Alexandria Engineering Journal (2023) 67, 609–632

HOSTED BY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

Binary light spectrum optimizer for knapsack problems: An improved model

Mohamed Abdel-Basset [a], **Reda Mohamed** [a], **Mohamed Abouhawwash** [b,c], **Ahmad M. Alshamrani** [d], **Ali Wagdy Mohamed** [e,f,*], **Karam Sallam** [g]

[a] *Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt*
[b] *Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt*
[c] *Department of Computational Mathematics, Science, and Engineering (CMSE), College of Engineering, Michigan State University, East Lansing, MI 48824, USA*
[d] *Statistics and Operations Research Department, College of Science, King Saud University, P.O. Box 2455, Riyadh 11451, Saudi Arabia*
[e] *Operations Research Department, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza 12613, Egypt*
[f] *Department of Mathematics and Actuarial Science School of Sciences Engineering, The American University in Cairo, Cairo 11835, Egypt*
[g] *School of IT and Systems, University of Canberra, Canberra, ACT 2601, Australia*

Received 20 October 2022; revised 4 December 2022; accepted 10 December 2022

KEYWORDS

0–1 knapsack problem;
Multidimensional knapsack;
Transfer function;
Light spectrum optimizer;
Swarm and evolutionary behaviors based improvement strategy

Abstract This paper presents a binary variant of a novel physics-based *meta*-heuristic optimization algorithm, namely Light spectrum optimizer (LSO), for tackling both the 0–1 knapsack (KP01) and multidimensional knapsack problems (MKP). Because of the continuous nature of the standard LSO that contradicts the knapsack problem's discrete nature, two various transfer functions: S-shaped and X-shaped, are used to convert the continuous values produced by LSO into discrete ones. Some binary solutions produced by the binary LSO (BLSO) may be infeasible, so an improvement-repair strategy is used to convert those solutions into feasible ones by making some improvements on them. Moreover, the classical LSO was modified in this study to propose a new binary variant, namely BMLSO, with better exploration and exploitation operators for overcoming the knapsack problems. Additionally, a novel method, which simulates the swarm intelligence behaviors and the simulated binary crossover (SBX) to accelerate the convergence speed with avoiding stuck into local minima, has been proposed for producing a new binary variant of MLSO known as BHLSO. To verify the performance of the proposed binary variants of LSO, 45 bench-

* Corresponding author at: Operations Research Department, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza 12613, Egypt. Department of Mathematics and Actuarial Science School of Sciences Engineering, The American University in Cairo, Cairo 11835, Egypt.
E-mail addresses: mohamedbasset@ieee.org (M. Abdel-Basset), redamohamed@zu.edu.eg (R. Mohamed), abouhaww@msu.edu (M. Abouhawwash), ahmadm@ksu.edu.sa (A.M. Alshamrani), aliwagdy@staff.cu.edu.eg, aliwagdy@staff.cu.edu.eg (A. Wagdy Mohamed), karam.sallam@canberra.edu.au (K. Sallam).

Peer review under responsibility of Faculty of Engineering, Alexandria University.

https://doi.org/10.1016/j.aej.2022.12.025

mark instances of KP01 and 30 benchmark instances of MKP used commonly in the literature have been used in our experiments. The experimental findings show the superiority of BHLSO for both KP01 and MKP compared with several well-known algorithms in terms of CPU time, convergence speed, and accuracy.

1. Introduction

No doubt that solving the knapsack problems either the 0–1 knapsack problems (KP01) or the multidimensional knapsack are indispensable for many fields, *i.e,* adaptive multimedia systems [1], principal budgeting [2], real estate property maintenance optimization [3], resource allocation [4], energy minimization [5,6], resource allocation [7], cargo loading [8], and many others [9–12]. Therefore, several algorithms of all kinds, including traditional ones such as dynamic programming [13], branch and bound [14] and implicit enumeration [15]; swarm intelligence algorithms such as equilibrium optimizer [16], complex-valued encoding satin bowerbird optimization algorithm [17], and marine predators algorithm [18]; and evolutionary algorithms such as genetic algorithms [19], and differential evolution [20]. Since solving knapsack problems needs to check several permutations that may include the optimal solution, the traditional methods' performance will deteriorate with increasing the number of the permutations that need to be checked, and this is also impractical in terms of the time complexity [21]. Therefore, the swarm intelligence algorithms and the evolutionary algorithms have widely contributed to overcoming this problem and developing high-quality outcomes due to their ability to solve several optimization problems in a reasonable time [22,23].

