1734.000	1575.000
	Ave	1820.400	1792.950	1820.300	1786.100	1818.950	1744.200	**1821.000**	1802.800	1636.150
	Best	**1821.000**	1817.000	**1821.000**	**1821.000**	**1821.000**	1784.000	**1821.000**	**1821.000**	1701.000
	SD	1.47E+00	1.30E+01	1.72E+00	3.67E+01	2.35E+00	3.12E+01	**0.00E+00**	3.32E+01	3.72E+01
	Time	7.74E−01	2.91E−01	**1.57E−02**	3.45E−01	7.07E−02	1.36E+00	**1.68E−02**	1.94E−01	2.97E−01
	F-rank	3.050	6.200	3.100	5.675	3.900	7.550	**2.675**	3.850	9.000
KP14	Worst	2020.000	1945.000	2003.000	1921.000	2016.000	1844.000	**2033.000**	1999.000	1703.000
	Ave	2032.350	1967.750	2027.400	1996.150	2028.900	1926.250	**2033.000**	2029.150	1769.400
	Best	2033.000	2002.000	2033.000	2033.000	2033.000	1985.000	**2033.000**	2033.000	1839.000
	SD	2.907	17.979	8.586	34.414	6.480	40.106	**0.000**	8.845	32.629
	Time	0.882	0.349	0.166	0.428	0.055	1.684	**0.006**	0.144	0.367
	F-rank	2.750	7.100	3.525	5.750	3.525	7.550	**2.650**	3.150	9.000
KP15	Worst	2438.000	2332.000	2438.000	2292.000	2440.000	2217.000	**2440.000**	2303.000	2021.000
	Ave	2440.900	2361.950	2440.800	2407.400	2440.550	2307.500	**2441.950**	2419.350	2095.500
	Best	**2444.000**	**2400.000**	**2444.000**	**2444.000**	**2444.000**	**2429.000**	**2444.000**	**2444.000**	2206.000
	SD	1.971	18.625	1.989	46.140	1.356	61.732	**2.012**	41.940	45.333
	Time	0.610	0.405	0.194	0.534	0.027	2.442	0.714	**0.163**	0.546
	F-rank	3.175	6.850	3.300	5.375	3.325	7.700	**2.450**	3.825	9.000
KP16	Worst	2630.000	2499.000	2627.000	2461.000	2631.000	2366.000	**2643.000**	1985.000	2192.000
	Ave	2640.600	2536.150	2639.700	2592.900	2639.850	2475.900	**2647.400**	2500.600	2289.300
	Best	**2651.000**	2597.000	**2651.000**	**2651.000**	**2651.000**	2593.000	**2651.000**	**2651.000**	2382.000
	SD	5.844	24.669	8.163	61.163	5.806	52.919	**4.083**	186.777	44.517
	Time	2.158	0.539	0.127	0.535	0.947	2.843	**0.015**	0.527	1.037
	F-rank	3.500	6.400	3.100	5.100	3.500	7.350	**1.775**	5.375	8.900
KP17	Worst	2896.000	2732.000	2865.000	2814.000	2901.000	2655.000	**2917.000**	2094.000	2322.000
	Ave	2914.900	2771.850	2909.800	2881.800	2916.200	2719.150	**2917.000**	2735.550	2410.900
	Best	**2917.000**	**2839.000**	**2917.000**	**2917.000**	**2917.000**	2813.000	**2917.000**	**2917.000**	2544.000
	SD	5.515	29.391	13.117	24.867	3.578	50.058	**0.000**	238.253	61.915
	Time	2.127	0.418	0.072	0.549	0.090	2.911	**0.138**	0.352	0.645
	F-rank	2.775	6.800	3.275	5.125	2.550	7.400	**2.425**	5.750	8.900
KP18	Worst	2809.000	2645.000	2790.000	2636.000	2814.000	2521.000	**2816.000**	1861.000	2269.000
	Ave	2813.900	2677.400	2811.750	2778.850	2816.200	2648.250	**2817.000**	2610.350	2341.000
	Best	**2818.000**	2732.000	**2818.000**	2814.000	**2818.000**	2733.000	**2818.000**	**2817.000**	2456.000
	SD	2.751	24.727	7.489	46.526	1.105	52.041	**0.459**	289.466	45.990
	Time	2.232	0.689	0.255	0.704	0.053	2.487	**0.011**	0.454	0.819
	F-rank	3.325	6.850	3.400	5.275	2.375	7.200	**1.900**	5.825	8.850
KP19	Worst	3197.000	3031.000	3205.000	3019.000	3208.000	2903.000	**3220.000**	2225.000	2539.000
	Ave	3215.450	3076.950	3215.600	3177.550	3219.750	3025.000	**3222.050**	3008.100	2645.850
	Best	3221.000	3132.000	3223.000	3217.000	3223.000	3135.000	**3223.000**	3223.000	2919.000
	SD	5.356	22.477	4.806	44.926	3.226	54.265	**1.099**	348.715	83.817
	Time	2.372	0.550	**0.237**	0.662	1.900	2.588	0.811	0.718	0.723
	F-rank	3.800	6.700	3.550	5.550	2.525	7.600	**1.400**	5.075	8.800
KP20	Worst	3590.000	3366.000	3587.000	3433.000	3597.000	3214.000	**3601.000**	2376.000	2818.000
	Ave	3599.500	3404.100	3599.950	3559.400	3606.400	3319.200	**3611.700**	3285.800	2946.200
	Best	3609.000	3509.000	3609.000	3614.000	3614.000	3397.000	**3614.000**	3609.000	3039.000
	SD	6.039	35.993	5.365	49.849	5.154	54.955	**3.420**	373.717	58.017
	Time	4.382	0.931	0.353	0.950	**0.027**	3.170	0.119	0.591	0.697
	F-rank	3.775	6.500	3.550	4.725	2.325	7.450	**1.325**	6.500	8.850

