Computers & Industrial Engineering 166 (2022) 107974

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Recent metaheuristic algorithms with genetic operators for high-dimensional knapsack instances: A comparative study

Mohamed Abdel-Basset [a], Reda Mohamed [a], Osama M. Elkomy [a], Mohamed Abouhawwash [b,c,*]

[a] Faculty of Computers and Informatics, Zagazig University, Shaibet an Nakareyah, Zagazig, 44519 Ash Sharqia Governorate, Egypt
[b] Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt
[c] Department of Computational Mathematics, Science, and Engineering (CMSE), Michigan State University, East Lansing, MI 48824, USA

ARTICLE INFO

Keywords:
0–1 knapsack problem
Transfer function
Genetic operators
Meta-heuristic algorithms

ABSTRACT

As a new attempt to effectively tackle the high-dimensional 0–1 knapsack (01KP) instances with uncorrelated, weakly-correlated, and strongly-correlated characteristics, in this paper, five lately-proposed meta-heuristic algorithms: horse herd optimization algorithm (HOA), gradient-based optimizer (GBO), red fox search optimizer (RFSO), golden eagle optimizer (GEO), and Bonobo optimizer (BO) have been transformed into binary ones by investigating the various V-shaped and S-shaped transfer functions to be applied to those high-dimensional 01KP problems, which are discrete ones; these binary variants are named BHOA, BGEO, BBO, BRFSO, and BBO. Furthermore, some genetic operators such as the one-point crossover operator and mutation operators have been borrowed to discover more permutations as a trying to avoid stuck into local minima for reaching better outcomes. These two operators are effectively integrated with those binary variants to propose other ones with better performance for achieving further improvements for tackling the high-dimensional 01KP instances called BIHOA, BIGEO, BIGBO, BIBO, and BIRFSO. Those genetic operators and recently-developed meta-heuristic algorithms-based high dimensional binary techniques have been extensively validated using 21 uncorrelated, weakly-correlated, and strongly-correlated 01KP instances with high-dimensions ranging between 100 and 10000, and the obtained outcomes were compared even witnessing which algorithm is the best. The experimental findings show the superiority of BIRFSO for the instances with dimensions greater than 500, and its competitivity for the others.

1. Introduction

Lately, the metaheuristic algorithms either single-based algorithms or population-based ones have strongly interfered in tackling several optimization problems divided into combinatorial problems, such as knapsack problem, DNA fragment assembly problem, feature selection, task scheduling, and travel salesman problem; and continuous ones such as parameter estimation of photovoltaic models, and nonlinear equations systems (Gadekallu et al., 2021; Bhattacharya et al., 2020; Gadekallu et al., 2020); some of those recent algorithms are compared in Table 1. Involving those optimization problems, the knapsack problems (KP) classified as discrete/combinatorial ones have won a significant interest by the metaheuristic researchers to find the near-optimal subset of items that maximize the profit with satisfying the knapsack capacity for optimally allocating the resources of several domain s(Li, He, Li, & Guo, 2021). There are several types of the knapsack problems, such as the multidimensional KP (MKP) (Wang, Zheng, & Wang, 2013), the quadratic KP (QKP) (Patvardhan, Bansal, & Srivastav, 2016), the discount 0–1 KP (Guldan, 2007), the bounded KP (Pisinger, 1995), and the set union KP (SUKP) (He, Xie, Wong, & Wang, 2018), which are considered as the extensions to the classical 0–1 KP (01KP) (Martello, 1990). The main challenge to these problems is finding the near-optimal subset in a reasonable time due to the NP-hard nature.

Within this work, an extensive investigation study will be conducted to investigate the performance of some of the newly-proposed meta-heuristic algorithms for the high dimensional 01KP instances as an attempt to propose an effective approach having strong performance to tackle this type of problem. Before explaining the works done for tackling the 0–1KP and why a new strong algorithm is significantly required, let's define the 01KP and describe its mathematical model. Each 01KP

* Corresponding author at: Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt.
E-mail addresses: mohamedbasset@zu.edu.eg (M. Abdel-Basset), redamoh@zu.edu.eg (R. Mohamed), osamaelkomy@zu.edu.eg (O.M. Elkomy), abouhaww@msu.edu, saleh1284@mans.edu.eg (M. Abouhawwash).

https://doi.org/10.1016/j.cie.2022.107974
Received 16 April 2021; Received in revised form 3 August 2021; Accepted 26 January 2022
Available online 1 February 2022
0360-8352/Published by Elsevier Ltd.

consists of n items, where each one has a profit p_i and a weight w_i. The objective of solving this problem is finding a subset of the items with a summation of the weights less than or equal to the knapsack capacity (c) and a maximized profit. The mathematical model of the 01KP is described below:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{n} x_i * p_i, \\ \text{subjectto} \quad & \sum_{i=1}^{n} w_i * x_i < c \end{aligned} \tag{1}$$

Several attempts have been done over the last decades to solve this problem, but don't fulfill the desired target even now as described next. In (Li, He, Liu, Guo, & Li, 2020), a new transfer function has been proposed to convert the continuous values produced by the whale optimization algorithm (WOA) into 0 and 1 ones for solving the 01KP; this variant is named a discrete whale optimization algorithm (DWOA). DWOA was extensively compared with a number of state-of-the-art algorithms using 25 instances with dimensions ranging between 16 and 24, which is so small, and hence, its performance for high-dimensional datasets is not determined as its main limitation although of its superiority for the small instances in comparison to the others.

The harmony search algorithm (Adamuthe et al., 2020) has been developed for the 01KP to find the optimal selection of items, which will maximize the profit. This algorithm was investigated using 43 instances of small-, and medium-scale; however, its performance for high-scale is

Table 1
Briefly discussion to some newly-proposed metaheuristic algorithms.

Algorithms and Year	Contributions
Equilibrium optimizer (EO, 2020) (Faramarzi et al., 2020)	This algorithm has been recently developed to simulate the generic mass balance equation for a control volume as a new attempt to propose a metaheuristic algorithm having a different search methodology can find better outcomes compared to the existing one. Compared to a satisfying number of the well-established well-known metaheuristic algorithms could come true better outcomes for a significant number of test functions. However, after observing its methodology it is obvious that this algorithm will maximize the exploration operator at the starting of the optimization process and fade away gradually with increasing the current iteration, and at the same time, the exploitation capability start weak and strength with increasing the iteration; this will reduce the convergence speed and sometimes might not be able to avoid stuck into local optima.
Marine predators algorithm (MPA, 2020) (Faramarzi et al., 2020)	Inspired by the behaviors of the predators in the ocean during attacking prey, this algorithm has been recently published for tackling the global optimization problem. This algorithm has good outcomes compared to some metaheuristic algorithms but suffers from high computational costs caused by the levy flight. Also, it could not avoid the disadvantages of the equilibrium optimizer mentioned previously.
Slime mould algorithm (SMA, 2020) (Li et al., 2020)	In this paper, the oscillation mode of slime mould has been mimicked to propose a new stochastic optimizer, namely slime mould optimizer, having strong features that enable it to adapt itself in the direction of the optimal solution. Relating the updating process with the fitness value of the previous generation to determine if the current individual needs an exploration operator or exploitation one, and hence this will improve the convergence speed in addition to avoiding stuck into local minima.
Gradient-based optimizer (GBO, 2020) (Ahmadianfar et al., 2020)	Integrating a gradient technique with the behaviors of the metaheuristic algorithms considers a good idea because the gradient technique might guide the individuals during the optimization process in the direction of the near-optimal solution. Based on this idea, GBO has been recently developed for tackling the global optimization problems and could fulfill outstanding outcomes compared to some state-of-the-arts. The gradient technique significantly suffers from falling into local optima and hence its guidance might not prevent stagnation into local minima.
Horse herd optimization algorithm (HOA, 2021) (MiarNaeimi et al., 2021)	A new population-based optimization algorithm known as horse herd optimization algorithm inspired by the horse herd behaviors, which consist of six important features: grazing (G), hierarchy (H), sociability (S), imitation (I), defense mechanism (D), and roam (R), has been proposed for tackling high-dimensional optimization problems.
Red fox search optimizer (RFSO, 2021) (Połap and Woźniak, 2021)	This algorithm was proposed based on mimicking the red fox behaviors for finding food, developing population, and hunting with running away from hunters. It was assessed using the global optimization problems that are efficiently solved by it in comparison to some state-of-the-art algorithms.
Golden eagle optimizer (GEO, 2021) (Mohammadi-Balani et al., 2021)	Inspired by the intelligent behavior of the golden eagle, a new metaheuristic optimization algorithm called golden eagle optimizer (GEO) has been recently proposed for tackling optimization problems. The mathematical model of the GEO is based on two steps: (1) attacking the prey to promoting the local search operator (exploitation), and (2) cruising to explore other regions for finding better foods (exploration). The experimental outcomes revealed the superiority of this algorithm with regard to six other metaheuristic algorithms.
Bonobo optimizer (BO, 2021) (Das and Pratihar, 2019)	A new population-based metaheuristic algorithm, namely BO, has been recently proposed for tackling the parameter estimation problem of the photovoltaic models based on mimicking the social behaviors and the developing process of bonobos. The bonobos follow a fission–fusion social strategy: the fission stage divides the bonobos into smaller groups to explore the search space for searching the food, then, they are again regathered to do some activities together (fusion behavior)
Archimedes optimization algorithm (AOA, 2021) (Hashim et al., 2021)	As an inspiration of physics motivates attention to proposing a new metaheuristic algorithm known Archimedes optimization algorithm (AOA) to mimic the Archimedes law. This algorithm is not only simple but also has few control parameters to be adjusted before starting the optimization process. The experimental outcomes of the AOA could be better than the others' outcomes used in comparison in terms of the solution quality and convergence speed.
Chaos game optimization (CGO, 2021) (Talatahari and Azizi, 2021)	Inspired by chaos theory, a new metaheuristic algorithm called chaos game optimization has been recently proposed for tackling optimization problems. This algorithm was evaluated using 239 mathematical test functions and compared with six different metaheuristic algorithms which have bad performance in confronts to this algorithm.
Jellyfish search optimizer (JSO, 2020) (Chou and Truong, 2021)	This algorithm known as artificial Jellyfish Search optimizer (JSO) was inspired by the behavior of jellyfish in the ocean and proposed for the first time for tackling global optimization problems. in specific, The behavior of the JSO for finding food in the ocean is based on: movements inside the swarm or following the ocean current and switching between these movements using a time control mechanism. This algorithm was validated on fifty small- and medium-scale test functions, in addition to 25 large-scale ones and some engineering optimization problems. then it was extensively compared to some well-known optimization algorithms to show that it performs best.

not shown. A population-based simulated annealing algorithm (PSA) (Moradi, Kayvanfar, & Rafiee, 2021) has been suggested for the optimal identification of the items that maximize the profit by satisfying the knapsack capacity. This algorithm was extensively assessed using several well-known instances with small-, medium-, and large-scale dimensions up to 10000 and compared with a number of the existing simulated annealing variants, and some of the other optimizers to see its superiority. The experimental findings show the superiority of this algorithm in comparison to the compared ones, but its performance for weakly correlated high-dimensional and strongly correlated high-dimension instances still needs a significant improvement.

Furthermore, a new binary variant of the elephant herding optimization (EHO) algorithm, namely briefly BinEHO, was adapted for tackling the 01KP (Hakli, 2020). This algorithm was validated 25 well-known 01KP instances and compared with some of the existing binary techniques, such as binary PSO (BPSO) (Ali, Luque, & Alba, 2020), modified BPSO (MBPSO) (Bansal & Deep, 2012), binary harmony search algorithm (NGHS) (Zou, Gao, Li, & Wu, 2011), discrete global best harmony search algorithm (DGHS) (Xiang, An, Li, He, & Zhang, 2014), a simplified binary artificial fish swarm algorithm (S-BAFS) (Azad, Rocha, & Fernandes, 2014), and improved monkey algorithm (Zhou, Chen, & Zhou, 2016). Despite its superiority for solving the small-scale 01KP instances, its performance for tackling the instances of high-scale was not examined and this made it not preferred for solving the 01KP instances with high-dimensionality. MBPSO (Bansal & Deep, 2012) was applied to find the optimal subset of the items for the instances with a scale reaching 500 dimensions. Also, some algorithms (Bhattacharjee & Sarmah, 2015; Bhattacharjee & Sarmah, 2015) have been assessed using well-known instances of small-, and medium-scale up to 75.

Consequently, most algorithms proposed in the literature have only dealt with the small-scaled 01KP instances and their performance for the high dimensional problem, which is harder because a huge number of the permutations needs to be observed compared to the small-, and medium-scaled dimensions, is not defined. Therefore, in this paper, five recently-developed metaheuristics optimization algorithms: Horse herd optimization algorithm (HOA), Gradient-based optimizer (GBO), Red fox search optimizer (RFSO), Golden eagle optimizer (GEO), and Bonobo optimizer (BO) have been extensively investigated for tackling the high-dimensional 01KP instances. At the outset, because those used algorithms were already proposed for solving the continuous problems, they are first converted into discrete algorithms by two well-known transfer function families: V-shaped and S-shaped to be applicable for solving these discrete problems. Afterward, the five investigated algorithms integrated with the best transfer function for each one have been extensively compared with each other for solving 21 well-known high-dimensional instances having dimensions ranging between 100 and 10000, and the obtained outcomes were analyzed using various statistical analyzes like the best, average, worst, error percent, and standard deviation values, in addition to the convergence speed to see the acceptance of each algorithm, and the computational cost for the speedup of each algorithm. Finally, the main contributions of this paper are listed as:

1. Investigating the performance of five recently-published meta-heuristic algorithms for tackling the 01KP with various scales, as the first time to the best of our knowledge those algorithms are proposed for tackling this problem.
2. Borrowing some genetic operators such as the one-point crossover operator and mutation operator to be integrated with those algorithms to discover more permutations for reaching the near-optimal solution, especially with the large-scale problem.
3. Investigated experiments show the effective role of the integrated genetic operators for reaching better outcomes with the large-scale problems, higher than 500.

The rest of this research is organized like that: Section 2 overview the five recently-developed algorithms, Section 3 describes what we propose, Section 4 shows the analyses of the outcomes obtained by the proposed algorithms, and Section 5 finally briefly presents our conclusion and what will be studied in the future.

2. Recently-proposed metaheuristics

In this section, five recently-proposed metaheuristic algorithms will be discussed to show their methodology in searching for the near-optimal solution for the optimization problems. Only the mathematical model of the gradient-based optimizer and horse herd optimization algorithm will be reviewed in this paper, while the other algorithms are only mentioned based on their search methodology.