Before speaking about the suggested algorithms in the literature for both KP01 and MKP, let's expose each one's mathematical model. At the outset, KP01 involves several n items with a profit p_i and a weight w_i, which needs to be addressed to select the items that maximize the total profit by keeping the sum of their weights less than the knapsack capacity (c). The mathematical formulation of the 0–1 KP is expressed as:

$$maximize f(\vec{x}) = \sum_{i=1}^{n} x_i * p_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i * x_i < c \tag{1}$$

In the same context, Similar to the KP01, The MKP consist of n items, where each one has m weights and a profit p_i. There is a knapsack of m dimensions, where the jth dimension has a capacity c_j. It also has m knapsacks, each of which has a capacity c_j. The objective of the MKP is finding the items that maximize the profit for each dimension of this knapsack with satisfying the capacity of the same dimension. This problem is mathematically defined as:

$$maximize f(\vec{x}) = \sum_{i=1}^{n} x_i * p_i \text{subject to} \sum_{i=1}^{n} w_{ji} * x_i \le c_j, \quad j = 0, 1, 2, \cdots\cdots\cdots, m \tag{2}$$

In the rest of this section, some of the recent papers published for tackling the KP01 and the MKP will be reviewed. At the outset, the improved quantum-inspired wolf pack algorithm (QWPA) [24] has been proposed for tackling the KP01 problems. The QWPA was improved using two main operations: quantum rotation and quantum collapse. The former is used to accelerate the convergence toward the optimal solution, while the latter is responsible for avoiding stuck into local minima. Practically, QWPA was validated on datasets with dimensions up to 1000 and compared with many optimization algorithms to check its efficiency. Furthermore, the Butterfly Optimization algorithms (MBO) improved by the opposition-based learning strategy (OBL), and the Gaussian perturbation has been proposed for tackling the KP01 [25]. OBL was applied to half population in the late phase of the optimization to increase the convergence speed toward the optimal solution. At the same time, Gaussian perturbation acts on individuals with weak fitness at each iteration to avoid falling into local optima. This algorithm was verified on three groups with 15 large-scale instances within each one.

In the same context, Feng [26] improved the efficiency of the MBO by the chaos map to enhance global optimization. Further, the MBO has also been improved by applying the Gaussian perturbation to increase the convergence speed. The flower pollination algorithm (FPA) [27] has been proposed for tackling the KP01. Because the standard FPA has been applied for tackling the continuous optimization problem, this contradicts the nature of the KP01, a binary problem. Therefore, the author used the sigmoid transfer function to convert the continuous values produced by FPA into binary ones. More than that, the author used a penalty function not to select the infeasible solutions as the best-so-far solution. Besides, those infeasible solutions were improved using an improvement-repair strategy.

Lai et al [28] proposed a novel quantum particle swarm optimization algorithm(QPSO) for tackling the 0–1 MKP problem. QPSO integrated the distance-based diversity-preserving strategy to manage the population and a local optimization method based on variable neighborhood variables to improve the solutions. This algorithm was verified on 250 benchmark instances and compared with a number of state-of-the-art algorithms to check its competitiveness. Furthermore, the Harmony search (HS) [29] has also been suggested for solving the single objective and multi-objective 0–1 knapsack problems. García, J., et al. [30] used the clustering DB-scan technique to obtain binary versions of the continuous *meta*-heuristic algorithms for tackling the MKP. The experimental results show that the algorithms integrated with the DB-scan operator can produce better results than the transfer function and random operators.

Beheshti, Z. [31] proposed a new transfer function known as x-shaped to convert the continuous values into discrete

ones. This function has been applied to produce a binary variant of the particle swarm optimization (XBPSO) for the MKP. This transfer function could increase the exploitation and exploration capabilities for creating more binary permutations, which could achieve better outcomes. Experimentally, XBPSO has been validated on 33 benchmark instances compared with a number of the BPSO and the other discrete algorithms to check its effectiveness. Abdel-Basset, M., et al. [32] proposed the modified Multi-Verse optimization (MMVO) for solving the KP01 and MKP using the V-shaped transfer function to convert the continuous values into binary ones. The MMVO has also been improved by re-initializing the population each a number of iterations to promote the exploration capability of the algorithm.