Bold value indicates the best outcome.

of their F-rank values is computed and presented in Fig. 10, which discloses the effectiveness of HKOA over the majority of the classical metaheuristic algorithms and its competitiveness with GA. From that, we conclude that BKOA could be classified as a high-performing optimizer for the KP01 problem since it could be superior to the compared classical metaheuristics and competitive with GA for the majority of the used instances. To make BKOA better than GA, we presented a new variant called HBKOA based on integrating BKOA with the EIS strategy. The results of HBKOA are discussed in the next section to elaborate on its effectiveness over GA.

4.4. Experiment 5: Comparison between HBKOA and GA

This section compares the outcomes of HBKOA to those of GA, as reported in Table 10. This table exposes that HBKOA and GA are competitive in terms of various performance metrics for the instances from KP1 to KP15, in addition to KP17, with the exception of time. In these instances, the proposed HBKOA could outperform in terms of the computational cost for the majority of test functions. This shows that it could converge to the near-optimal solution faster than GA. For the other instances, KP16, KP18, KP19, and KP20, HBKOA could be competitive

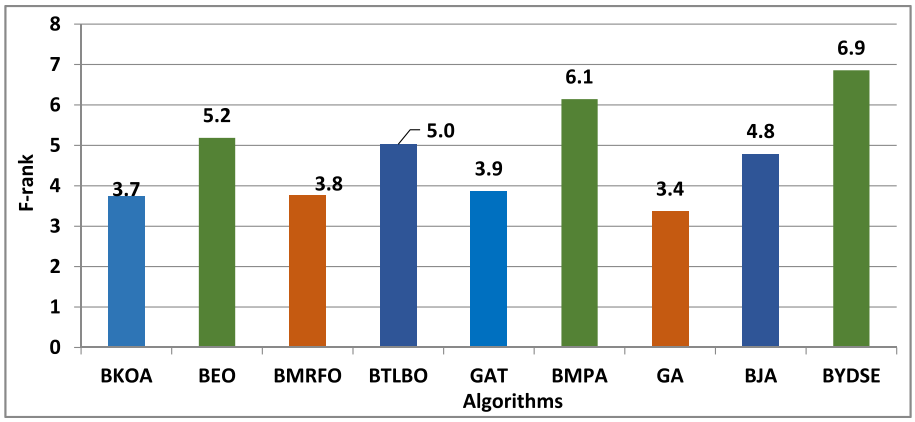


Fig. 10. Average F-rank of BKOA and various competitors.

Table 10
Comparison between HBKOA and GA over the instances from KP1 and KP20.