2.1. *Horse herd optimization algorithm (HOA)*

Based on six important features: grazing (G), hierarchy (H), sociability (S), imitation (I), defense mechanism (D), and roam (R) of horse herding behaviors, a new population-based optimization algorithm has been proposed for tackling high-dimensional optimization problems (MiarNaeimi, Azizyan, & Rashki, 2021). These features are related with the horse based on its age. Those six mentioned behaviors are related to the age of the horses. Specifically, each horse is updated within the optimization process using the following formula with taking into consideration the age of each one to determine which feature is from its advantages:

$$\overrightarrow{x}_i^{t,AGE} = \overrightarrow{V}_i^{t,AGE} + \overrightarrow{x}_i^{(t-1),AGE}, AGE = \alpha,\beta,\gamma,\delta \tag{2}$$

$x_i^{t,AGE}$ is a vector including the current position of the i^{th} horse, the current iteration is symbolized as t, and $\overrightarrow{V}_i^{t,AGE}$ expresses the velocity of the i^{th} horse. Based on the age of each horse, its velocity will be updated as described in the following equations:

$$\begin{aligned} \overrightarrow{V}_i^{t,\alpha} &= \overrightarrow{G}_i^{t,\alpha} + \overrightarrow{D}_i^{t,\alpha} \\ \overrightarrow{V}_i^{t,\beta} &= \overrightarrow{G}_i^{t,\beta} + \overrightarrow{H}_i^{t,\beta} + \overrightarrow{S}_i^{t,\beta} + \overrightarrow{D}_i^{t,\beta} \\ \overrightarrow{V}_i^{t,\gamma} &= \overrightarrow{G}_i^{t,\gamma} + \overrightarrow{H}_i^{t,\gamma} + \overrightarrow{S}_i^{t,\gamma} + \overrightarrow{I}_i^{t,\gamma} + \overrightarrow{D}_i^{t,\gamma} + \overrightarrow{R}_i^{t,\gamma} \\ \overrightarrow{V}_i^{t,\delta} &= \overrightarrow{G}_i^{t,\delta} + \overrightarrow{I}_i^{t,\delta} + \overrightarrow{R}_i^{t,\delta} \end{aligned} \tag{3}$$

Where α indicates the horses having ages greater than 15, δ distinguishes the horses with ages at the interval of 0 and 5, γ stands for the horses with ages between 5 and 10, and β refers to the horses whose ages lie at the range of 10 and 15. Those four symbols, which indicate various ages of the horses, are determined by the HOA based on sorting ascendingly the obtained fitness values and the 0.1 horses with the best fitness values will represent α, the next 0.2 horses indicate β,γ has the next 0.3 of the best horses, while the other hoses will represent δ.

2.1.1. *Grazing behavior*

The grazing behavior of the horses related to all ages: $\alpha,\beta,\gamma,\delta$ is mathematically formulated below:

$$\overrightarrow{G}_i^{t,AGE} = g_t(\breve{u} + \breve{i}\, P)[x_i^{t-1}], AGE = \alpha,\beta,\gamma,\delta \tag{4}$$

$$g_i^{t,AGE} = g_i^{t-1,AGE} * w_g \quad (5)$$

Where $\overrightarrow{G}_i^{t,AGE}$ is the movement parameter of the i^{th} horse. $\check{u}$ and $\check{i}$ respectively are the inferior and uppermost grazing boundary and recommended 0.95 and 1.05. P is a random number generated between 0 and 1. g is recommended 1.5 for all ages to represent a coefficient value.

2.1.2. Hierarchy behavior

This behavior only related to the β and γ horses as studied in (Waring, 1983; McDonnell et al., 2003) is mathematically described as:

$$\overrightarrow{H}_i^{t,AGE} = h_i^{t,AGE}[x_*^{t-1} - x_i^{t-1}], AGE = \beta, \gamma \quad (6)$$

$$h_i^{t,AGE} = h_i^{t-1,AGE} * w_h \quad (7)$$

h is a coefficient value to determine how far the horse will follow the most experienced one.

2.1.3. Sociability behavior

This social behavior confined only to the β and γ horses is mathematically described as follows:

$$\overrightarrow{S}_i^{t,AGE} = s_i^{t,AGE}[\frac{1}{N}\sum_{j=1}^{N} x_j^{t-1} - x_i^{t-1}], AGE = \beta, \gamma \quad (8)$$

$$s_i^{t,AGE} = s_i^{t-1,AGE} * w_s \quad (9)$$

$\overrightarrow{S}_i^{t,AGE}$ is computed to determine the tendency of the current horse to the herding in the current generation. N indicates the population size.

2.1.4. Imitation behavior

This behavior is only confined to the γ horses which try to mimic a number pN of the best horses as mathematically elaborated in the following equations.

$$\overrightarrow{I}_i^{t,AGE} = i_i^{t,AGE}[\frac{1}{pN}\sum_{j=1}^{pN} x_j^{t-1} - x_i^{t-1}], AGE = \gamma \quad (10)$$

$$m_i^{t,AGE} = m_i^{t-1,AGE} * w_i \quad (11)$$

pN is preferred to represent 10% of the best horses of the current generation as said in the original research (MiarNaeimi et al., 2021).

2.1.5. Defense mechanism behavior

This defense behavior of the horses owned only to α, β, and γ is mathematically described as.

$$\overrightarrow{D}_i^{t,AGE} = -d_i^{t,AGE}[\frac{1}{qN}\sum_{j=1}^{qN} x_j^{t-1} - x_i^{t-1}], AGE = \alpha, \beta, \gamma \quad (12)$$

$$d_i^{t,AGE} = d_i^{t-1,AGE} * w_d \quad (13)$$

qN indicates the horses with the worst position and recommended 20% of the population size.

2.1.6. Roam behavior

The mathematical model of this behavior is built as:

$$\overrightarrow{R}_i^{t,AGE} = r_i^{t,AGE} P x_i^{t-1}, AGE = \gamma, \delta \quad (14)$$

$$r_i^{t,AGE} = r_i^{t-1,AGE} * w_r \quad (15)$$

$r_i^{t-1,AGE}$ is a factor used to represent the random movement. The standard HOA is explained in Algorithm 1.

Algorithm 1. The steps of the standard HOA

1: Initialize a group of N horses, $\overrightarrow{x}_i^{t,AGE} (i \in N)$.
2: Initialize the HOA's parameters.
3: Compute the objective value of each horse, x_i.
4: t = 0;
5: **while** $(t < t_{max})$ **do**
6: Determines the ages of horses.
7: Compute the velocity related the age of each horse.
8: Update the horses.
9: Evaluate each horse, $\overrightarrow{x}_i^{t,AGE}$.
10: $t++$
11: **end while**
12: Return x_*^t.

2.2. Gradient-based optimizer (GBO)

Ahmadianfar (Ahmadianfar, Bozorg-Haddad, & Chu, 2020) developed a new population-based optimization algorithm known as gradient-based optimizer (GBO) based on following a gradient technique: the newton's method to guide the solutions through the optimization process to the valid direction of the near-optimal solution. Generally, the GBO algorithm is compounded by the gradient search rule and the local escaping operator described thoroughly later.

2.2.1. Gradient search rule (GSR)

This rule is used to integrate the gradient-based directions with the GBO algorithm for guiding the solutions inside the population to the true direction of the desired outcome. To balance between the exploration and exploitation operators, a significant factor ρ_1, is used to do that as an attempt to avoid local minima and accelerate the convergence speed at the same time. ρ_1 is mathematically modeled as:

$$\rho_1 = 2 \times r \times \alpha - \alpha \quad (16)$$

$$\alpha = \left|\beta \times \sin(\frac{3\pi}{2} + \sin(\beta \times \frac{3\pi}{2}))\right| \quad (17)$$

$$\beta = \beta_{min} + (\beta_{max} - \beta_{min}) \times (1 - (\frac{t}{t_{max}})^3)^2 \quad (18)$$

Where β_{min} and β_{max} are respectively two constant-values of 0.2 and 1.2. t_{max} is the maximum function evaluation. Afterward, ρ_1 is related with the GSR to manage exploration and exploitation operators for achieving an equilibrium between them during the whole optimization process as described in the following formula:

$$GSR = r \times \rho_1 \times \frac{(2\Delta x \times X_n)}{(X_w^{t-1} - X_*^{t-1} + \epsilon)} \quad (19)$$

ϵ is a tiny value between 0 and 0.1 to eliminate the division by zero. X_w^{t-1} is the worst solution at the current generation, while X_*^{t-1} is the best one. Δx is formulated as following:

$$\Delta x = \overrightarrow{r} \times |S| \quad (20)$$

$$S = \frac{((X_*^{t-1} - X_a^{t-1}) + \delta)}{2} \quad (21)$$

$$\delta = 2 \times r_2 \times (|\frac{(X_a^{t-1} + X_b^{t-1} + X_c^{t-1} + X_d^{t-1})}{4}| - X_i^{t-1}) \quad (22)$$

Where r_2 is a number created randomly at the interval of 0 and 1, $a \neq b \neq c \neq d$ are randomly-selected indices from the solutions. Then, according to the GSR strategy, a new solution could be obtained by the following formula:

$$X1_i^t = X_i^{t-1} - GSR \quad (23)$$

To extensively locally search nearby around the current solution, Eq. 23 is updated by the direction of movement (DM) described as:

$$X1_i^t = X_i^{t-1} - GSR + DM \quad (24)$$

$$DM = r \times \rho_2 \times (X_*^{t-1} - X_i^{t-1}) \quad (25)$$

$$\rho_2 = 2 \times r \times \alpha - \alpha \quad (26)$$

According to (Ahmadianfar et al., 2020), $X1_i^t$ could be reformulated as follows:

$$X1_i^t = x_i^{t-1} - r \times \rho_1 \times \frac{2\Delta x \times X_n}{(yp_i^{t-1} - yq_i^{t-1} + \epsilon)} + r \times \rho_2 \times (X_*^{t-1} - X_i^{t-1}) \quad (27)$$

yp_i^{t-1} and yq_i^{t-1} are computed using the following formulas:

$$yp_i^{t-1} = y_n + \Delta x \quad (28)$$

$$yq_i^{t-1} = y_n - \Delta x \quad (29)$$

y_n is a vector equal to the average of the current solution vector X_i^{t-1}, and the z_i^{t-1} calculated as:

$$y_i = \frac{X_i^{t-1)}}{z_i^{t-1}} | z_i^{t-1} = x_n - \vec{r} \times \frac{2\Delta x \times x_n}{(X_w^{t-1} - X_*^{t-1} + \epsilon)} \quad (30)$$

Also, to enhance the exploitation operator around to promote the best-so-far solution for improving the convergence rate, a new vector known as $X2_i^t$ is generated like Eq. 27 with swapping x_i^{t-1} by x_*^{t-1} as described mathematically below:

$$X2_i^t = x_*^{t-1} - r \times \rho_1 \times \frac{2\Delta x \times X_n}{(yp_i^{t-1} - yq_i^{t-1} + \epsilon)} + r \times \rho_2 \times (x_*^{t-1} - x_i^{t-1}) \quad (31)$$

Finally, The two vectors: $X1_i^t$, and $X2_i^t$ generated previously mix with another one: $X3_i^t$ to generate the next position for the ith solution as follows:

$$x_i^t = r_a(r_b \times X1_i^t + (1 - r_b) \times X2_i^t) + (1 - r_a) \times X3_i^t \quad (32)$$

$$X3_i^t = x_i^{t-1} - \rho_1 \times (X2_i^t - X1_i^t) \quad (33)$$

2.2.2. Local escaping operator stage

Besides, a new operator called a local escaping operator (LEO) is integrated with the GBO with a probability pr (recommended 0.5) to increase its local optima avoidance capability. The mathematical model of this operator is as follows:

$$x_i^{t-1} = \begin{cases} x_i^{t-1} + f_1(u_1 x_*^{t-1} - u_2 x_k^t) + f_2\rho_1(u_3(X2_i^t - X1_i^t)) + \dfrac{u_2(x_a^{t-1} - x_b^{t-1})}{2} & r < 0.5 \quad and \quad r_1 < pr \\ x_*^{t-1} + f_1(u_1 x_*^{t-1} - u_2 x_k^{t-1}) + f_2\rho_1(u_3(X2_i^t - X1_i^t)) + \dfrac{u_2(x_a^{t-1} - x_b^{t-1})}{2} & r \geqslant 0.5 \quad and \quad r_1 < pr \end{cases} \quad (34)$$

f_1, and f_2 are two randomly-generated numerical values based on the uniform distribution at the range of -1 and 1. u_1, u_2, and u_3 are three various randomly assigned numbers using the following equations:

$$u_1 = \begin{cases} 2r_1 & if\ \mu_1 < 0.5 \\ 1 & otherwise \end{cases} \quad (35)$$

$$u_2 = \begin{cases} r_1 & if\ \mu_1 < 0.5 \\ 1 & otherwise \end{cases} \quad (36)$$

$$u_3 = \begin{cases} r_1 & if\ \mu_1 < 0.5 \\ 1 & otherwise \end{cases} \quad (37)$$

Where μ_1 and r_1 are two random numbers between 0 and 1. x_k^t presented in Eq. 34 is computed using the following mathematical function:

$$x_k^t = \begin{cases} x_r & if\ \mu_2 < 0.5 \\ x_p^{t-1} & otherwise \end{cases} \quad (38)$$

x_p^{t-1} is a randomly-selected solution from the population at the current generation. x_r is a randomly-generated position vector within the search space of the optimization problem. μ_2 is a numerical value created randomly within 0 and 1. The pseudo-code of the GBO is briefly studied in Algorithm 2.

Algorithm 2. The standard GBO

```
1: Create an initial population of N solutions, x_i(i ∈ N).
2: Initialize pr and E parameters.
3: Evaluation and determination of X_w^{t-1} and X_*^{t-1}.
4: t = 2;
5: while do (t < t_max)
6:    for each i solutions do
7:      for each j dimensions do
8:         Find a ≠ b ≠ c ≠ d ≠ i from the population.
9:         Update x_ij^t according to Eq. 32.
10:     end for
11:     Update x_i^t according to Eq. 34.
12:   end for
13:   t + +
14:   Extract X_w^{t-1} and X_*^{t-1}.
15: end while
16: Return X_*^{t-1}.
```

2.3. Red fox search optimizer (RFSO)

A new meta-heuristic algorithm [24] based on simulating the red fox behaviors for finding food, developing population, and hunting with running away from hunters has been recently developed for solving the mathematical optimization problems; this algorithm was named red fox search optimizer (RFSO). The mathematical model of this algorithm is given in the original paper (Połap & Woźniak, 2021).

2.4. Golden eagle optimizer (GEO)

Based on the intelligent behavior of the golden eagle in adjusting their speed at different stages of the spiral trajectory for catching, a new swarm-based optimization algorithm called golden eagle optimizer (GEO) was proposed for tackling the single-, and multi-objective optimization problems. The mathematical model of the GEO is based on two steps: (1) attacking the prey to promoting the local search operator (exploitation), and (2) cruising to explore other regions for finding better foods (exploration). The mathematical model of each step is extensively described in (Mohammadi-Balani, Nayeri, Azar, & Taghizadeh-Yazdi, 2021).

2.5. Bonobo optimizer (BO)

A new population-based metaheuristic algorithm, namely BO, has been recently proposed for tackling the parameter estimation problem of the photovoltaic models based on mimicking the social behaviors and the developing process of bonobos (Das & Pratihar, 2019). The bonobos follow a fission–fusion social strategy: the fission stage divides the bonobos into smaller groups to explore the search space for searching the food, then, they are again regathered to do some activities together (fusion behavior). BO follows two different phases to determine the mating behavior of the bonobos' community: the first one is known as the positive phase and happens when appearing better solution than the current best-so-far one, while the second one is known as the negative phase occurs on the contrary. The number of consecutive times, where the positive phase happens, is called a positive phase count (PPC); meanwhile, the negative phase count (NPC) indicates the number of consecutive times where the negative phase is applied. more information about BO is found in (Das & Pratihar, 2019).

3. The proposed algorithms.

The five meta-heuristic algorithms mentioned earlier have been proposed for tackling the continuous optimization problems, which make them inapplicable for the 01KP. Therefore, in this section, a binary variant of each one of those five algorithms will be explained based on four main steps: initialization phase, evaluation, repairing and improvement (RI) strategy, and satisfaction conditions, in addition to another step used to promote the performance of this algorithm for tackling the problem with huge dimensions by borrowing some genetic operators: one-point crossover operator and mutation operator to search around the obtained solutions to avoid stuck into local minima for increasing the search ability of the proposed algorithm.