Baykasoğlu [33] improved the firefly algorithm (FA) using partial random restarts and an adaptive move procedure to develop a strong variant able to effectively solve the dynamic multidimensional knapsack (DMK) problems. This variant has been compared to three competitors: the genetic algorithm (GA), differential evolution (EO), and classical FA, to reveal its performance. In [34], three different representations were thoroughly compared in order to assess their performance on the DMK problem. The experimental findings proved that indirect representation is more suitable for this problem. Ozsoydan [35] proposed an effective binary swarm-based optimization technique based on particle swarm optimization (PSO) and GA for solving the set-union knapsack problem, which is widely used in real-life applications, like information security systems. In addition, an optional mutation operator to decrease the population diversity has been developed to improve the performance of this swarm-based technique.

In [36], two strategies, namely the complex-value encoding method and the greedy strategy, were designed to improve the performance of the wind-driven optimization (WDO) algorithm in terms of population diversity and premature convergence when tackling the 0–1 knapsack problem; this improved algorithm was named complex-valued encoding WDO (CWDO). CWDO was verified using three classes of test instances: small-scale, large-scale, and standard test cases, and compared to several other algorithms. In terms of stability and performance, CWDO could outperform all of the compared optimizers. Zhou [37] proposed a novel complex-valued encoding bat algorithm (CPBA) with better population diversity and better convergence speed for solving the 0–1 knapsack problem. The CPBA improves the ability to explore, and it is useful for solving both the small-scale and the large-scale 0–1 knapsack problem. There are several other works on the knapsack problems that have been proposed over the last two decades [38–44]. Unfortunately, the majority of the existing algorithms proposed for the knapsack problems have one or more of the following drawbacks: lack of population diversity, premature convergence, and slow convergence speed. As a result, in this paper, we attempt to propose a new powerful binary technique capable of effectively diversifying the population while also improving convergence speed.

A new physics-based *meta*-heuristic algorithm, namely Light Spectrum Optimizer (LSO), inspired by the meteorological phenomenon, has been proposed to tackle the continuous optimization algorithms [45]. The significant success achieved by the LSO for solving several challengeable CEC benchmarks (CEC2014, CEC2017, CEC2020, and CEC2022) motivates us to offer its binary variant using the sigmoid transfer function for tackling KP01 and the X-shaped transfer function for MKP [45]. Besides, the binary LSO (BLSO) was modified to propose a new variant, namely the binary modified LSO (BMLSO), to strengthen the exploration and exploitation operator when tackling the knapsack problems, especially the large-scale problems. As another attempt to promote the exploration and exploitation capability of the modified LSO, a new novel method based on integrating the swarm behaviors with the simulated binary crossover (SBX) to help the algorithm check more permutations that may involve better outcomes was proposed. This method, abbreviated as SEI, is effectively integrated with the BLSO to produce a new variant known as BHLSO. These three binary variants used an improvement-repair strategy to convert the infeasible solutions, which don't come true due to the constraints of both KP01 and MKP, into feasible ones by making some improvements on those solutions. To verify the performance of both BLSO and BHLSO, 45 benchmark instances for KP01 and 30 instances for MKP commonly used in the literature have been used. The experimental outcomes elaborate on the superiority of BHLSO and BLSO compared to a number of recent robust *meta*-heuristic algorithms in terms of convergence speed and the quality of the outcomes. Finally, the contributions of this paper are summarized as follows:

- Propose a binary variant of LSO for tackling KP01 and MPK using the S- and X-shaped transfer functions, respectively.
- Propose a modified variant of LSO, namely MLSO, with stronger exploration and exploitation operators to effectively tackle the KP01 and MPK problems.
- Propose a new novel method based on integrating the swarm intelligence's behavior with the simulated binary crossover to further promote the exploration and exploitation operators of MLSO. This method is combined with MLSO to produce a binary variant known as BHLSO.
- These proposed variants were validated on well-known instances of both KP01 and MKP and compared with many well-known robust algorithms to check their competitiveness. The experimental outcomes show the superiority of both BMLSO and BHLSO in terms of quality and convergence speed.