Name		BKOA	GA	Name		BKOA	GA	Name		BKOA	GA
KP1	Worst	**295.000**	**295.000**	**KP8**	Worst	**9767.000**	**9767.000**	**KP15**	Worst	**2440.000**	**2440.000**
	Ave	**295.000**	**295.000**		Ave	**9767.000**	**9767.000**		Ave	**2442.200**	**2441.950**
	Best	**295.000**	**295.000**		Best	**9767.000**	**9767.000**		Best	**2444.000**	**2444.000**
	SD	**0.000**	**0.000**		SD	**0.00E+00**	**0.00E+00**		SD	**0.894**	2.012
	Time	7.89E−03	**6.99E−03**		Time	**1.77E−02**	2.74E−02		Time	**0.004**	0.714
KP2	Worst	**1024.000**	**1024.000**	**KP9**	Worst	**130.000**	**130.000**	**KP16**	Worst	**2651.000**	2643.000
	Ave	**1024.000**	**1024.000**		Ave	**130.000**	**130.000**		Ave	**2651.000**	2647.400
	Best	**1024.000**	**1024.000**		Best	**130.000**	**130.000**		Best	**2651.000**	**2651.000**
	SD	**0.000**	**0.000**		SD	**0.00E+00**	**0.00E+00**		SD	**0.00E+00**	4.083
	Time	**3.04E−03**	2.78E−02		Time	**1.05E−04**	1.20E−03		Time	**0.001**	0.015
KP3	Worst	**35.000**	**35.000**	**KP10**	Worst	**1025.000**	**1025.000**	**KP17**	Worst	**2917.000**	**2917.000**
	Ave	**35.000**	**35.000**		Ave	**1025.000**	**1025.000**		Ave	**2917.000**	**2917.000**
	Best	**35.000**	**35.000**		Best	**1025.000**	**1025.000**		Best	**2917.000**	**2917.000**
	SD	**0.000**	**0.000**		SD	**0.00E+00**	**0.00E+00**		SD	**0.00E+00**	**0.00E+00**
	Time	**1.18E−04**	8.64E−04		Time	**1.90E−03**	3.77E−03		Time	**0.001**	0.138
KP4	Worst	**23.000**	**23.000**	**KP11**	Worst	**1437.000**	**1437.000**	**KP18**	Worst	**2817.000**	2816.000
	Ave	**23.000**	**23.000**		Ave	**1437.000**	**1437.000**		Ave	**2817.000**	2816.500
	Best	**23.000**	**23.000**		Best	**1437.000**	**1437.000**		Best	**2817.000**	**2817.000**
	SD	**0.000**	**0.000**		SD	**0.00E+00**	**0.00E+00**		SD	**0.00E+00**	0.459
	Time	**9.17E−05**	1.12E−03		Time	**2.15E−03**	3.95E−03		Time	**0.002**	**0.006**
KP5	Worst	**481.069**	**481.069**	**KP12**	Worst	**1689.000**	**1689.000**	**KP19**	Worst	**3223.000**	3220.000
	Ave	**481.069**	**481.069**		Ave	**1689.000**	**1689.000**		Ave	**3223.000**	3222.050
	Best	**481.069**	**481.069**		Best	**1689.000**	**1689.000**		Best	**3223.000**	**3223.000**
	SD	**2.33E−13**	**2.33E−13**		SD	**0.00E+00**	**0.00E+00**		SD	**0.00E+00**	1.099
	Time	**1.07E−03**	5.96E−03		Time	**2.00E−03**	4.45E−03		Time	**0.006**	0.811
KP6	Worst	**52.000**	**52.000**	**KP13**	Worst	**1821.000**	**1821.000**	**KP20**	Worst	**3614.000**	3601.000
	Ave	**52.000**	**52.000**		Ave	**1821.000**	**1821.000**		Ave	**3614.000**	3611.700
	Best	**52.000**	**52.000**		Best	**1821.000**	**1821.000**		Best	**3614.000**	**3614.000**
	SD	**0.00E+00**	**0.00E+00**		SD	**0.00E+00**	**0.00E+00**		SD	**0.00E+00**	3.420
	Time	**1.08E−03**	1.60E−03		Time	**3.05E−03**	1.68E−02		Time	**0.001**	0.119
KP7	Worst	**107.000**	**107.000**	**KP14**	Worst	**2033.000**	**2033.000**				
	Ave	**107.000**	**107.000**		Ave	**2033.000**	**2033.000**				
	Best	**107.000**	**107.000**		Best	**2033.000**	**2033.000**				
	SD	**0.00E+00**	**0.00E+00**		SD	**0.00E+00**	**0.00E+00**				
	Time	2.99E−03	**9.82E−04**		Time	**2.00E−03**	**6.00E−03**				

Bold value indicates the best outcome.

for the best fitness value and superior in terms of Ave, Worst, SD, and Time.

4.5. Experiment 6: Comparison between some metaheuristics with EIS

The performance of EIS with some other metaheuristics is investigated in this section to reveal its effectiveness. EIS is integrated with BEO, BMRFO, BTLBO, BMPA, BJA, and BYDSE to present new variants named HBEO, HBMRFO, HBTLBO, HBMPA, HBJA, and HBYDSE. All those hybrid algorithms are separately executed 20 times, and their results are analyzed in terms of six performance metrics, as reported in Tables 11 and 12. Those tables disclose that this strategy could improve the performance of the hybrid algorithms, especially HBKOA, HBEO, and HBMRFO, which could achieve outstanding outcomes for all instances. This is affirmed in Fig. 11, which shows that HBEO comes in first with an average F-rank value of 4.51, followed by both HBKOA and HBMRFO with values of 4.52 and 4.53, respectively, while HBJA is the worst. Fig. 12 is presented to reveal the superiority of the hybrid binary algorithms over the classical binary algorithms. This figure shows that the classical binary algorithms are inferior to their hybrid variants.

Table 11
Comparison among hybrid metaheuristics over the instances from KP1 and KP10.