3.1. Initialization.

Since the 01KP is a discrete problem optimally solved by finding the subset of items, which maximizes the profit with coming true the knapsack capacity constraint, therefore a population of N solutions with d dimensions will be created and initialized randomly with 0's value to distinguish the unselected item and 1's value to indicate the selected as described in the following equation:

$$x_{i,j} = \begin{cases} 1 & if\ r > 0.5 \\ 0 & otherwise \end{cases} \tag{39}$$

Where r is a random number generated based on the uniform distribution between 0 and 1, and $x_{i,j}$ indicates the j^{th} dimension in the i^{th} individual.

3.2. Evaluation and RI strategy.

After completing the initialization step, the evaluation step will be fired to evaluate each solution for determining the quality of each one for solving this problem, which could be dealt with as a minimization problem by reducing the profit of the unselected items with satisfying the knapsack capacity. Generally, the fitness function used to evaluate the obtained solutions is expressed as:

$$f(x) = \sum_{z=1}^{d} w_z * x_z \leqslant c \tag{40}$$

Where w_z is a variable contains the weight value of the z^{th} item, and x_z is used to indicates if the z^{th} item is selected (including a value of 1) or not (including a value of 1). c indicates the knapsack capacity. Some of the solutions called infeasible solutions don't subject to the constraint of the knapsack capacity, and hence they could not be selected to represent the optimal solution for this problem notwithstanding its small profit. Therefore, a fixing strategy has been used by removing repeatedly the item with the smallest $\frac{pr_z}{w_z}$ even the solutions become feasible. Those feasible solutions will be improved using another improvement strategy which will add the knapsack to the item repeatedly with the highest $\frac{pr_z}{w_z}$ with coming true the knapsack capacity constraints. Those two strategies used to convert the infeasible solution into a feasible one then improving it is abbreviately called RIS and described in Algorithm 3.

Table 2
S-shaped and V-shaped Transfer Functions.

S-Shaped	V-Shaped
1- $F(\overrightarrow{x}) = \frac{1}{1+e^{-2*a}}$	5- $F(\overrightarrow{x}) = \left\lvert erf(\frac{\sqrt{\pi}}{2}a)\right\rvert$
2- $F(\overrightarrow{x}) = \frac{1}{1+e^{-a}}$	6- $F(\overrightarrow{x}) = \lvert tanh(a)\rvert$
3- $F(\overrightarrow{x}) = \frac{1}{1+e^{\frac{-a}{2}}}$	7- $F(\overrightarrow{x}) = \left\lvert \frac{a}{\sqrt{1+a^2}}\right\rvert$
4- $F(\overrightarrow{x}) = \frac{1}{1+e^{\frac{-a}{3}}}$	8- $F(\overrightarrow{x}) = \left\lvert \frac{2}{\pi} arctan(\frac{\pi}{2}a)\right\rvert$

Algorithm 3. Repairing and improving (RI) Algorithm

```
Input: x_i.
// Repairing algorithm.
while do (f(x_i) > c)
    Eliminate the item with the lowest pr_z/w_z.
end while
// Improving algorithm.
while do (f(x_i) ⩽ c)
    Puch the item with the highest pr_z/w_z in the knapsack.
end while
Return repaired improved x_i.
```

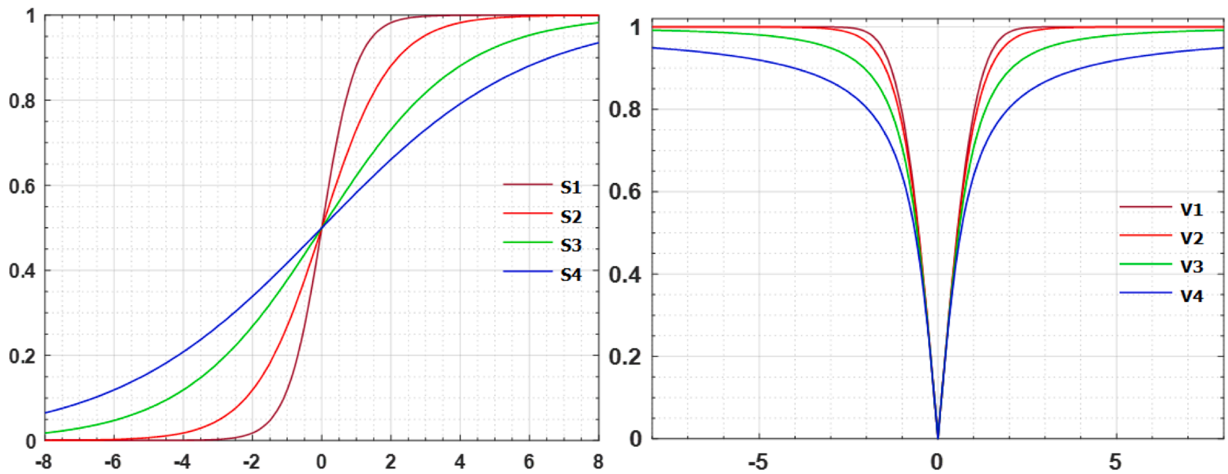
3.3. Transfer Functions

To make the metaheuristic algorithms released for tackling the continuous problems applicable to the 01KP with the discrete nature, eight well-known transfer functions of the V-shaped and S-shaped families described in Table 2 and depicted in Fig. 1 are here investigated to normalize the obtained continuous values between 0 and 1 then those normalized values are converted into 0 and 1 by Eq. 41. For example, the RFSO produces continuous values during the optimization process, which need to be converted into binary values to be adequate for tackling the knapsack problems. To do that, the sigmoid and V-Shaped transfer functions are used. In our experiments, the sigmoid transfer functions (S-Shaped) described mathematically in Table 2 and depicted in Fig. 1 could come true better outcomes with the continuous optimization algorithms when tackling the KP10 as shown in our papers (Abdel-Basset, Mohamed, & Mirjalili, 2021; Abdel-Basset, Mohamed, Chakrabortty, Ryan, & Mirjalili, 2021), so they are used in this research to see the performance of the five-investigated metaheuristic algorithms under these functions.

$$\overrightarrow{x}_{bin}(\overrightarrow{x}) = \begin{cases} 1 & if\ F(\overrightarrow{x}) \geqslant rand \\ 0 & otherwise \end{cases} \tag{41}$$

3.4. Genetic operators

The five proposed algorithms will employ the one-point crossover operator to explore more solutions for reaching better outcomes for the high-dimensional datasets. Specifically, the one-point crossover is used to generate the new offsprings from two parents by selecting randomly a point on the parents and the tails will be swapped between the two parents to produce two new off-springs as pictured in Fig. 2.



(a) S-shaped Transfer functions.	(b) V-shaped Transfer functions.

Fig. 1. Depiction of V-shaped and S-shaped transfer functions.

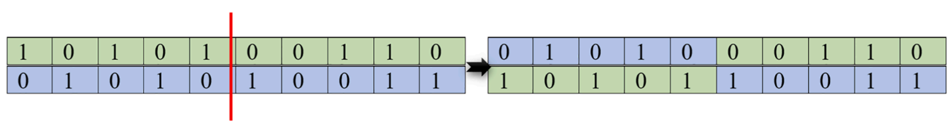


Fig. 2. Depiction of the one-point crossover.

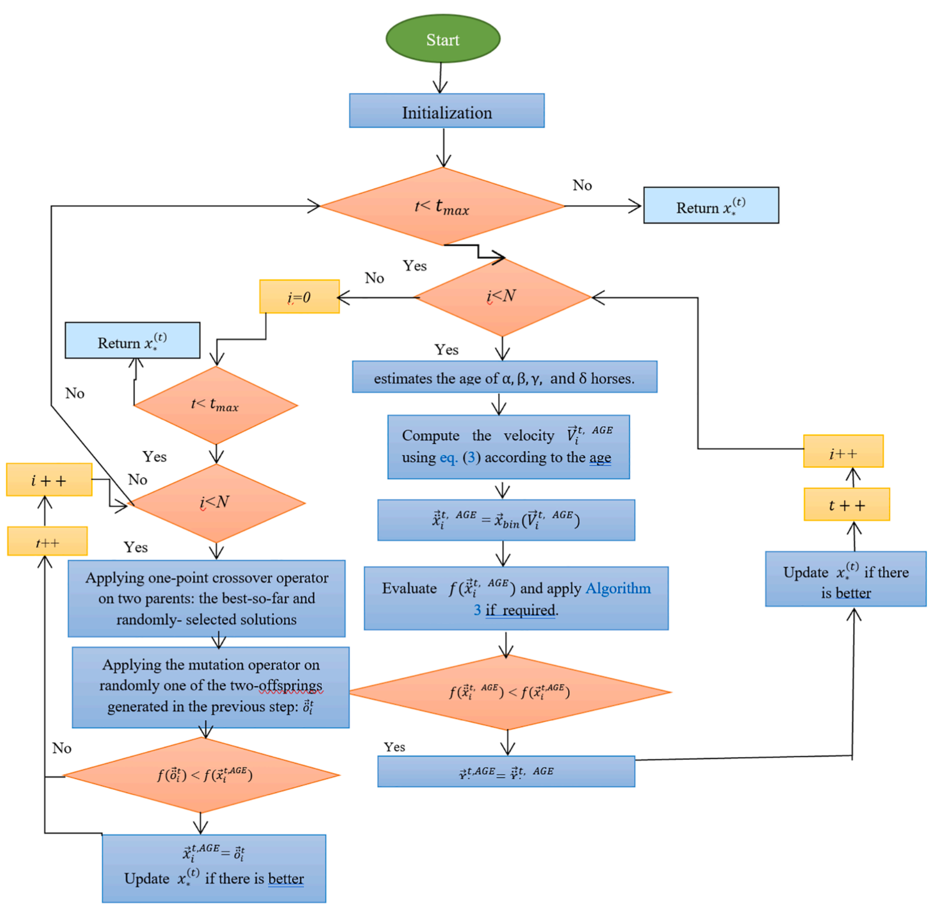


Fig. 3. Flowchart of BIHOA algorithm.

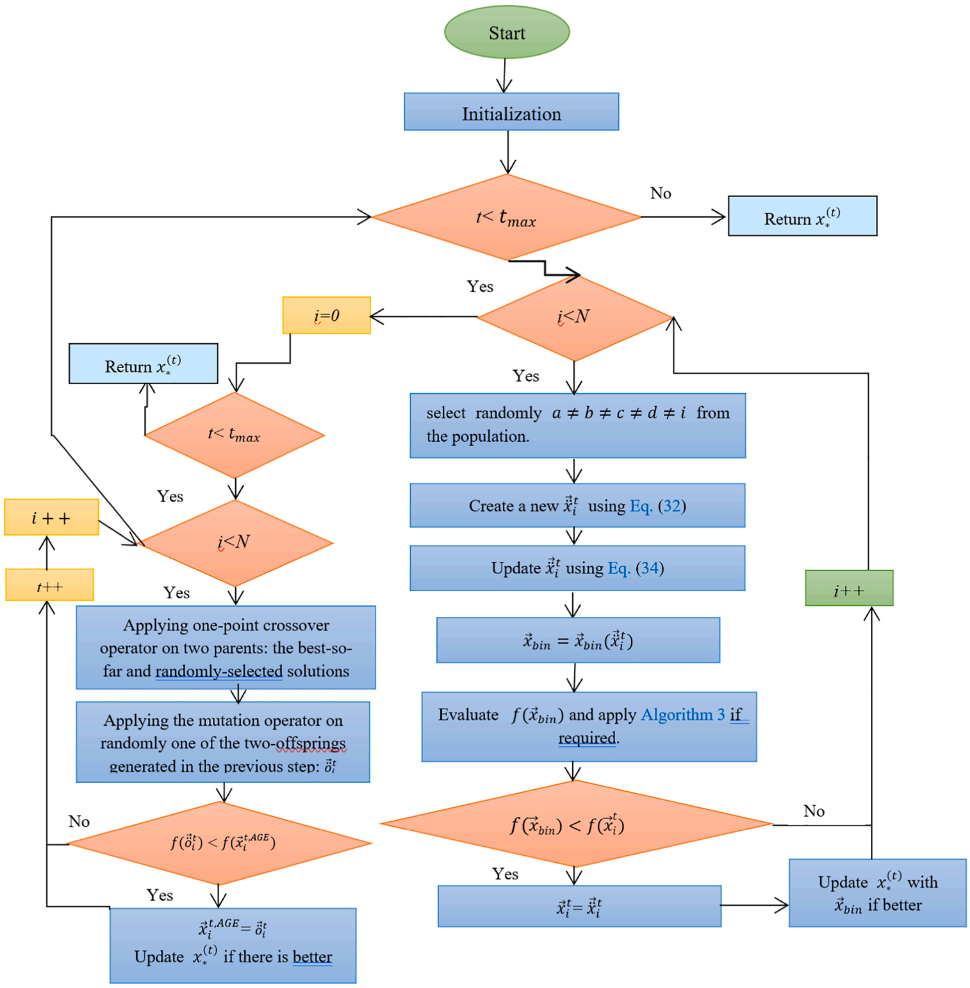


Fig. 4. Flowchart of BIGBO algorithm.

Within our work, one offspring of those will be randomly selected to be compared with the best local-one of the current solution as a new attempt to avoid stuck into local minima and help in reaching better outcomes. In a case, the crossover operator couldn't alone fulfill better outcomes because the solutions are already inside a local optimum and hence a new change needs to be made to alter entirely the selected offspring. Therefore, the mutation operator was borrowed to flip some values in this selected offspring according to a mutation probability (MP) recommended $1/d$ by several papers.

3.5. *The binary improved variant of HOA (BIHOA)*

Fig. 3 depicts the steps of the binary HOA integrated with the one-point crossover and mutation operators. Broadly speaking, at the outset, N solutions will be initialized randomly with binary values to determine which items will be selected. Then those initial solutions will be evaluated using Eq. 41 to determine which solutions could minimize the profit of the unselected items by satisfying the knapsack capacity constraint. After that, the optimization process will immediately begin to update the velocity of each horse according to its age, but this velocity involves continuous values contradicted with the knapsack problem, which requires binary values, therefore one of the transfer functions described in Table 2 will be used to convert the continuous values found in the velocity vector and save them in the position of the current horse, which is again evaluated and assigned to the best-local position if it is better. This process will continue until all the horses are updated with taking into consideration updating the best-so-far solution if any of the updated ones are better. After finishing this phase, the one-point

Table 3
Description of the high-dimensional KP01 instances.