The rest of this research is arranged as that: Section 2 overview the light Spectrum Optimizer, Section 3 explains the proposed algorithm, Section 4 presents discussion and comparison, and Section 5 shows a conclusion about our work and future work.

2. Light spectrum optimizer

Recently, a novel *meta*-heuristic algorithm known as Light Spectrum Optimizer (LSO) has been proposed for tackling the single-objective optimization test functions. LSO has been inspired by the meteorological phenomenon, which says that the light rays reflection, refraction, and dispersions with different angles because of going through the raindrops cause the colorful rainbow spectrum rays with a reflective index ranging between 1.331 as $\boldsymbol{k}^{red}$ and 1.344 as $\boldsymbol{k}^{\text{violet}}$. In the optimization process, each ray out of the generated rays represents a candidate solution to the optimization problem. At the beginning of

the process, a group of N rays are generated with d dimensions for each one, which are randomly initialized within their upper and lower bound of each dimension according to the following equation:

$$\overrightarrow{\boldsymbol{x_i}} = \overrightarrow{\boldsymbol{L}} + \left(\overrightarrow{\boldsymbol{U}} - \overrightarrow{\boldsymbol{L}}\right).\overrightarrow{\boldsymbol{r}}, \boldsymbol{i} = 1, 2, 3, \cdots\cdots., \boldsymbol{N} \tag{2}$$

where $\overrightarrow{x_i}$ is a vector used to contain the values of the i^{th} ray. $\overrightarrow{L}$ and $\overrightarrow{U}$ are two vectors, including the lower and upper search space for the optimization problem's dimensions, respectively. In brief, like the metaheuristic algorithms, the optimization process of LSO is divided into two stages: Generating new colorful ray as exploration, and Colorful rays scattering as exploitation, which are extensively described in the next sections.

2.1. Generating new colorful ray: Exploration mechanism

The optimization process will start with updating the reflective index between $\boldsymbol{k^{red}}$ and $\boldsymbol{k^{violet}}$ as that:

$$\boldsymbol{k^r} = \boldsymbol{k^{red}} + \boldsymbol{r}_1 \times (\boldsymbol{k^{violet}} - \boldsymbol{k^{red}}) \tag{3}$$

where $\boldsymbol{r}_1$ is a random number ranging between 0 and 1. To control the reflection and refraction of light rays, a probability between 0 and 1 abbreviated as p is used and assigned a value of 0.8. But, to manage the scattering of the colorful rainbow curve, a probability generated randomly between 0 and 1 known as q is used.

At the optimization process, the directions of the rainbow spectrums are determined according to the normal vector of inner refraction $\overrightarrow{\boldsymbol{x_{nA}}}$, inner reflection $\overrightarrow{\boldsymbol{x_{nB}}}$, and outer refraction $\overrightarrow{\boldsymbol{x_{nC}}}$ generated successively using the following equations:

$$\overrightarrow{x_{nA}} = \frac{\overrightarrow{x_r(t)}}{norm\left(\overrightarrow{x_r(t)}\right)} \tag{4}$$

$$\overrightarrow{x_{nB}} = \frac{\overrightarrow{x_p(t)}}{norm\left(\overrightarrow{x_p(t)}\right)} \tag{5}$$

$$\overrightarrow{x_{nC}} = \frac{\overrightarrow{x^*}}{norm\left(\overrightarrow{x^*}\right)} \tag{6}$$

where $\overrightarrow{x_r(t)}$ is a solution selected arbitrarily from the population in the current iteration t, $\overrightarrow{x_p(t)}$ is the current solution of the current ray, $\overrightarrow{x^*}$ is the best-so-far solution, $norm(.)$ is the normalized vector. Regarding the incident light ray $\overrightarrow{x_{L0}}$, eq. (8) is used to compute it.