Name		HBKOA	HBEO	HBMRFO	HBTLBO	HBMPA	HBJA	HBYDSE
KP1	Worst	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	294.000	**295.000**
	Ave	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	294.800	**295.000**
	Best	**295.000**	**295.000**	**295.000**	**295.000**	**295.000**	295.000	**295.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.410	**0.000**
	Time	7.89E−03	5.04E−04	1.08E−03	7.91E−04	6.44E−04	1.75E−03	**5.09E−04**
	F-rank	**4.900**	**4.900**	**4.900**	**4.900**	**4.900**	5.800	**4.900**
KP2	Worst	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**
	Ave	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**
	Best	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**	**1024.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	Time	3.04E−03	1.16E−03	9.02E−04	1.01E−03	1.62E−03	1.39E−03	**9.24E−04**
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
KP3	Worst	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**
	Ave	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**
	Best	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**	**35.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	Time	1.18E−04	2.19E−04	5.77E−05	8.88E−05	9.82E−05	**5.87E−05**	9.11E−05
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
KP4	Worst	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**
	Ave	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**
	Best	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**	**23.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	Time	9.17E−05	2.51E−04	7.58E−05	**5.83E−05**	7.23E−05	1.11E−04	8.01E−05
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
KP5	Worst	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**	431.709	**481.069**
	Ave	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**	476.206	**481.069**
	Best	**481.069**	**481.069**	**481.069**	**481.069**	**481.069**	481.069	**481.069**
	SD	**2.33E−13**	**2.33E−13**	**2.33E−13**	**2.33E−13**	**2.33E−13**	1.50E+01	**2.33E−13**
	Time	1.07E−03	4.62E−04	5.90E−04	3.81E−04	4.31E−04	2.26E−02	**4.18E−04**
	F-rank	**4.925**	**4.925**	**4.925**	**4.925**	**4.925**	5.375	**4.925**
KP6	Worst	**52.000**	**52.000**	**52.000**	**52.000**	**52.000**	**52.000**	**52.000**
	Ave	**52.000**	**52.000**	**52.000**	**52.000**	**52.000**	**52.000**	**52.000**
	Best	**52.000**	**52.000**	**52.000**	**52.000**	**52.000**	**52.000**	**52.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Time	1.08E−03	3.19E−04	4.33E−04	4.40E−04	**7.60E−05**	2.22E−04	4.43E−04
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
KP7	Worst	**107.000**	**107.000**	**107.000**	**107.000**	**107.000**	105.000	**107.000**
	Ave	**107.000**	**107.000**	**107.000**	**107.000**	**107.000**	106.200	**107.000**
	Best	**107.000**	**107.000**	**107.000**	**107.000**	**107.000**	107.000	**107.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.01E+00	**0.00E+00**
	Time	2.99E−03	3.32E−04	1.42E−04	1.16E−03	3.25E−04	1.01E−04	**7.54E−05**
	F-rank	**4.675**	**4.675**	**4.675**	**4.675**	**4.675**	6.475	**4.675**
KP8	Worst	**9767.000**	**9767.000**	**9767.000**	9762.000	9761.000	9713.000	9757.000
	Ave	**9767.000**	**9767.000**	**9767.000**	9765.250	9765.150	9733.950	9762.050
	Best	**9767.000**	**9767.000**	**9767.000**	**9767.000**	**9767.000**	**9767.000**	**9767.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.77E+00	2.39E+00	1.43E+01	2.58E+00
	Time	1.77E−02	1.88E−02	2.19E−02	1.21E−02	2.52E−02	1.03E−01	**9.40E−03**
	F-rank	**3.500**	**3.500**	**3.500**	5.800	5.175	8.725	7.625
KP9	Worst	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**
	Ave	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**
	Best	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**	**130.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Time	1.05E−04	3.01E−04	1.10E−04	9.17E−04	8.10E−05	**6.69E−05**	1.57E−04
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
KP10	Worst	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**
	Ave	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**
	Best	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**	**1025.000**
	SD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Time	1.90E−03	1.66E−03	1.94E−03	7.67E−04	**7.44E−04**	9.76E−04	9.13E−04
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**

Bold value indicates the best outcome.

4.6. Experiment 7: Comparison under high-dimensional instances (F21-F32)

In this section, the performance of three well-performing hybrid algorithms (HBKOA, HBEO, and HBMRFO) is assessed using 12 high-dimensional instances with a number of dimensions ranging between 100 and 1000. According to the correlation between the weights and profits, those instances are classified as follows: KP21–KP24 are classified as uncorrelated instances, KP25-KP28 are classified as weakly correlated instances, and KP29–KP32 are classified as strongly correlated instances. For those instances, three hybrid algorithms are executed 20 independent times, and the outcomes of various utilized performance metrics are computed and reported in Table 13. By inspection, we found that HBMRFO is better for weakly correlated and uncorrelated instances with a number of dimensions greater than 200. All algorithms could achieve competitive outcomes for all instances with dimensions smaller than or equal to 200. For strongly correlated instances with a number of dimensions greater than 200, all algorithms are competitive in terms of the best fitness, while both HBEO and HBMRFO are better in terms of Ave, Worst, SD, and F-rank. Regarding the

Table 12
Comparison among hybrid algorithms over the instances from KP11 and KP20.