ID	Capacity	D	Opt	ID	Capacity	D	Opt	ID	Capacity	D	Opt
Uncorrelated				Weakly-correlated				Strongly-correlated			
$KP1_{100}$	995	100	9147	$KP2_{100}$	995	100	1514	$KP3_{100}$	997	100	2397
$KP1_{200}$	1008	200	11238	$KP2_{200}$	1008	200	1634	$KP3_{200}$	997	200	2697
$KP1_{500}$	2543	500	28857	$KP2_{500}$	2543	500	4566	$KP3_{500}$	2517	500	7117
$KP1_{1000}$	5002	1000	54503	$KP2_{1000}$	5002	1000	9052	$KP3_{1000}$	4990	1000	14390
$KP1_{2000}$	10011	2000	110625	$KP2_{2000}$	10011	2000	18051	$KP3_{2000}$	9819	2000	28919
$KP1_{5000}$	25016	5000	276457	$KP2_{5000}$	25016	5000	44356	$KP3_{5000}$	24805	5000	72505
$KP1_{10000}$	49877	10000	563647	$KP2_{10000}$	49877	10000	90204	$KP3_{10000}$	49519	10000	146919

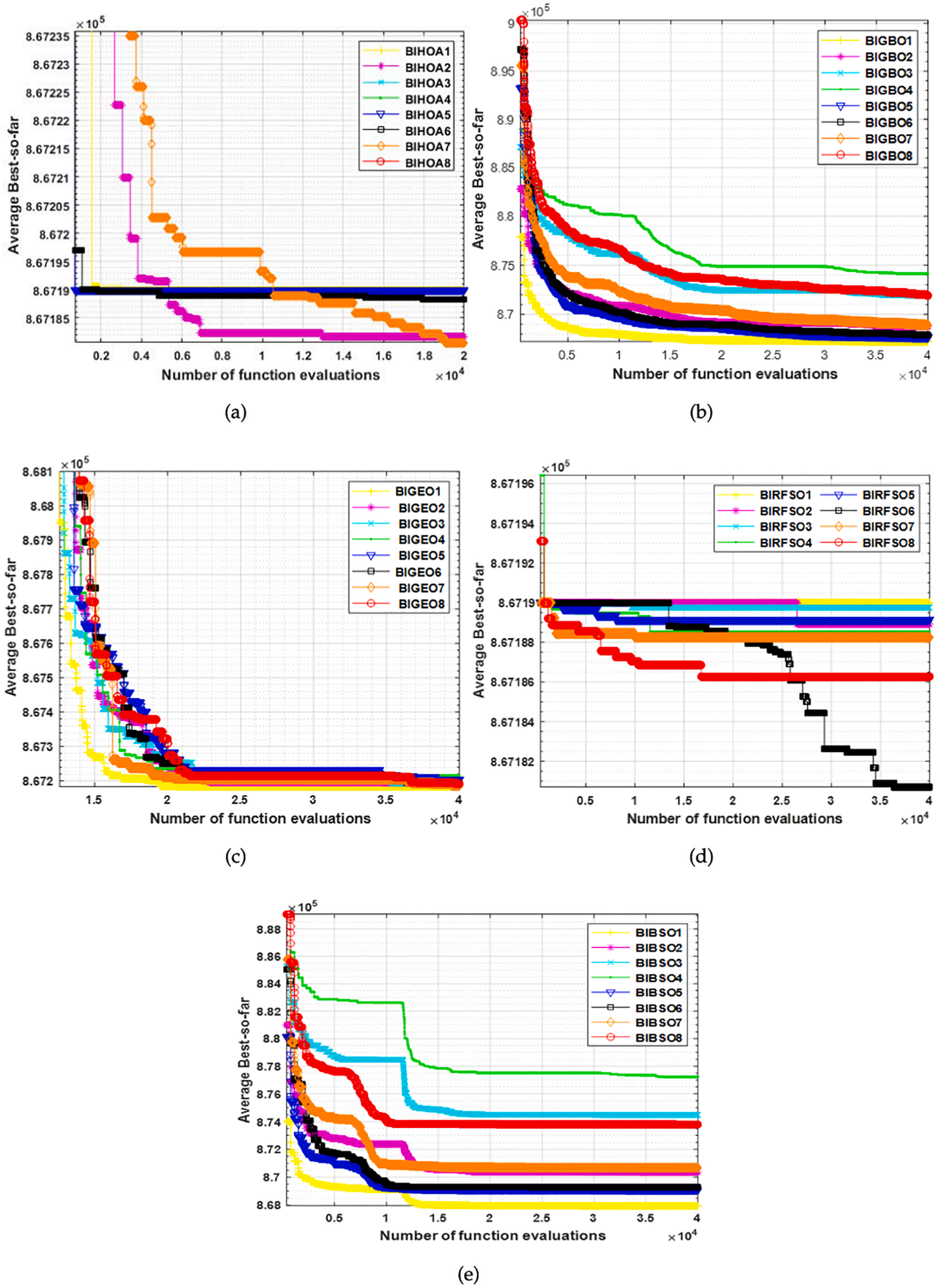


Fig. 5. Investigation of various transfer functions with five-observed metaheuristic algorithms.

crossover operator will be applied on two solutions: one selected randomly from the population and the other is the best-so-far solutions as an attempt to promote the exploitation capability to accelerate the convergence speed in the right direction of the optimal solution, especially with the high-dimensional problems. This crossover operator will generate two offsprings as described earlier, one of them will be selected randomly while the other is skipped to reduce the number of function evaluations consumed by this operator to give the standard algorithm a larger chance for searching for a better solution. Also, this offspring will be mutated by flipping its bits that lie within the mutation probability

Table 4
Comparison of the uncorrelated instances.

Id		BIHOA	BHOA	BIGBO	BGBO	BIGEO	BGEO	BIRFSO	BRFSO	BIBO	BBO
$KP1_{100}$	Worst	8929.000	8940.000	8940.000	**8990.000**	8900.000	8940.000	8929.000	8929.000	7815.000	8021.000
	Avg	9119.440	9119.880	9130.440	**9134.440**	9101.280	9094.760	9005.560	8989.960	8747.800	8741.320
	Best	**9147.000**	**9147.000**	**9147.000**	**9147.000**	**9147.000**	**9147.000**	**9147.000**	**9147.000**	**9147.000**	**9147.000**
	SD	65.346	64.034	57.316	**43.471**	84.919	78.307	93.251	92.143	255.604	239.699
	ER(%)	0.301	0.296	0.181	**0.137**	0.500	0.571	1.546	1.717	4.364	4.435
$KP1_{200}$	Worst	11227.000	11227.000	**11238.000**	11227.000	**11238.000**	**11238.000**	**11238.000**	11227.000	9630.000	10338.000
	Avg	11235.800	11237.560	**11238.000**	11237.120	**11238.000**	**11238.000**	**11238.000**	11237.560	10999.600	10854.360
	Best	11238.000	11238.000	**11238.000**	11238.000	**11238.000**	**11238.000**	**11238.000**	11238.000	11238.000	11227.000
	SD	4.491	2.200	**0.000**	3.046	**0.000**	**0.000**	**0.000**	2.200	421.918	291.277
	ER(%)	0.020	0.004	**0.000**	0.008	**0.000**	**0.000**	**0.000**	0.004	2.121	3.414
$KP1_{500}$	Worst	**28834.000**	**28834.000**	**28834.000**	**28834.000**	**28834.000**	**28834.000**	**28834.000**	**28834.000**	**26162.000**	**26299.000**
	Avg	28834.000	28834.920	**28839.520**	28836.760	28836.760	28834.920	28834.000	28834.920	27645.800	27706.840
	Best	28834.000	**28857.000**	**28857.000**	**28857.000**	**28857.000**	**28857.000**	28834.000	**28857.000**	28834.000	28732.000
	SD	**0.000**	4.600	10.025	7.628	7.628	4.600	0.000	4.600	717.042	521.223
	ER(%)	0.080	0.077	**0.061**	0.070	0.070	0.077	0.080	0.077	4.197	3.986
$KP1_{1000}$	Worst	54428.000	54421.000	54370.000	54264.000	54074.000	54174.000	**54471.000**	54412.000	43285.000	49605.000
	Avg	54490.400	54492.760	54459.960	54431.280	54374.320	54318.200	**54497.480**	54492.000	51187.520	51158.600
	Best	54503.000	54503.000	54503.000	54503.000	54503.000	54481.000	**54503.000**	54503.000	54503.000	52267.000
	SD	17.767	18.897	44.659	57.280	99.082	86.257	**10.252**	20.680	2137.998	633.896
	ER(%)	0.023	0.019	0.079	0.132	0.236	0.339	**0.010**	0.020	6.083	6.136
$KP1_{2000}$	Worst	**110547.000**	**110547.000**	109759.000	109741.000	109157.000	108871.000	**110547.000**	110547.000	100006.000	101332.000
	Avg	110561.280	110559.440	110443.120	110342.280	110325.680	109614.280	**110563.720**	110557.480	103291.880	103844.200
	Best	110580.000	110593.000	110578.000	110555.000	110578.000	110463.000	**110593.000**	110578.000	106561.000	106821.000
	SD	**13.446**	14.239	169.571	215.349	361.328	412.042	14.501	11.822	1588.221	1253.167
	ER(%)	0.058	0.059	0.164	0.256	0.271	0.914	**0.055**	0.061	6.629	6.130
$KP1_{5000}$	Worst	276035.000	275577.000	272608.000	272070.000	273258.000	269241.000	**276064.000**	276087.000	248542.000	251997.000
	Avg	276326.280	275988.960	274603.680	274691.360	275624.440	270658.960	**276352.000**	276340.320	257576.800	256767.800
	Best	276427.000	276379.000	276086.000	275981.000	276379.000	272292.000	276399.000	**276456.000**	275207.000	262301.000
	SD	99.317	213.646	974.295	932.261	795.265	898.678	**82.376**	94.163	5925.570	2424.521
	ER(%)	0.047	0.169	0.670	0.639	0.301	2.097	**0.038**	0.042	6.829	7.122
$KP1_{10000}$	Worst	562085.000	561082.000	555399.000	553338.000	556069.000	544244.000	**562155.000**	562124.000	503671.000	518746.000
	Avg	562870.120	561708.720	558590.880	559183.520	561349.400	547645.800	**563320.120**	563059.160	520903.560	523119.440
	Best	563483.000	562509.000	562673.000	561632.000	**563605.000**	551777.000	**563605.000**	**563605.000**	553393.000	528077.000
	SD	363.253	405.885	1770.224	2088.685	2367.066	1875.571	**428.449**	537.680	12073.405	2583.782
	ER(%)	0.138	0.344	0.897	0.792	0.408	2.839	**0.058**	0.104	7.583	7.190

Bold values indicate the best results.

determined previously. Finally, all mentioned before but the initialization process will be applied even the termination conditions are satisfied.

3.6. The binary improved variant of GBO (BIGBO)

Fig. 4 explains the steps of the binary GBO improved using the one-point crossover and mutation operator to produce another variant, namely BIGBO. Likewise for RFSO, GEO, and BO, are converted into binary variants, namely BRFSO, BGEO, and BBO using various transfer functions described before and integrated with the RI strategy to convert the infeasible solutions into feasible ones. Ultimately, those three algorithms were integrated with the one-point crossover and the mutation operators in the same way as BIHOA and BIGBO to produce other three variants, namely HIGEO, BIBO, and BIRFSO.

4. Experimental Results.

The proposed algorithms will be investigated in this section using uncorrelated, weakly-correlated, and strongly-correlated high-dimensional 01KP instances widely used in the literature with their characteristics described in Table 3 in terms of the instance ID (ID), the number of dimensions (D), the knapsack capacity (Capacity), and the optimal-known solution (Opt) (Moradi et al., 2021; Ezugwu, Pillay, Hirasen, Sivanarain, & Govender, 2019). The rest of this section is as follows:

- Section 4.1: shows the parameter settings and the performance metrics.
- Section 4.2: investigates the various transfer functions.
- Section 4.3: describes the outcomes on uncorrelated high-dimensional 01KP instances.
- Section 4.4: describes the outcomes on weakly-correlated high-dimensional 01KP instances.
- Section 4.5: describes the outcomes on strongly-correlated high-dimensional 01KP instances.

4.1. Performance Metrics and parameter settings

Each proposed algorithm is executed 25 independent trials on each instance out of 21 instances described before using the same environmental conditions, a population size (N) of 30, and a number of function evaluations of $200 * D$. Then the obtained maximum profits have been analyzed for each algorithm using six statistical performance metrics: the best, worst, average (Avg), standard deviation (SD), CPU time, and the error rate between the average obtained profit and the optimal-known one according to Eq. 42. More than that, to show graphically the difference between the algorithms, the boxplot was used to compare the outcomes obtained by the various observed algorithms. Additionally, the convergence speed was graphically depicted to show the accelerate between the algorithms.

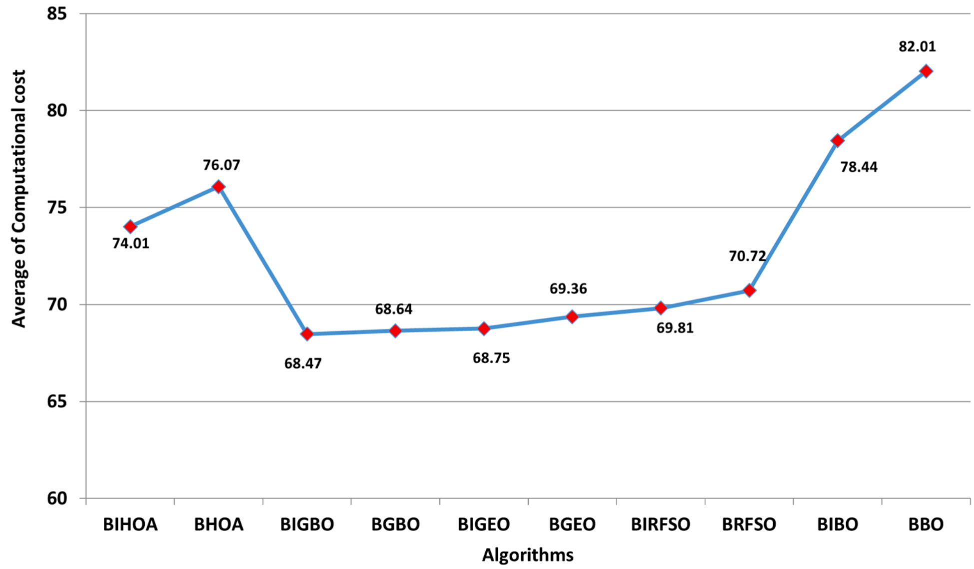


Fig. 6. Comparison among algorithms on uncorrelated instances.

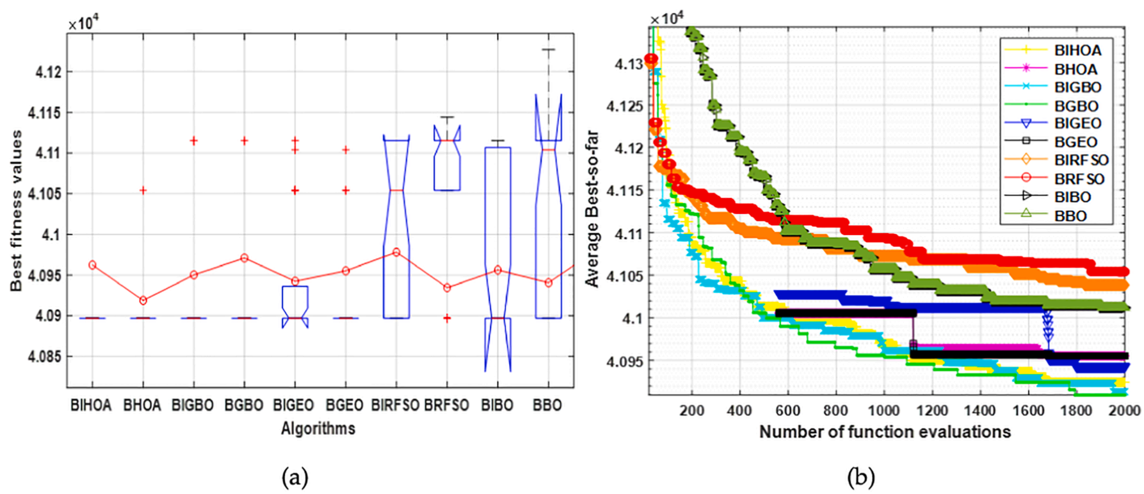


Fig. 7. Comparison on uncorrelated KP_{100} instance.