Name		HBKOA	HBEO	HBMRFO	HBTLBO	HBMPA	HBJA	HBYDSE
KP11	Worst	**1437.000**	**1437.000**	**1437.000**	**1437.000**	**1437.000**	1424.000	**1437.000**
	Ave	**1437.000**	**1437.000**	**1437.000**	**1437.000**	**1437.000**	1436.350	**1437.000**
	Best	**1437.000**	**1437.000**	**1437.000**	**1437.000**	**1437.000**	1437.000	**1437.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	2.907	**0.000**
	Time	0.002	**0.001**	**0.001**	0.005	0.003	0.007	0.001
	F-rank	**4.875**	**4.875**	**4.875**	**4.875**	**4.875**	5.075	**4.875**
KP12	Worst	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**
	Ave	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**
	Best	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**	**1689.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	Time	0.002	**0.001**	0.002	0.003	0.002	0.003	**0.001**
	F-rank	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
KP13	Worst	**1821.000**	**1821.000**	**1821.000**	**1821.000**	1817.000	1371.000	**1821.000**
	Ave	**1821.000**	**1821.000**	**1821.000**	**1821.000**	1820.000	1798.500	**1821.000**
	Best	**1821.000**	**1821.000**	**1821.000**	**1821.000**	**1821.000**	**1821.000**	**1821.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	1.777	100.623	**0.000**
	Time	0.003	**0.001**	0.003	0.031	0.580	0.016	0.005
	F-rank	**4.575**	**4.575**	**4.575**	**4.575**	5.675	4.825	**4.575**
KP14	Worst	**2033.000**	**2033.000**	**2033.000**	**2033.000**	**2033.000**	**2033.000**	**2033.000**
	Ave	**2033.000**	**2033.000**	**2033.000**	**2033.000**	**2033.000**	**2033.000**	**2033.000**
	Best	**2033.000**	**2033.000**	**2033.000**	**2033.000**	**2033.000**	**2033.000**	**2033.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	Time	0.002	0.002	**0.001**	0.002	0.109	0.016	0.003
	F-rank	4.750	4.750	4.750	4.750	4.750	4.750	4.750
KP15	Worst	**2440.000**	**2440.000**	**2440.000**	**2440.000**	**2440.000**	2438.000	**2440.000**
	Ave	**2440.200**	**2440.400**	**2440.000**	**2440.200**	**2440.400**	2440.300	**2440.200**
	Best	**2444.000**	**2444.000**	**2440.000**	**2444.000**	**2444.000**	**2444.000**	**2444.000**
	SD	0.894	1.231	**0.000**	0.894	1.231	1.342	0.894
	Time	0.004	**0.001**	**0.001**	0.023	0.068	0.010	0.001
	F-rank	5.300	5.075	5.525	5.275	5.075	5.275	5.300
KP16	Worst	**2651.000**	**2651.000**	**2651.000**	2643.000	**2651.000**	2003.000	**2651.000**
	Ave	**2651.000**	**2651.000**	**2651.000**	2650.200	**2651.000**	2617.800	**2651.000**
	Best	**2651.000**	**2651.000**	**2651.000**	**2651.000**	**2651.000**	**2651.000**	**2651.000**
	SD	**0.000**	**0.000**	**0.000**	2.093	**0.000**	144.730	**0.000**
	Time	**0.001**	0.012	0.003	0.046	0.013	0.025	**0.001**
	F-rank	**4.325**	**4.325**	**4.325**	4.925	4.325	5.050	**4.325**
KP17	Worst	**2917.000**	**2917.000**	**2917.000**	**2917.000**	**2917.000**	2012.000	**2917.000**
	Ave	**2917.000**	**2917.000**	**2917.000**	**2917.000**	**2917.000**	2869.400	**2917.000**
	Best	**2917.000**	**2917.000**	**2917.000**	**2917.000**	**2917.000**	**2917.000**	**2917.000**
	SD	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	201.943	**0.000**
	Time	**0.001**	0.005	**0.001**	0.027	0.015	0.011	0.003
	F-rank	**4.875**	**4.875**	**4.875**	**4.875**	**4.875**	5.550	**4.875**
KP18	Worst	**2817.000**	**2817.000**	**2817.000**	2817.000	2817.000	1908.000	**2817.000**
	Ave	**2817.000**	**2817.000**	**2817.000**	2817.200	2817.000	2655.400	**2817.000**
	Best	**2817.000**	**2817.000**	**2817.000**	2818.000	2817.000	2818.000	**2817.000**
	SD	**0.000**	**0.000**	**0.000**	0.410	0.000	330.771	**0.000**
	Time	**0.002**	0.004	**0.002**	0.042	0.026	0.137	**0.002**
	F-rank	**4.900**	**4.900**	**4.900**	4.025	4.900	6.050	**4.900**
KP19	Worst	**3223.000**	**3223.000**	**3223.000**	3221.000	3221.000	2129.000	**3223.000**
	Ave	**3223.000**	**3223.000**	**3223.000**	3222.800	3222.400	3018.700	**3223.000**
	Best	**3223.000**	**3223.000**	**3223.000**	**3223.000**	**3223.000**	**3223.000**	**3223.000**
	SD	**0.000**	**0.000**	**0.000**	0.616	0.940	418.793	**0.000**
	Time	**0.006**	**0.006**	**0.006**	0.051	0.037	0.035	0.033
	F-rank	**3.950**	**3.950**	**3.950**	4.350	5.200	5.925	**3.950**
KP20	Worst	**3614.000**	**3614.000**	**3614.000**	3604.000	**3614.000**	2373.000	**3614.000**
	Ave	**3614.000**	**3614.000**	**3614.000**	3612.350	**3614.000**	3336.350	**3614.000**
	Best	**3614.000**	**3614.000**	**3614.000**	**3614.000**	**3614.000**	**3614.000**	**3614.000**
	SD	**0.000**	**0.000**	**0.000**	3.438	**0.000**	486.201	**0.000**
	Time	**0.001**	0.006	0.003	0.056	0.205	0.063	0.016
	F-rank	**4.325**	**4.325**	**4.325**	5.225	4.325	6.325	**4.325**

*The bold **value indicates the best outcome**.*

computational cost, HBMRFO is better for 11 out of 12 instances, as highlighted in bold in Table 13. To sum up, HBMRFO could perform strongly in the majority of high-dimensional instances. This is due to the EIS, which could significantly improve the performance of some algorithms while having a weak influence on others. Therefore, in the future, the performance of this strategy will be investigated with some other metaheuristic algorithms to find better outcomes for KP01. Ultimately, HBMRFO is considered a strong alternative for solving high-dimensional knapsack problems since it could fulfill outstanding outcomes for the majority of high-dimensional instances.

5. Conclusion and future work

In this study, the Kepler optimization algorithm is adapted by utilizing eight V- and S-shaped transfer functions to present a new robust binary variant for KP01; this variant is called BKOA. Several studies were carried out to assess the performance of BKOA against several competing optimizers for solving 20 well-known KP01 instances with dimensions ranging from 4 to 75. The experimental results reveal that BKOA outperforms several metaheuristic algorithms, with the exception of the genetic algorithm, which somewhat outperforms BKOA. To

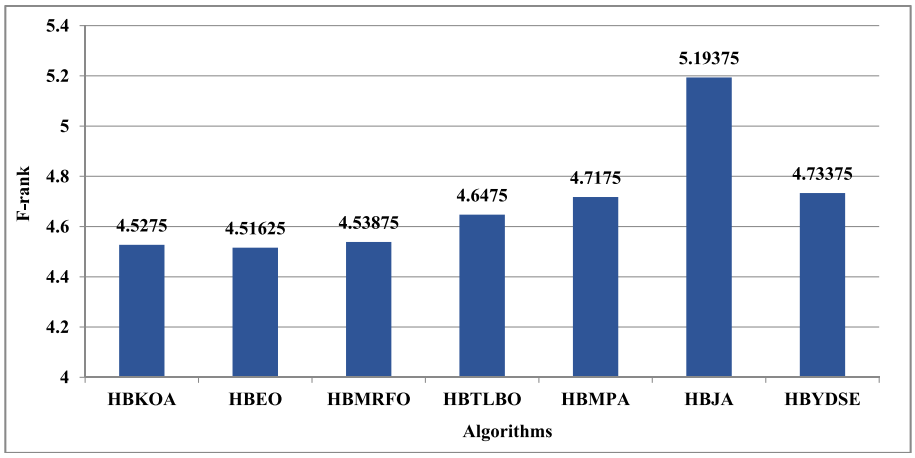


Fig. 11. Average F-rank of HBKOA and various competitors.

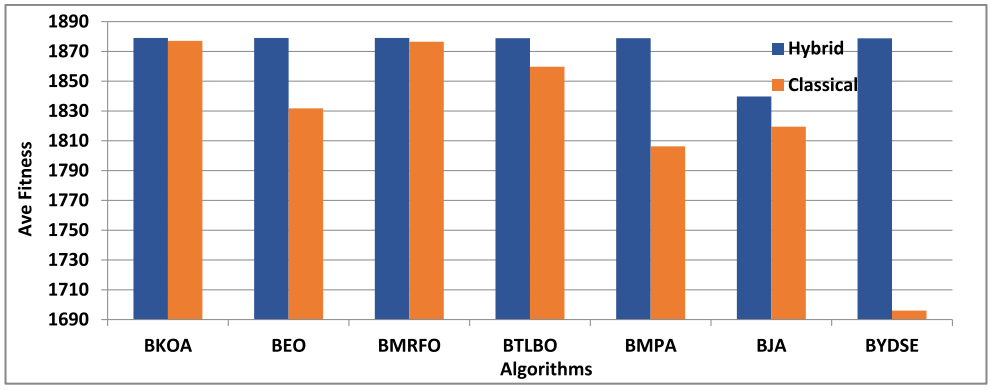


Fig. 12. Comparison between classical binary algorithms and their hybrid variants.

Table 13
Comparison over high-dimensional instances (KP21-KP32).

	KP21						**KP22**					
	Worst	Ave	Best	SD	Time	F-rank	Worst	Ave	Best	SD	Time	F-rank
HBKOA	**9147.000**	**9147.000**	**9147.000**	**0.00E+00**	2.95E−01	**1.750**	**11238.000**	**11238.000**	**11238.000**	**0.00E+00**	3.02E−01	**2.0**
HBEO	8897.000	9034.800	9147.000	1.15E+02	**9.49E−03**	2.500	**11238.000**	**11238.000**	**11238.000**	**0.00E+00**	8.83E−02	**2.0**
HBMRFO	**9147.000**	**9147.000**	**9147.000**	**0.00E+00**	2.21E−02	**1.750**	**11238.000**	**11238.000**	**11238.000**	**0.00E+00**	**5.54E−02**	**2.0**
	KP23						**KP24**					
HBKOA	28131.000	28437.650	28659.000	1.55E+02	2.27E+00	3.000	50499.000	51103.450	51926.00	3.73E+02	3.52E+00	3.0
HBEO	28638.000	28755.000	28805.000	5.14E+01	4.98E−01	1.900	52279.000	52561.000	53199.00	2.10E+02	1.12E+00	2.0
HBMRFO	**28794.000**	**28828.200**	**28857.000**	**1.73E+01**	**3.63E−01**	1.100	**53801.000**	**54019.200**	**54503.00**	**1.62E+02**	**6.28E−01**	**1.0**
	KP25						**KP26**					
HBKOA	1512.000	1513.250	1514.000	5.50E−01	2.84E+00	2.200	1598.000	1622.100	1634.00	8.20E+00	6.56E−01	2.9
HBEO	1512.000	1512.850	1514.000	5.87E−01	4.34E−01	2.600	**1634.000**	**1634.000**	**1634.00**	**0.00E+00**	4.75E−02	**1.5**
HBMRFO	**1514.000**	**1514.000**	**1514.000**	**0.00E+00**	**3.53E−02**	**1.200**	1627.000	1633.650	**1634.00**	**1.57E+00**	**2.56E−02**	**1.5**
	KP27						**KP28**					
HBKOA	4228.000	4319.850	4400.000	5.17E+01	1.85E+00	3.000	8116.000	8257.550	8449.00	7.74E+01	4.11E+00	3.0
HBEO	4498.000	4532.950	4556.000	1.70E+01	3.59E−01	1.925	8731.000	8785.650	8867.00	4.02E+01	1.07E+00	2.0
HBMRFO	**4523.000**	**4553.750**	**4566.000**	**8.61E+00**	**2.60E−01**	**1.075**	**8948.000**	**8984.850**	**9010.00**	**1.46E+01**	**5.39E−01**	**1.0**
	KP29						**KP30**					
HBKOA	2396.000	2396.900	**2397.000**	3.08E−01	2.63E+00	1.825	**2697.000**	**2697.000**	**2697.00**	**0.00E+00**	8.21E−02	**2.0**
HBEO	2375.000	2393.050	**2397.000**	5.45E+00	4.08E−01	2.475	**2697.000**	**2697.000**	**2697.00**	**0.00E+00**	3.03E−02	**2.0**
HBMRFO	**2397.000**	**2397.000**	**2397.000**	**0.00E+00**	**3.79E−02**	**1.700**	**2697.000**	**2697.000**	**2697.00**	**0.00E+00**	**4.44E−03**	**2.0**
	KP31						**KP32**					
HBKOA	7103.000	7114.400	**7117.000**	3.90E+00	1.75E+00	2.600	14248.000	14282.800	**14390.00**	**1.04E+01**	3.75E+00	2.9
HBEO	**7117.000**	**7117.000**	**7117.000**	**0.00E+00**	2.45E−01	**1.700**	**14290.000**	14349.000	**14390.00**	4.94E+01	1.11E+00	1.8
HBMRFO	**7117.000**	**7117.000**	**7117.000**	**0.00E+00**	**3.89E−02**	**1.700**	**14290.000**	**14379.950**	**14390.00**	3.08E+01	**1.15E−01**	**1.1**