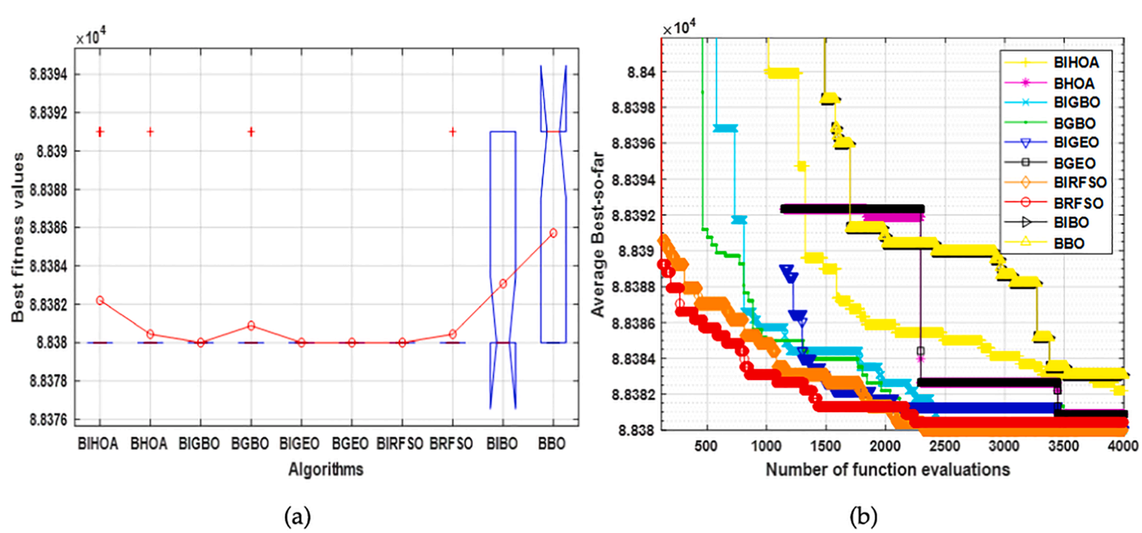
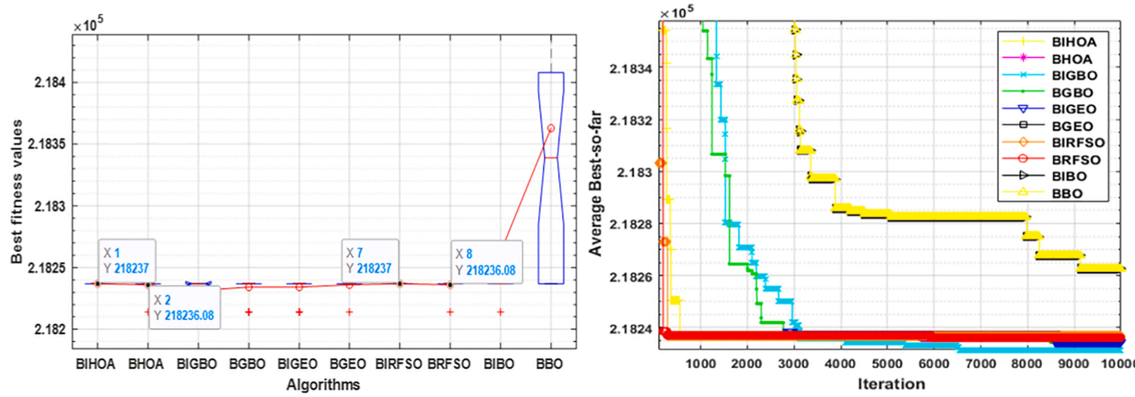
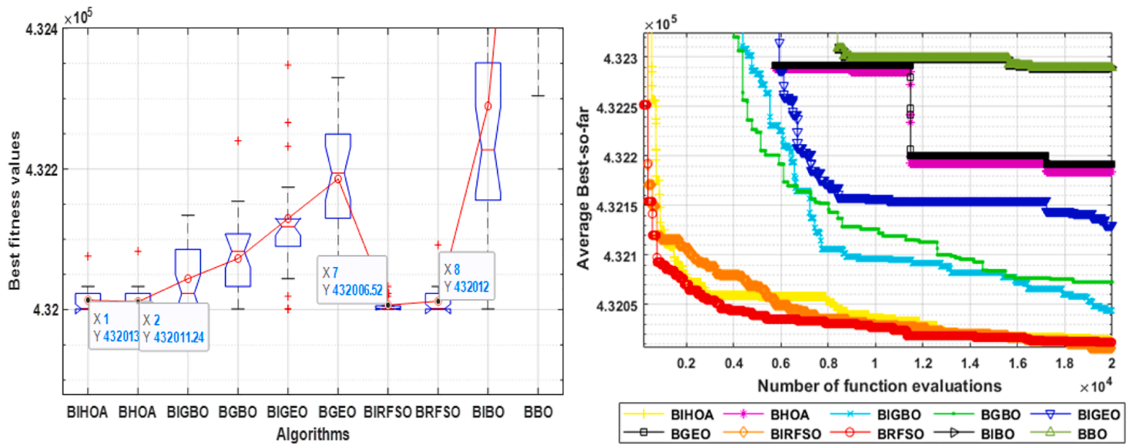


Fig. 8. Comparison on uncorrelated KP_{200} instance.



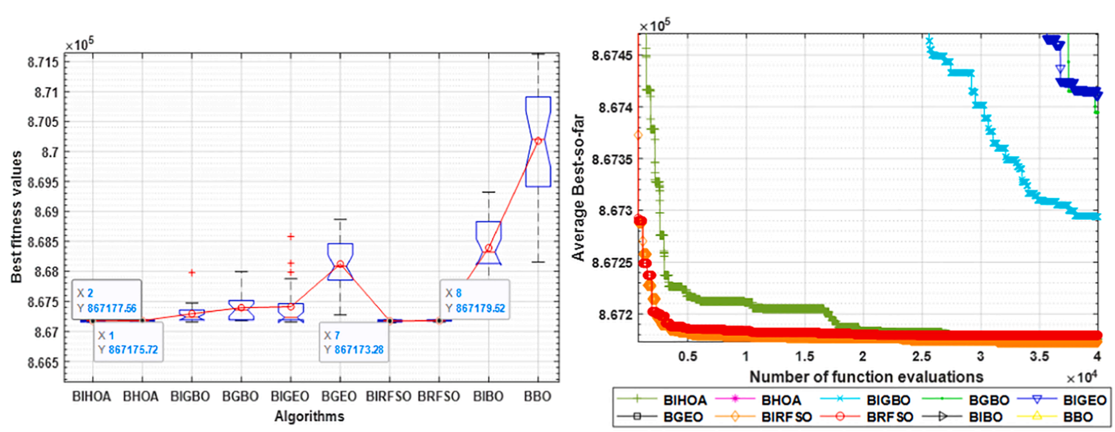
(a) (b)

Fig. 9. Comparison on uncorrelated KP_{500} instance.



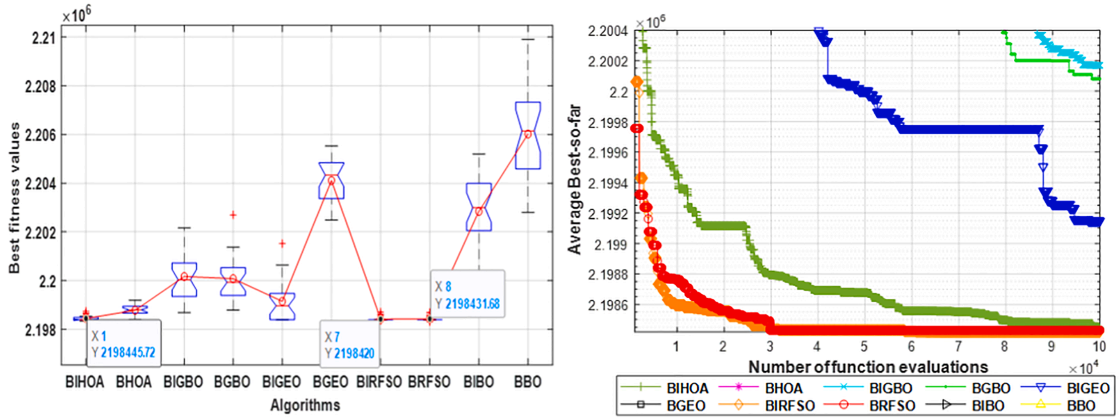
(a) (b)

Fig. 10. Comparison on uncorrelated KP_{1000} instance.



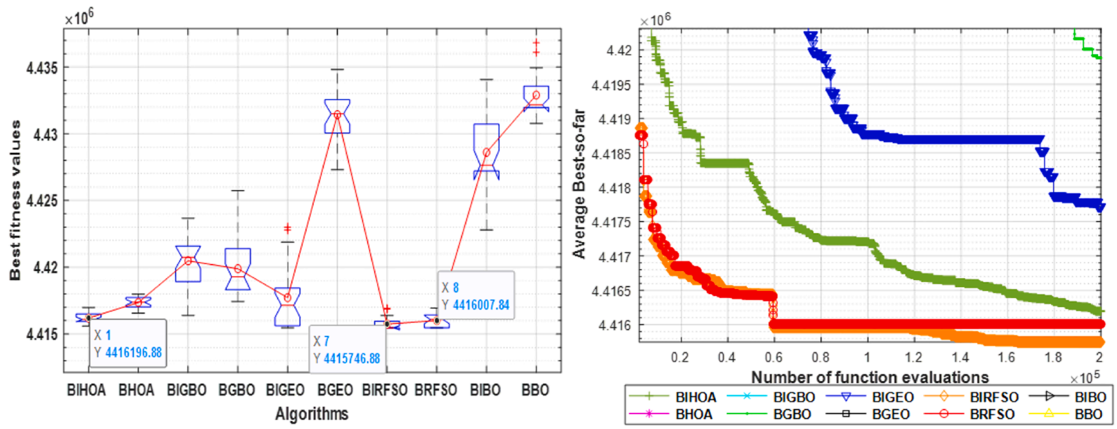
(a) (b)

Fig. 11. Comparison on uncorrelated KP_{2000} instance.



(a) (b)

Fig. 12. Comparison on uncorrelated KP_{5000} instance.



(a) (b)

Fig. 13. Comparison on uncorrelated KP_{10000} instance.

$$ER(\%) = \frac{Opt - Avg}{Opt} * 100 \tag{42}$$

4.2. Experiment 1: Investigation of various transfer functions

To find the transfer function affecting positively on the performance of each observed algorithm, extensive experiments have been done by running each algorithm with all transfer functions on the uncorrelated KP_{500} instance 30 independent trials and depicting the average best-so-far fitness values obtained within these runs in Fig. 5 (the ids concatenated with the algorithms' names refer to the transfer function employed as shown in Table 1), which shows that the best transfer functions for BIHOA, BIGBO, BIGEO, BIRFSO, and BIBSO respectively are the ones with the following ids: 7, 1, 1, 6, and 1.

4.3. Experiment 2: uncorrelated high dimensional 01KP instances

After extracting the relevant transfer function for each algorithm observed in this paper, it is the turn to compare all those algorithms with each other under various statistical analyses to see which of them could reach better profits. Therefore, on the uncorrelated instances, each algorithm is executed 25 independent times, and the various statistical analyses mentioned before are exposed in Table 4, which illustrates the superiority of BIRFSO on the instances with dimensions higher than 500; meanwhile, the converged performance for the other instances among the algorithms has attended. Broadly speaking, according to Table 4, BIGBO, BIGEO, BGEO, and BIRFSO could fulfill the optimal solution of $KP1_{200}$ in all independent runs, while both $KP1_{100}$ and $KP1_{500}$ could be solved more accurately using BGBO, and BIGBO, respectively. For the rest of the instances (higher than 500), BIRFO proves its proficiency for reaching better outcomes in comparison to all the others. Furthermore, Fig. 6 is presented to show the average of the computational cost consumed by each algorithm until implementing all the uncorrelated instances, which confirms that BIRFSO could come true reasonable time for well solving those instances compared to the others, while BBO needs the highest computational cost to tackle those instances. As a result for any uncorrelated instance with a number of dimensions higher than 500, BIRFSO is a strong alternative to all the existing ones since it could come true better outcomes in a reasonable time.

As a new attempt to appear the performance of the algorithms, Figs. 7–13 is below pictured to depict the boxplot of the fitness values, and the averaged convergence speed obtained by each algorithm on each uncorrelated instance. From those figures, it is notified that the performance of the algorithms are approximately converged until the KP_{500} instance; however, for the others, BIRFSO appears superior performance in terms of the final accuracy and the convergence speed. More speaking, Fig. 7 presented the boxplot of the outcomes obtained by different algorithms for tackling the KP_{100} instance; this figure is evident that BIRFSO is approximately converged with BHOA as shown by the red line drawn inside the boxplot figure to determine the average of the

Table 5
Comparison on the weakly uncorrelated instance.

Id		BIHOA	BHOA	BIGBO	BGBO	BIGEO	BGEO	BIRFSO	BRFSO	BIBO	BBO
$KP1_{100}$	Worst	1502.000	**1514.000**	1502.000	**1514.000**	**1514.000**	**1514.000**	**1514.000**	**1514.000**	1363.000	1276.000
	Avg	1513.520	**1514.000**	1513.520	**1514.000**	**1514.000**	**1514.000**	**1514.000**	**1514.000**	1486.840	1466.920
	Best	**1514.000**	**1514.000**	**1514.000**	**1514.000**	**1514.000**	**1514.000**	**1514.000**	**1514.000**	1512.000	1512.000
	SD	2.400	**0.000**	2.400	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	45.978	66.032
	ER(%)	0.032	**0.000**	0.032	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	1.794	3.110
$KP1_{200}$	Worst	1623.000	1629.000	1627.000	1627.000	**1629.000**	**1634.000**	**1629.000**	1623.000	1348.000	1501.000
	Avg	1632.200	1633.800	1632.680	1633.520	**1633.760**	**1634.000**	**1633.800**	1633.560	1597.040	1589.200
	Best	**1634.000**	**1634.000**	**1634.000**	**1634.000**	**1634.000**	**1634.000**	**1634.000**	**1634.000**	**1634.000**	**1634.000**
	SD	3.536	1.000	2.340	1.686	1.012	**0.000**	1.000	2.200	68.498	47.174
	ER(%)	0.110	0.012	0.081	0.029	0.015	**0.000**	0.012	0.027	2.262	2.742
$KP1_{500}$	Worst	**4552.000**	**4552.000**	**4552.000**	**4552.000**	**4552.000**	**4552.000**	**4552.000**	**4552.000**	4218.000	4269.000
	Avg	4555.040	4555.680	**4555.880**	4554.960	4555.720	4555.280	4555.680	4555.360	4412.240	4434.640
	Best	4556.000	4556.000	**4559.000**	4559.000	4557.000	4557.000	4556.000	4556.000	4554.000	4551.000
	SD	1.744	1.108	**1.333**	1.947	1.242	1.696	1.108	1.497	103.893	73.167
	ER(%)	0.240	0.226	**0.222**	0.242	0.225	0.235	0.226	0.233	3.367	2.877
$KP1_{1000}$	Worst	**9046.000**	**9046.000**	**9046.000**	**9046.000**	**9046.000**	9036.000	**9046.000**	**9046.000**	8544.000	8535.000
	Avg	9049.080	**9050.320**	9048.920	9048.240	9048.040	9046.240	**9049.800**	9049.640	8822.760	8784.320
	Best	**9051.000**	**9051.000**	9051.000	**9051.000**	**9051.000**	**9051.000**	**9051.000**	**9051.000**	9046.000	8951.000
	SD	2.235	**1.600**	2.290	2.278	2.475	3.382	1.979	2.039	123.404	98.480
	ER(%)	0.032	**0.019**	0.034	0.042	0.044	0.064	0.024	0.026	2.532	2.957
$KP1_{2000}$	Worst	**18038.000**	**18038.000**	18001.000	18000.000	17979.000	17946.000	**18038.000**	18038.000	17140.000	17270.000
	Avg	18043.600	**18043.960**	18033.880	18027.520	18025.760	17989.160	**18043.960**	18043.800	17505.720	17527.200
	Best	18046.000	**18047.000**	**18047.000**	18045.000	18046.000	18033.000	18046.000	18046.000	18019.000	17856.000
	SD	3.215	3.116	10.982	12.484	20.185	22.451	**2.791**	2.972	201.672	146.736
	ER(%)	0.041	**0.039**	0.095	0.130	0.140	0.343	**0.039**	0.040	3.021	2.902
$KP1_{5000}$	Worst	44339.000	44338.000	44116.000	44190.000	44004.000	43917.000	**44351.000**	44349.000	42527.000	42767.000
	Avg	44349.920	44348.800	44276.040	44292.480	44275.880	44065.080	**44351.280**	44351.160	43063.760	43128.720
	Best	44353.000	44354.000	44351.000	44351.000	44353.000	44200.000	**44353.000**	**44353.000**	44201.000	43467.000
	SD	3.174	4.133	58.943	37.364	108.617	74.404	**0.614**	0.800	400.049	186.441
	ER(%)	0.014	0.016	0.180	0.143	0.181	0.656	**0.011**	0.011	2.913	2.767
$KP1_{10000}$	Worst	90104.000	90041.000	89404.000	89551.000	89237.000	88939.000	90136.000	**90137.000**	86344.000	86679.000
	Avg	90166.360	90096.240	89812.840	89883.040	89969.760	89196.800	**90183.920**	90182.000	87536.520	87259.120
	Best	90200.000	90143.000	90195.000	90105.000	**90201.000**	89502.000	90200.000	90200.000	89576.000	87850.000
	SD	24.988	25.973	218.455	150.456	275.339	149.303	**17.851**	20.203	1056.108	356.670
	ER(%)	0.042	0.119	0.434	0.356	0.260	1.117	**0.022**	0.024	2.957	3.265

Bold values indicate the best results.

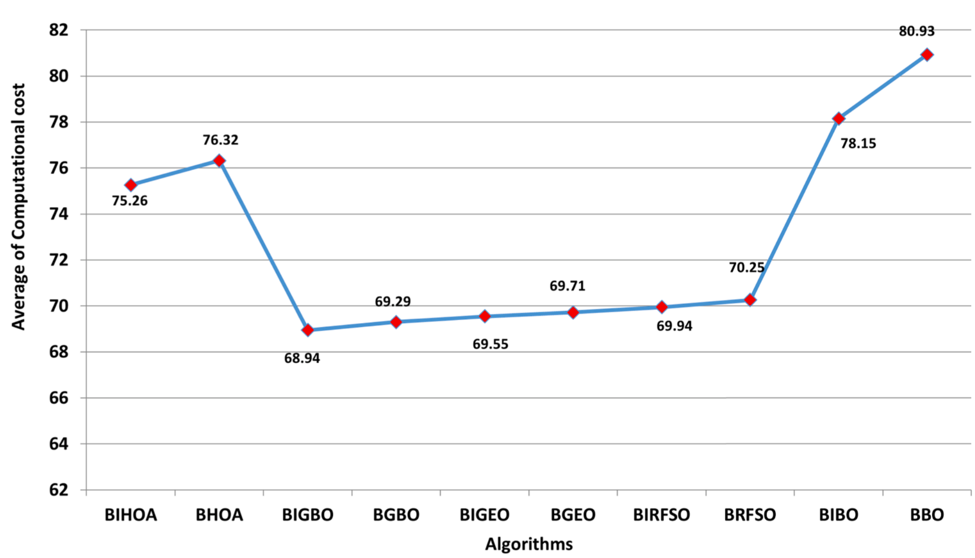
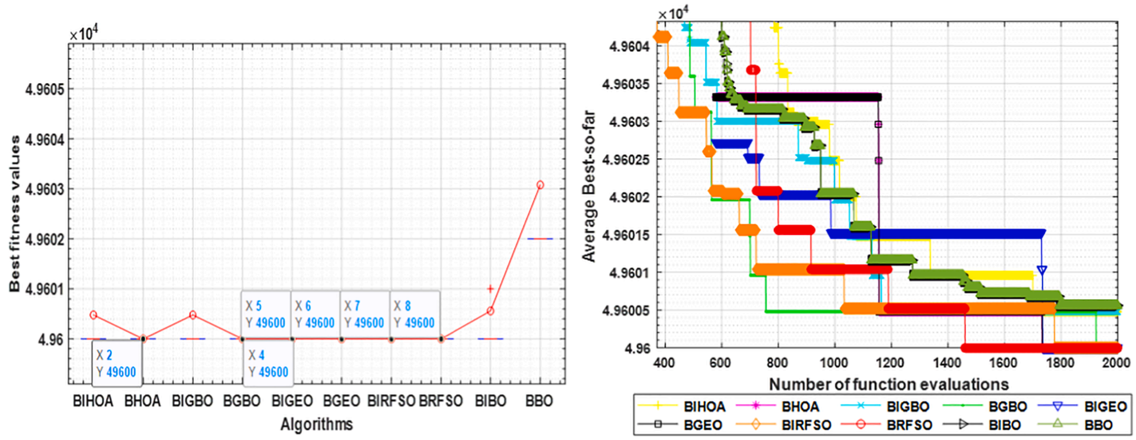
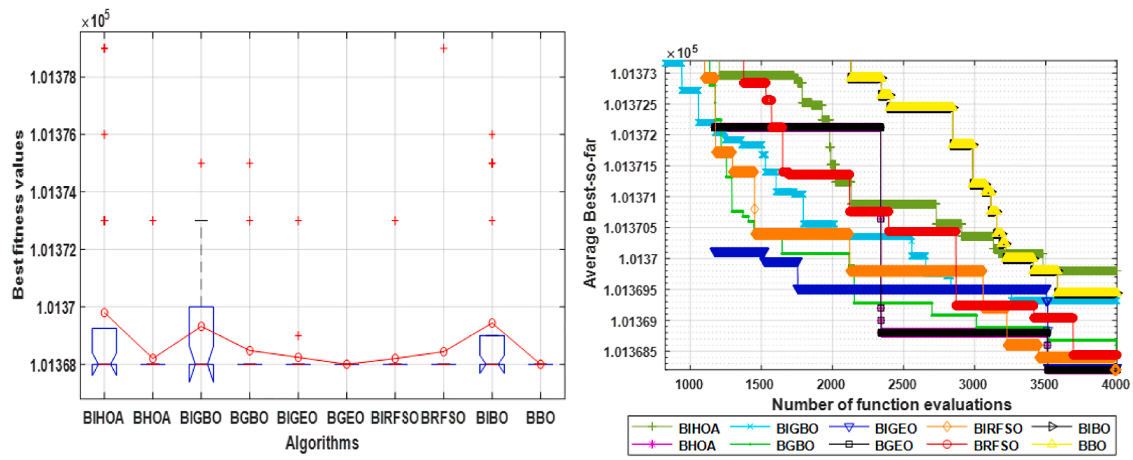


Fig. 14. Comparison on the weakly-correlated instances.



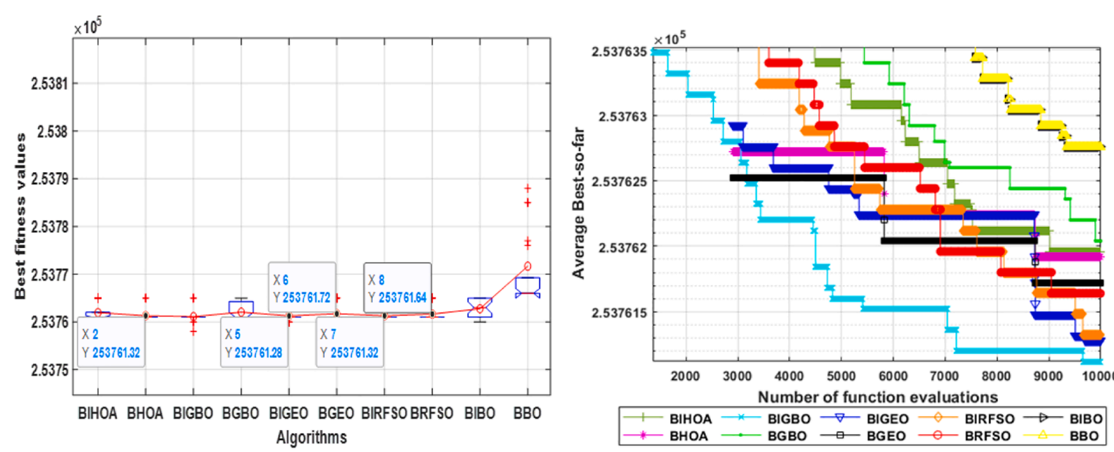
(a) (b)

Fig. 15. Comparison on weakly correlated KP_{100} instance.



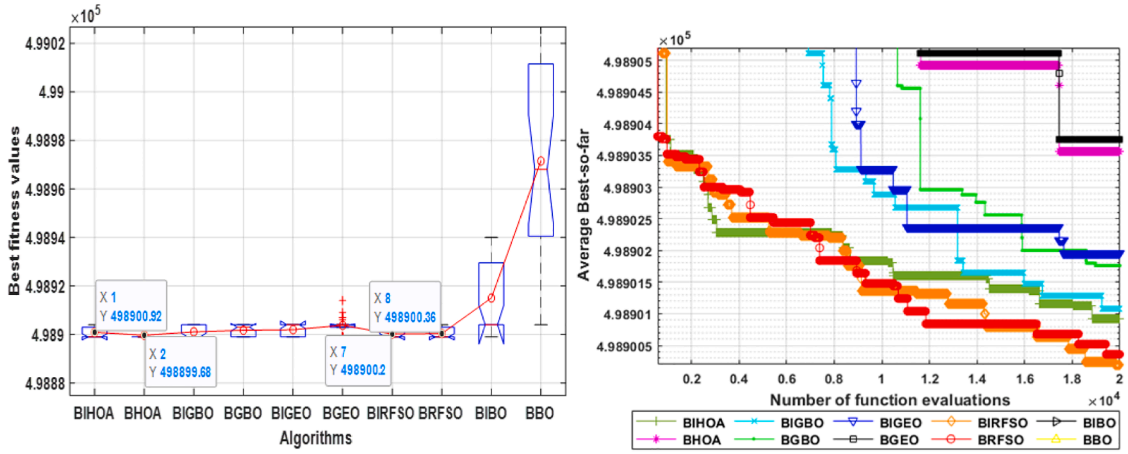
(a) (b)

Fig. 16. Comparison on weakly correlated KP_{200} instance.



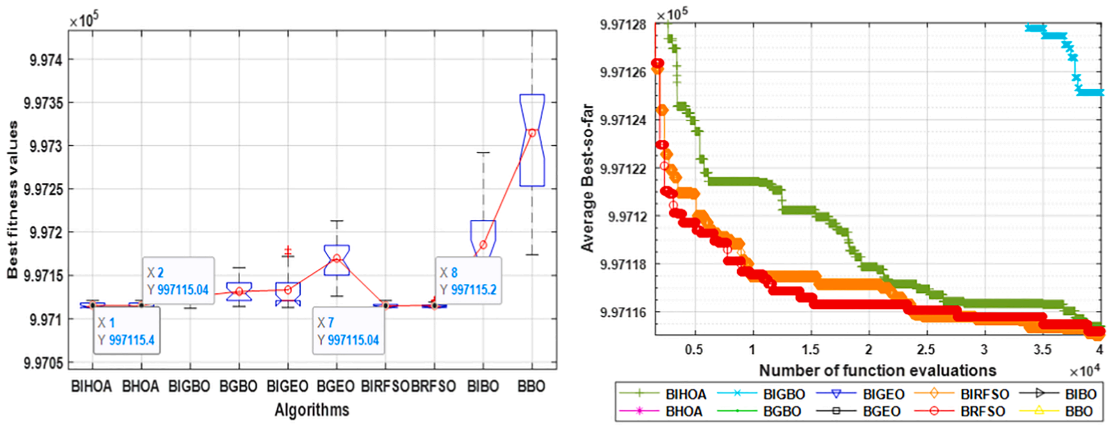
(a) (b)

Fig. 17. Comparison on weakly correlated KP_{500} instance.



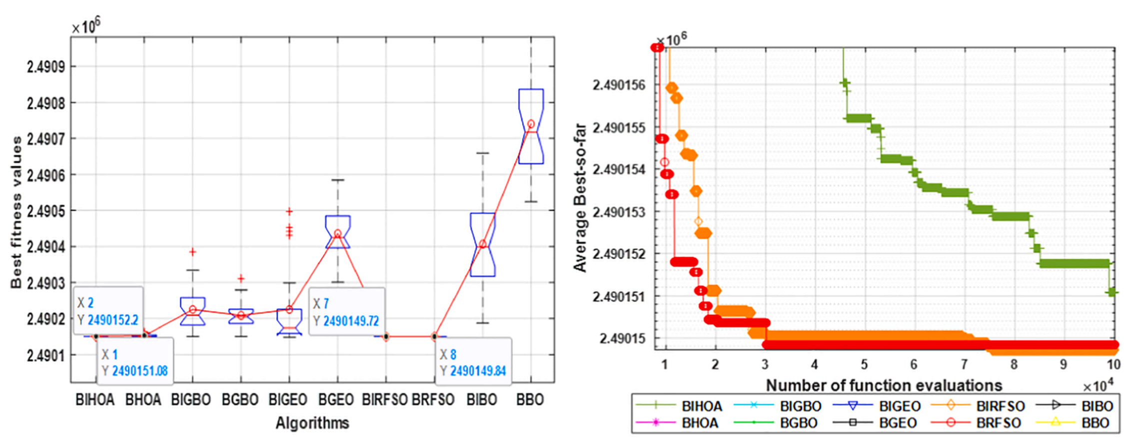
(a) (b)

Fig. 18. Comparison on weakly correlated KP_{1000} instance.



(a) (b)

Fig. 19. Comparison on weakly correlated KP_{2000} instance.



(a) (b)

Fig. 20. Comparison on weakly correlated KP_{5000} instance.