outperform GA, BKOA is efficiently combined with a novel technique known as enhanced improvement strategy (EIS) to propose a new version known as HBKOA. This variant has superior exploration and exploitation capabilities, which make it outperform GA and the others across all performance metrics. Furthermore, the impact of EIS on the performance of some binary metaheuristic algorithms such as BEO, BMPA, BMRFO, BTLBO, BJA, and BYDSE is investigated in additional experiments. The results of these experiments disclose that HBKOA, HBMRFO, and HBEO are competitive for small and medium-dimensional KP01 instances; meanwhile, HBMRFO is better than all others for high-dimensional KP01 instances. Future work involves applying BKOA to tackle multidimensional knapsack problems and feature selection. In addition, a multi-objective variant of KOA is considered in the future for solving various multi-objective optimization problems, like multi-objective feature selection, multi-objective task scheduling problems, and several others. Also, KOA will be applied to solve several other single-objective optimization problems like image segmentation problems, fragment assembly problems, and parameter estimation of fuel cell and photovoltaic systems.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Funding

This research is supported by the Researchers Supporting Project number (RSP2023R389), King Saud University, Riyadh, Saudi Arabia.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] E. Baş, Binary Aquila Optimizer for 0–1 knapsack problems, Eng. Appl. Artif. Intel. 118 (2023), 105592.
[2] Z. Halim, Optimizing the DNA fragment assembly using metaheuristic-based overlap layout consensus approach, Appl. Soft Comput. 92 (2020), 106256.
[3] M. Abdel-Basset, et al., Binary light spectrum optimizer for knapsack problems: An improved model, Alex. Eng. J. 67 (2023) 609–632.
[4] M. Banaie-Dezfouli, M.H. Nadimi-Shahraki, and Z. Beheshti, *BE-GWO: Binary extremum-based grey wolf optimizer for discrete optimization problems.* Applied Soft Computing, 2023: p. 110583.
[5] G. Yildizdan, E. Baş, A Novel Binary Artificial Jellyfish Search Algorithm for Solving 0–1 Knapsack Problems, Neural Process. Lett. (2023) 1–67.
[6] B. Ervural, H. Hakli, A binary reptile search algorithm based on transfer functions with a new stochastic repair method for 0–1 knapsack problems, Comput. Ind. Eng. 178 (2023), 109080.
[7] M. Abdel-Basset, et al., New binary marine predators optimization algorithms for 0–1 knapsack problems, Comput. Ind. Eng. 151 (2021), 106949.
[8] B. Abdollahzadeh, et al., An enhanced binary slime mould algorithm for solving the 0–1 knapsack problem, Eng. Comput. (2021) 1–22.
[9] M. Abdel-Basset, R. Mohamed, S. Mirjalili, A Binary Equilibrium Optimization Algorithm for 0–1 Knapsack Problems, Comput. Ind. Eng. (2020), 106946.
[10] X. Li, W. Fang, S. Zhu, An improved binary quantum-behaved particle swarm optimization algorithm for knapsack problems, Inf. Sci. (2023), 119529.
[11] G.O. Büyüköz, H. Hakli, *Binary Honey Badger Algorithm for 0-1 Knapsack Problem.* Journal of Intelligent Systems: Theory and Applications. **6**(2): p. 108-118.
[12] X. Du, et al., A novel binary multi-swarms fruit fly optimisation algorithm for the 0–1 multidimensional knapsack problem, Int. J. Bio-Inspired Computation 21 (1) (2023) 1–10.
[13] E. Ballinas, O. Montiel, Hybrid quantum genetic algorithm with adaptive rotation angle for the 0–1 Knapsack problem in the IBM Qiskit simulator, Soft. Comput. 27 (18) (2023) 13321–13346.
[14] S. Harifi, A binary ancient-inspired Giza Pyramids Construction metaheuristic algorithm for solving 0–1 knapsack problem, Soft. Comput. 26 (22) (2022) 12761–12778.
[15] N. Moradi, V. Kayvanfar, M. Rafiee, An efficient population-based simulated annealing algorithm for 0–1 knapsack problem, Eng. Comput. (2021) 1–20.
[16] J. Cao, et al., A modified artificial bee colony approach for the 0–1 knapsack problem, Appl. Intell. 48 (2018) 1582–1595.