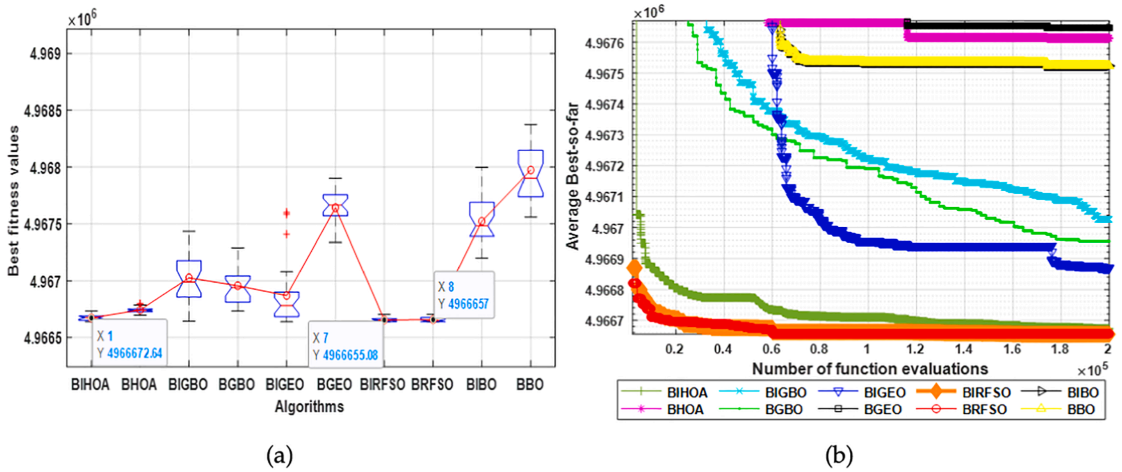


Fig. 21. Comparison on weakly correlated KP_{10000} instance.

Table 6
Comparison of the strongly correlated instances.

Id		BIHOA	BHOA	BIGBO	BGBO	BIGEO	BGEO	BIRFSO	BRFSO	BIBO	BBO
$KP1_{100}$	Worst	2381.000	2381.000	2390.000	**2396.000**	2390.000	2390.000	2390.000	2381.000	2203.000	2090.000
	Avg	2392.960	2394.440	2395.200	**2396.200**	2393.920	2394.360	2391.440	2389.880	2348.000	2294.600
	Best	**2397.000**	**2397.000**	**2397.000**	**2397.000**	**2397.000**	**2397.000**	2396.000	2396.000	2396.000	2390.000
	SD	4.057	3.820	2.345	**0.408**	3.013	2.782	2.615	2.205	60.614	82.909
	ER(%)	0.169	0.107	0.075	**0.033**	0.128	0.110	0.232	0.297	2.044	4.272
$KP1_{200}$	Worst	2693.000	2694.000	2695.000	2693.000	**2694.000**	2693.000	**2694.000**	2689.000	2501.000	2524.000
	Avg	2696.480	2696.640	2696.800	2696.480	**2696.800**	**2696.800**	**2696.880**	2696.560	2656.080	2647.960
	Best	**2697.000**	**2697.000**	**2697.000**	**2697.000**	**2697.000**	**2697.000**	**2697.000**	**2697.000**	**2697.000**	**2697.000**
	SD	1.159	0.907	0.500	1.085	0.645	0.816	**0.600**	1.685	54.951	47.020
	ER(%)	0.019	0.013	0.007	0.019	0.007	0.007	**0.004**	0.016	1.517	1.818
$KP1_{500}$	Worst	7114.000	7114.000	7112.000	7110.000	7113.000	7109.000	7113.000	**7115.000**	6616.000	6711.000
	Avg	7115.800	7115.720	7116.120	7115.640	7116.000	7115.600	7115.880	**7116.360**	6892.840	6875.320
	Best	**7117.000**	**7117.000**	**7117.000**	**7117.000**	**7117.000**	**7117.000**	**7117.000**	**7117.000**	**7117.000**	7016.000
	SD	0.957	0.980	1.269	1.997	1.155	1.780	1.054	**0.810**	147.798	86.012
	ER(%)	0.017	0.018	0.012	0.019	0.014	0.020	0.016	**0.009**	3.150	3.396
$KP1_{1000}$	Worst	14387.000	**14388.000**	14290.000	14286.000	14284.000	14284.000	14386.000	14386.000	13394.000	13587.000
	Avg	14389.560	**14389.720**	14380.040	14381.960	14366.120	14344.760	14389.715	14389.600	13801.720	13822.120
	Best	**14390.000**	**14390.000**	**14390.000**	**14390.000**	**14390.000**	**14390.000**	**14390.000**	**14390.000**	14290.000	14187.000
	SD	0.870	**0.597**	27.259	20.274	40.069	47.171	0.891	0.913	253.038	162.283
	ER(%)	0.003	**0.002**	0.069	0.056	0.166	0.314	**0.002**	0.003	4.088	3.946
$KP1_{2000}$	Worst	28915.000	**28917.000**	28805.000	28808.000	28708.000	28617.000	28916.000	28914.000	26643.000	27412.000
	Avg	28918.000	28918.400	28894.320	28895.760	28897.640	28733.040	**28918.480**	28917.920	27756.800	27725.200
	Best	**28919.000**	**28919.000**	**28919.000**	28918.000	**28919.000**	28813.000	**28919.000**	**28919.000**	28818.000	28314.000
	SD	1.190	0.816	41.619	32.711	50.377	60.859	**0.872**	1.412	442.974	243.652
	ER(%)	0.003	**0.002**	0.085	0.080	0.074	0.643	**0.002**	0.004	4.019	4.128
$KP1_{5000}$	Worst	72405.000	72392.000	71784.000	72000.000	71393.000	71004.000	**72486.000**	72400.000	67714.000	67805.000
	Avg	72492.920	72437.080	72224.440	72229.760	72292.560	71460.600	**72500.200**	72484.760	69005.200	69033.760
	Best	72505.000	72505.000	72500.000	72503.000	72502.000	71704.000	**72505.000**	**72505.000**	71893.000	69689.000
	SD	26.831	49.245	150.719	134.701	268.739	182.883	**4.924**	33.130	995.803	425.771
	ER(%)	0.017	0.094	0.387	0.380	0.293	1.440	**0.007**	0.028	4.827	4.788
$KP1_{10000}$	Worst	146589.000	146413.000	144910.000	145416.000	144907.000	143619.000	**146618.000**	146606.000	137428.000	138394.000
	Avg	146767.360	146525.600	146118.960	146001.080	146349.720	144048.560	**146851.720**	146781.200	139687.400	139670.520
	Best	146915.000	146718.000	146596.000	146704.000	**146918.000**	144413.000	**146918.000**	146888.000	144311.000	140760.000
	SD	72.804	86.892	484.516	325.232	544.334	212.322	**85.113**	108.532	1699.460	534.527
	ER(%)	0.103	0.268	0.545	0.625	0.387	1.954	**0.046**	0.094	4.922	4.934

Bold values indicate the best results.

outcomes. Likewise, for KP_{200} and KP_{500}, BIRFSO could be significantly competitive compared to some of the other metaheuristic algorithms. for the other instances, it proved that it is the best for tackling any uncorrelated instances with dimensions greater than 500.

4.4. Experiment 2: Weakly-correlated high dimensional 01KP instances

In the above section, it was proved that RIFSO could be competitive for the instance with dimensions up to 500, and superior for the other instances in a comparison made to see the efficiency of the different observed algorithms. however, that's not enough to confirm its

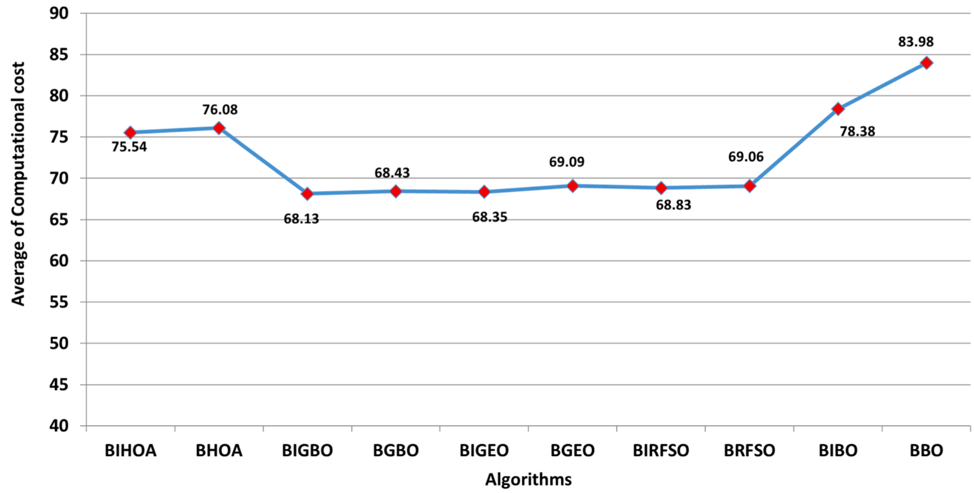
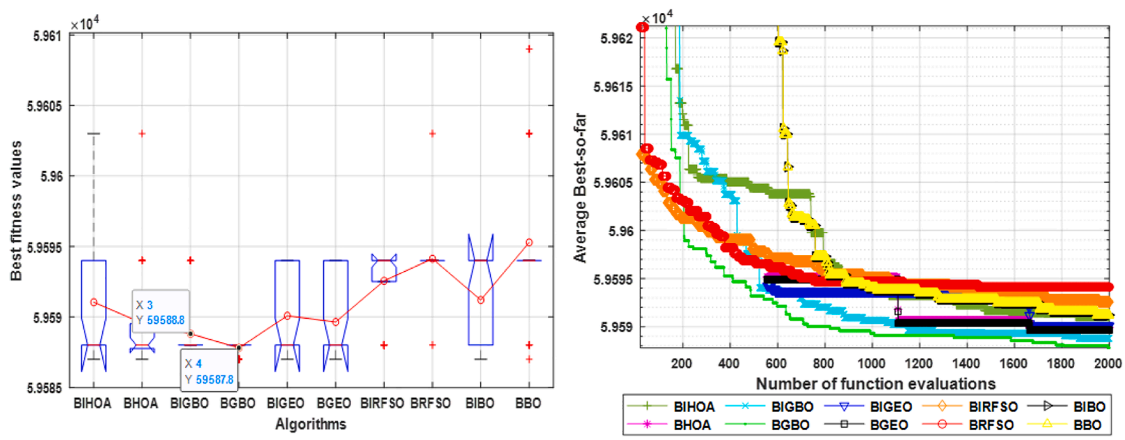
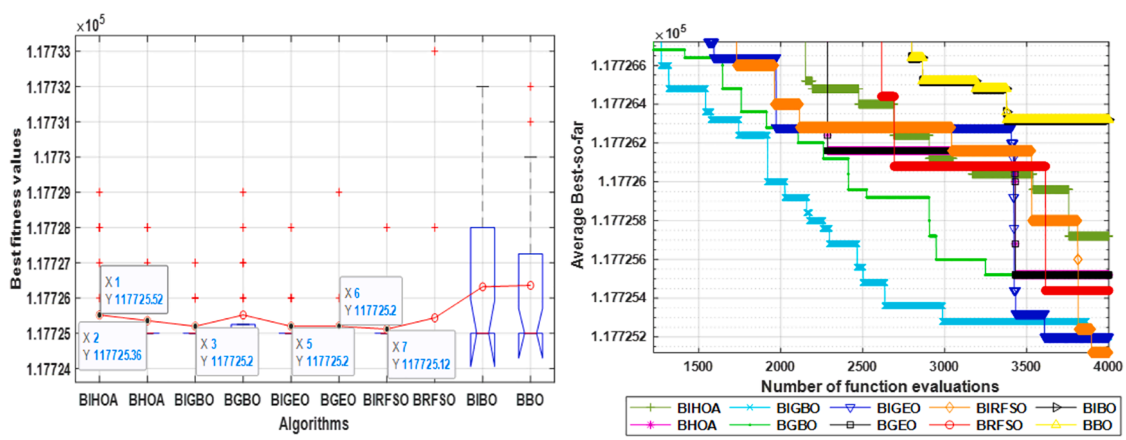


Fig. 22. Comparison of the strongly-correlated instances.



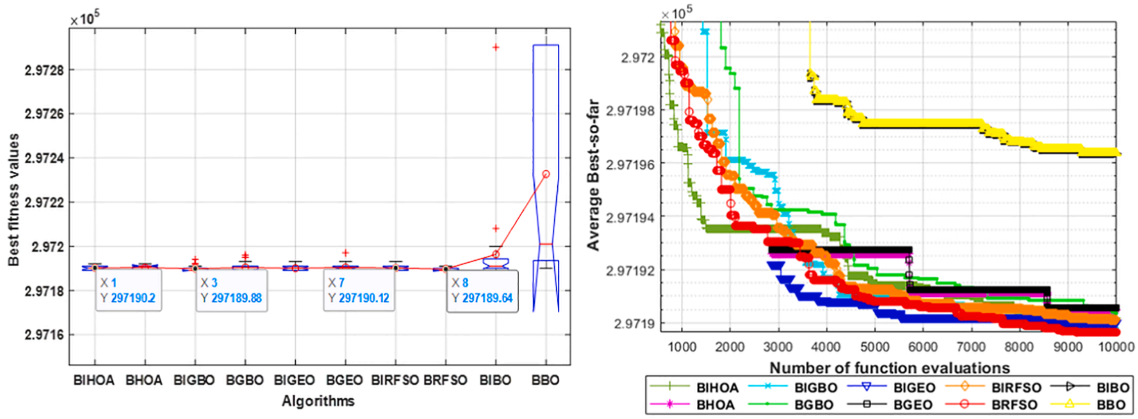
(a) (b)

Fig. 23. Comparison on strongly correlated KP_{100} instance.



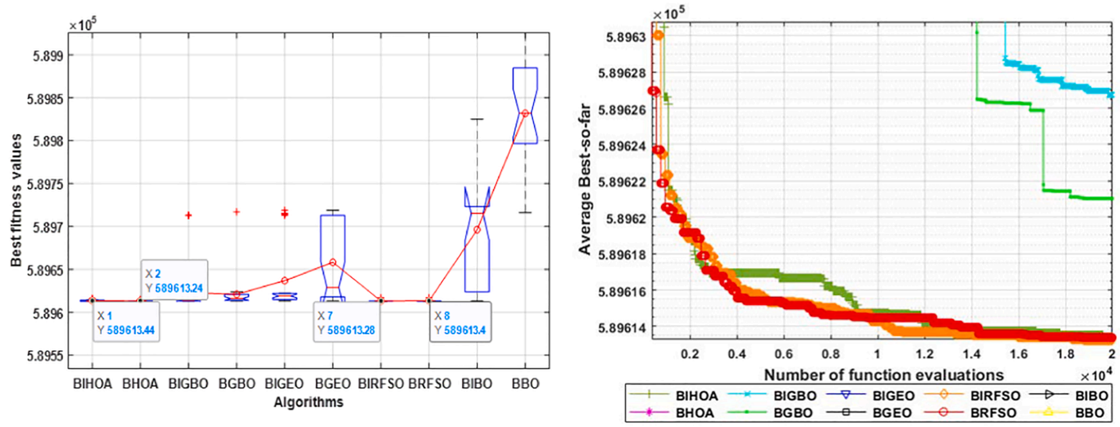
(a) (b)

Fig. 24. Comparison on strongly correlated KP_{200} instance.



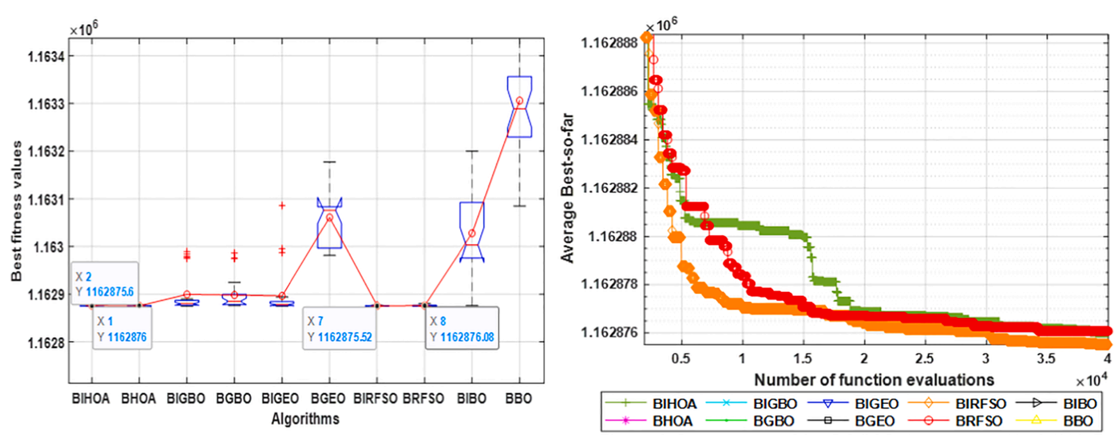
(a) (b)

Fig. 25. Comparison on strongly correlated KP_{500} instance.



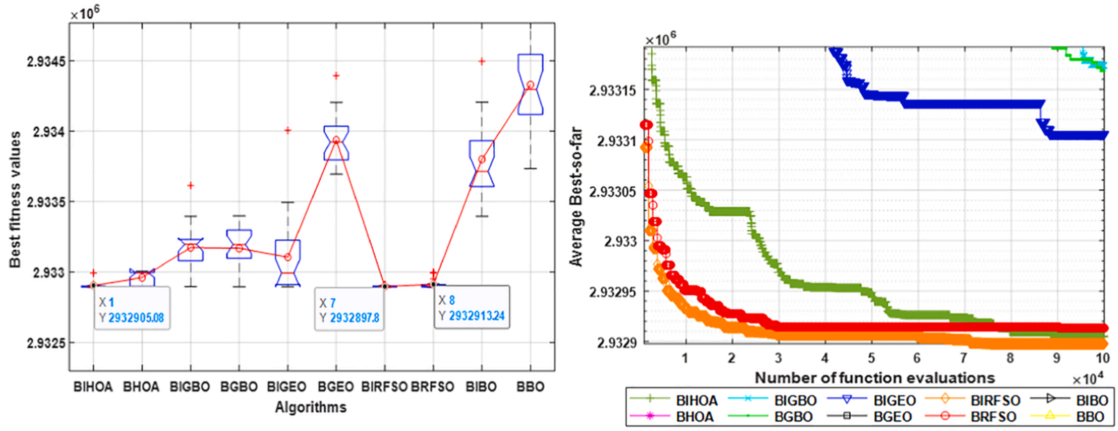
(a) (b)

Fig. 26. Comparison on strongly correlated KP_{1000} instance.



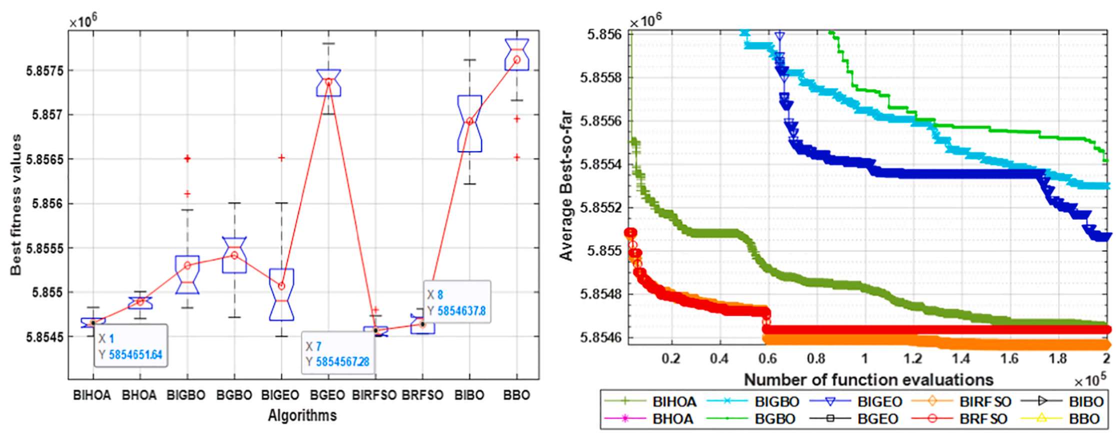
(a) (b)

Fig. 27. Comparison on strongly correlated KP_{2000} instance.



(a) (b)

Fig. 28. Comparison on strongly correlated KP_{5000} instance.



(a) (b)

Fig. 29. Comparison on strongly correlated KP_{10000} instance.

superiority, therefore another benchmark with 7 weakly correlated instances is addressed in this section to see the sensitivity of the algorithms. Table 5 is presented to exhibit the results of analyzing the best profits obtained within 25 independent runs by each algorithm on the weakly-correlated instance. As a result of observing this table, the superiority of BIRFSO is confirmed on the instances with dimensions greater than 1000, and the competitivity among the algorithms for the other instances. Generally speaking, in Table 5, it is noticeable that BHOA, BGBO, BIGEO, BIRFSO, and BRFSO could reach the optimal outcome for $KP2_{100}$ in all runs performed independently, and $KP2_{200}$ was optimally solved using also BGEO. However, unfortunately, for the other instances, the algorithms could not reach the desired outcomes, but some of them were so near, for instance, BIGBO could reach an average of 4555 for $KP2_{500}$ which is so near to its optimal outcome: 4559. Also, it is observable from the same table that BIRFSO could be superior to the other compared ones when the number of dimensions exceeds 1000. In addition, Fig. 14 shows the effectiveness of BIRFSO in achieving a reasonable consumption time compared with the high accuracy of the obtained outcomes.

Figs. 15–21 are below used to show the effectiveness of the algorithms in terms of the boxplot of the fitness values and the averaged convergence speed, which show that the performance of BIRFSO is almost converged for KP_{100},KP_{200}, and KP_{500} instances, and superior on the other instances.

4.5. Experiment 3: Strongly-correlated high dimensional 01KP instances.

Table 6 is below presented to display the analysis of the outcomes obtained by the algorithms on the strongly correlated 01KP instances. Based on the outcomes presented in this table, BIRFSO can be the best for the instances having dimensions higher than 1000, while its performance on the others is significantly competitive with the other algorithms; BGBO, as the best one on the $KP1_{100}$ instance, could fulfill an average value of 2396 for $KP1_{100}$ while BIRFSO fulfilled an average of 2391; BIRFSO could be the best for $KP1_{200}$ with an average of 2696.880, while the second-best one had an average value of 2696.800; BRFSO could be the best for $KP1_{500}$ with an average of 7116.360, while BIGBO as the second-best one had an average value of 7116.120; BHOA could be the best for $KP1_{1000}$ with an average of 14389.720, while the second-best one: BIRFSO had an average value of 14389.715; for the rest instances, BIRFSO could be superior to the others. About the computational cost required by each algorithm until finishing the optimization process on all the strongly correlated instances, Fig. 22 is presented to show that BIRFSO can occupy the fifth rank after BIGBO, BIGEO, BGBO, and GEO which have poor performance compared to BIRFSO and hence BIRFSO can accomplish better outcomes in a reasonable computational time.

Figs. 23–29 which involves the boxplot and the convergence speed curve for the fitness values are presented to show which algorithm is faster and better. Inspecting these figures shows that BIRFSO can be the best in terms of the convergence curve and the fitness value on KP_{200}, KP_{2000},KP_{5000}, and KP_{10000}. For the other instances, BGBO can come true better convergence and average fitness value on KP_{100} instance, while BRFSO and BHOA can be the best on KP_{500} and KP_{1000}, respectively.

5. Conclusion and future work

Recently, several meta-heuristic optimization algorithms have been proposed for tackling the optimization problems, such as horse herd optimization algorithm (HOA), gradient-based optimizer (GBO), Bonobo optimizer (BO), golden eagle optimizer (GEO), and red fox search optimizer (RFSO); however, their performance for the discrete optimization problems such as the classical 0–1 knapsack problem have not been addressed. Therefore, those five algorithms are transformed into binary variants using various transfer functions to be able to solve the knapsack problem. The knapsack problem is a discrete problem to find the optimal selection of items that will maximize the profit by satisfying the knapsack capacity constraints. Unfortunately, some obtained solutions are infeasible because they could not satisfy the knapsack capacity constraint. So, the fixing and improvement strategy to convert those infeasible solutions into feasible ones, then improve them. Finally, to further improve the performance of those algorithms for tackling especially the high dimensional 01KP instances, the one-point crossover and mutation operators are effectively hybridized to explore other solutions intractable by those algorithms alone. Finally, those various variants have been validated using 21 widely-used uncorrelated, weakly-correlated, and strongly-correlated 01KP instances with several dimensions up to 10000, and compared with each other using various performance measures to show which one is more superior.

Our future work involves observing the performance of this algorithm for other kinds of the knapsack problems such as multidimensional knapsack, discount 0–1 knapsack, and the set union knapsack, and the quadratic knapsack problems.

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

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

Abdel-Basset, M., Mohamed, R., & Mirjalili, S. (2021). A binary equilibrium optimization algorithm for 0–1 knapsack problems. *Computers & Industrial Engineering, 151*, 106946.

Adamuthe, A. C. Sale, V. N. & Mane, S. U. (2020). "Solving single and multi-objective 01 knapsack problem using harmony search algorithm," Journal of Scientific Research, vol. 64, no. 1, 2020.

Ahmadianfar, I., Bozorg-Haddad, O., & Chu, X. (2020). Gradient-based optimizer: A new metaheuristic optimization algorithm. *Information Sciences, 540*, 131–159.

Ali, A. B., Luque, G., & Alba, E. (2020). An efficient discrete pso coupled with a fast local search heuristic for the dna fragment assembly problem. *Information Sciences, 512*, 880–908.

Azad, M. A. K., Rocha, A. M. A. & Fernandes, E.M. (2014). "A simplified binary artificial fish swarm algorithm for 0–1 quadratic knapsack problems." Journal of Computational and Applied mathematics, vol. 259 (pp. 897–904).

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

Bhattacharjee, K. K., & Sarmah, S. P. (2015). A binary cuckoo search algorithm for knapsack problems. In *2015 International Conference on Industrial Engineering and Operations Management (IEOM)* (pp. 1–5). IEEE.

Bhattacharjee, K. K., & Sarmah, S. P. (2015). A binary firefly algorithm for knapsack problems. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 73–77). IEEE.

Bhattacharya, S., Maddikunta, P. K. R., Kaluri, R., Singh, S., Gadekallu, T. R., Alazab, M., Tariq, U., et al. (2020). A novel pca-firefly based xgboost classification model for intrusion detection in networks using gpu. *Electronics, 9*(2), 219.

Chou, J.-S., & Truong, D.-N. (2021). A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. *Applied Mathematics and Computation, 389*, 125535.

Das, A. K., & Pratihar, D. K. (2019). A new bonobo optimizer (bo) for real-parameter optimization. In *2019 IEEE Region 10 Symposium (TENSYMP)* (pp. 108–113). IEEE.

Ezugwu, A. E., Pillay, V., Hirasen, D., Sivanarain, K., & Govender, M. (2019). A comparative study of meta-heuristic optimization algorithms for 0–1 knapsack problem: Some initial results. *IEEE Access, 7*, 43979–44001.

Faramarzi, A., Heidarinejad, M., Mirjalili, S., & Gandomi, A. H. (2020). Marine predators algorithm: A nature-inspired metaheuristic. *Expert Systems with Applications, 152*, 113377.

Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. *Knowledge-Based Systems, 191*, 105190.

Gadekallu, T. R., Alazab, M., Kaluri, R., Maddikunta, P. K. R., Bhattacharya, S., Lakshmanna, K., & Parimala, M. (2021). Hand gesture classification using a novel cnn-crow search algorithm. *Complex & Intelligent Systems*, 1–14.

Gadekallu, T. R., Rajput, D. S., Reddy, M. P. K., Lakshmanna, K., Bhattacharya, S., Singh, S., Jolfaei, A., & Alazab, M. (2020). A novel pca–whale optimization-based deep neural network model for classification of tomato plant diseases using gpu. *Journal of Real-Time Image Processing*, 1–14.

Guldan, B. (2007). *Heuristic and exact algorithms for discounted knapsack problems*. Germany: University of Erlangen-Nürnberg.

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

Hashim, F. A., Hussain, K., Houssein, E. H., Mabrouk, M. S., & Al-Atabany, W. (2021). Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. *Applied Intelligence, 51*(3), 1531–1551.

He, Y., Xie, H., Wong, T.-L., & Wang, X. (2018). A novel binary artificial bee colony algorithm for the set-union knapsack problem. *Future Generation Computer Systems, 78*, 77–86.

Li, S., Chen, H., Wang, M., Heidari, A. A., & Mirjalili, S. (2020). Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems, 111*, 300–323.

Li, Z., He, Y., Li, Y., & Guo, X. (2021). A hybrid grey wolf optimizer for solving the product knapsack problem. *International Journal of Machine Learning and Cybernetics, 12*(1), 201–222.

Li, Y., He, Y., Liu, X., Guo, X., & Li, Z. (2020). A novel discrete whale optimization algorithm for solving knapsack problems. *Applied Intelligence, 50*, 3350–3366.

Martello, S. (1990). "Knapsack problems: algorithms and computer implementations," Wiley-Interscience series in discrete mathematics and optimization.

McDonnell, S., et al. (2003). *Practical field guide to horse behavior: the equid ethogram*. The Blood-Horse Inc.

MiarNaeimi, F., Azizyan, G., & Rashki, M. (2021). Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems. *Knowledge-Based Systems, 213*, 106711.

Mohammadi-Balani, A., Nayeri, M. D., Azar, A., & Taghizadeh-Yazdi, M. (2021). Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Computers & Industrial Engineering, 152*, 107050.

Moradi, N., Kayvanfar, V., & Rafiee, M. (2021). An efficient population-based simulated annealing algorithm for 0–1 knapsack problem. *Engineering with Computers*, 1–20.

Patvardhan, C., Bansal, S., & Srivastav, A. (2016). Parallel improved quantum inspired evolutionary algorithm to solve large size quadratic knapsack problems. *Swarm and Evolutionary Computation, 26*, 175–190.

Pisinger, D. (1995). A minimal algorithm for the bounded knapsack problem. In *International Conference on Integer Programming and Combinatorial Optimization* (pp. 95–109). Springer.

Połap, D., & Woźniak, M. (2021). Red fox optimization algorithm. *Expert Systems with Applications, 166*, 114107.

Talatahari, S., & Azizi, M. (2021). Chaos game optimization: a novel metaheuristic algorithm. *Artificial Intelligence Review, 54*(2), 917–1004.

Wang, L., Zheng, X.-L., & Wang, S.-Y. (2013). A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. *Knowledge-Based Systems, 48*, 17–23.

Waring, G. (1983). "The behavioral traits and adaptations of domestic and wild horses, including ponies," Horse behavor.

Xiang, W.-L., An, M.-Q., Li, Y.-Z., He, R.-C., & Zhang, J.-F. (2014). A novel discrete global-best harmony search algorithm for solving 0–1 knapsack problems. *Discrete Dynamics in Nature and Society, 2014*.

Zhou, Y., Chen, X., & Zhou, G. (2016). An improved monkey algorithm for a 0–1 knapsack problem. *Applied Soft Computing, 38*, 817–830.

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