[17] Y. Feng, et al., Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization, Neural Comput. & Applic. 28 (2017) 1619–1634.
[18] Y. Wang, W. Wang, Quantum-inspired differential evolution with grey wolf optimizer for 0–1 knapsack problem, Mathematics 9 (11) (2021) 1233.
[19] R.M. Rizk-Allah, A.E. Hassanien, New binary bat algorithm for solving 0–1 knapsack problem, Complex Intelligent Syst. 4 (2018) 31–53.
[20] M. Abdel-Basset, et al., Recent metaheuristic algorithms with genetic operators for high-dimensional knapsack instances: A comparative study, Comput. Ind. Eng. 166 (2022), 107974.
[21] K. Liu, et al., A hybrid harmony search algorithm with distribution estimation for solving the 0–1 knapsack problem, Math. Probl. Eng. 2022 (2022).
[22] Z. Shu, et al., A modified hybrid rice optimization algorithm for solving 0–1 knapsack problem, Appl. Intell. 52 (5) (2022) 5751–5769.
[23] L. Fang, Y. Yao, X. Liang, New Binary Archimedes Optimization Algorithm and its application, Expert Syst. Appl. (2023), 120639.
[24] B. Li, Z.-B. Tang, Multiple level binary imperialist competitive algorithm for solving heterogeneous multiple knapsack problem, J. Comput. Appl. (2023).
[25] M. Abdel-Basset, et al., Performance Optimization and Comprehensive Analysis of Binary Nutcracker Optimization Algorithm: A Case Study of Feature Selection and Merkle-Hellman Knapsack Cryptosystem, Complexity 2023 (2023).
[26] T.T. Dhivyaprabha, P. Subashini, Synergistic Fibroblast Optimization Algorithm for Solving Knapsack Problem, in: Modern Artificial Intelligence and Data Science: Tools, Techniques and Systems, Springer, 2023, pp. 295–306.
[27] Y. Kang, et al., TMHSCA: a novel hybrid two-stage mutation with a sine cosine algorithm for discounted 0–1 knapsack problems, Neural Comput. Applic. 35 (17) (2023) 12691–12713.
[28] J.C. Bansal, et al., Sine Cosine Algorithm for Discrete Optimization Problems, in: Sine CoSine Algorithm for Optimization, Springer, 2023, pp. 65–86.
[29] S. Gupta, R. Su, S. Singh, Diversified sine–cosine algorithm based on differential evolution for multidimensional knapsack problem, Appl. Soft Comput. 130 (2022), 109682.
[30] L.J. Mohammed, Z. Algamal, Solving 0–1 Knapsack problem by an improved binary coyote optimization algorithm, Math. Stat. Eng. Appl. 71 (3) (2022) 1432–1448.
[31] K.M. Sallam, A.A. Abohany, R.M. Rizk-Allah, An enhanced multi-operator differential evolution algorithm for tackling knapsack optimization problem, Neural Comput. & Applic. 35 (18) (2023) 13359–13386.
[32] Y. Feng, G.-G. Wang, A binary moth search algorithm based on self-learning for multidimensional knapsack problems, Futur. Gener. Comput. Syst. 126 (2022) 48–64.
[33] H.S. Alamri, et al. *Solving 0/1 knapsack problem using opposition-based whale optimization algorithm (OWOA).*
[34] M. Abdel-Basset, et al., Kepler optimization algorithm: A new metaheuristic algorithm inspired by Kepler's laws of planetary motion, Knowl.-Based Syst. 268 (2023), 110454.
[35] C.-L. Huang, C.-J. Wang, A GA-based feature selection and parameters optimizationfor support vector machines, Expert Syst. Appl. 31 (2) (2006) 231–240.
[36] K.K. Ghosh, et al., S-shaped versus V-shaped transfer functions for binary Manta ray foraging optimization in feature selection problem. 2021. **33**(17): p. 11027-11041.
[37] M. Allam, M. Nandhini, Optimal feature selection using binary teaching learning based optimization algorithm, J. King Saud Univ.-Comput. Inform. Sci. 34 (2) (2022) 329–341.
[38] A. Chaudhuri, T.P. Sahu, Binary Jaya algorithm based on binary similarity measure for feature selection, J. Ambient Intell. Hum. Comput. (2021) 1–18.
[39] M. Abdel-Basset, et al., Young's double-slit experiment optimizer: A novel metaheuristic optimization algorithm for global and constraint optimization problems, Comput. Methods Appl. Mech. Eng. 403 (2023), 115652.
[40] A.E. Ezugwu, et al., A comparative study of meta-heuristic optimization algorithms for 0–1 knapsack problem: Some initial results, IEEE Access 7 (2019) 43979–44001.