$$X_{mean} = \frac{\sum_i^N \overrightarrow{x_i}}{N} \tag{7}$$

$$\overrightarrow{x_{L0}} = \frac{X_{mean}}{norm(X_{mean})} \tag{8}$$

where X_{mean} indicate the mean of the population at the current iteration. And, the vectors of inside and outside refracted and reflected light rays are computed as that:

$$\overrightarrow{x_{L1}} = \frac{1}{k^r}\left[\overrightarrow{x_{L0}} - \overrightarrow{x_{nA}}\left(\overrightarrow{x_{nA}} \cdot \overrightarrow{x_{L0}}\right)\right] - \overrightarrow{x_{nA}}\left|1 - \frac{1}{(k^r)^2} + \frac{1}{(k^r)^2}\left(\overrightarrow{x_{nA}} \cdot \overrightarrow{x_{L0}}\right)^2\right|^{\frac{1}{2}} \tag{9}$$

$$\overrightarrow{x_{L2}} = \overrightarrow{x_{L1}} - 2\overrightarrow{x_{nB}}\left(\overrightarrow{x_{L1}} \cdot \overrightarrow{x_{nB}}\right) \tag{10}$$

$$\overrightarrow{x_{L3}} = k^r\left[\overrightarrow{x_{L2}} - \overrightarrow{x_{nC}}\left(\overrightarrow{x_{nC}} \cdot \overrightarrow{x_{L2}}\right)\right] + \overrightarrow{x_{nC}}\left|1 - (k^r)^2 + (k^r)^2\left(\overrightarrow{x_{nC}} \cdot \overrightarrow{x_{L2}}\right)^2\right|^{\frac{1}{2}} \tag{11}$$

where $\overrightarrow{x_{L1}}$, $\overrightarrow{x_{L2}}$, and $\overrightarrow{x_{L3}}$ are three vectors representing the inner refracted, inner reflected, and outer refracted light rays, successively.

After the calculation of the rays' directions, the candidate solutions are calculated based on the value of a probability generated randomly between 0 and 1; this value is assigned in a variable symbolized p. In particular, if the value of p is smaller than a randomly generated value at the interval [0, 1], then the updated solution at the iteration $t + 1$ is calculated according to the following formula:

$$\overrightarrow{x_{t+1}} = \overrightarrow{x_t} + \epsilon RV_1^n GI\left(\overrightarrow{x_{L1}} - \overrightarrow{x_{L3}}\right) \times \left(\overrightarrow{x_{r1}} - \overrightarrow{x_{r2}}\right) \tag{12}$$

Otherwise, the updated solution will be calculated according to the following formula:

$$\overrightarrow{x_{t+1}} = \overrightarrow{x_t} + \epsilon RV_2^n GI\left(\overrightarrow{x_{L2}} - \overrightarrow{x_{L3}}\right) \times \left(\overrightarrow{x_{r3}} - \overrightarrow{x_{r4}}\right) \tag{13}$$

where $\overrightarrow{x_{t+1}}$ indicates the newly-generated solution, $\overrightarrow{x_t}$ stands for the current solution at iteration t. $r1$, $r2$, $r3$, and $r4$ are indices of four solutions selected randomly from the current population. RV_1^n and RV_2^n are uniformly-generated vectors. ϵ is a scaling factor calculated according to (14). GI is an adaptive control factor based on the inverse incomplete gamma function and calculated by (15):

$$\epsilon = a \times RV_3^n \tag{14}$$

where RV_3^n is a vector of randomly generated numbers that are normally distributed and have a mean and standard deviation of zero and one, respectively. An adaptive parameter named a is calculable using (16).

$$GI = a \times r^{-1} \times P^{-1}(a, 1) \tag{15}$$

GI stands for an adaptive control factor. r is a random number generated uniformly between $[0, 1]$ that is inversed to promote the exploration operator. P^{-1} is the inverse incomplete gamma function with respect to a.

$$\mathrm{a} = \mathrm{RV}_2\left(1 - \left(\frac{t}{Tmax}\right)\right) \tag{16}$$

where t indicates the current iteration number, RV_2 is a numerical value generated uniformly between $[0, 1]$ and $Tmax$ is the maximum iteration.

2.2. Colorful rays scattering: Exploitation mechanism

This stage assists in dispersing the light in the direction of the current solution, the best-so-far solution, and a solution chosen at random from the current population to enhance the exploitation operator. The following is the mathematical representation of scattering near the current solution:

$$\overrightarrow{x_{t+1}} = \overrightarrow{x_t} + RV_3 \times \left(\overrightarrow{x_{r1}} - \overrightarrow{x_{r2}}\right) + RV_4^n \times (R < \beta) \times \left(\overrightarrow{x^*} - \overrightarrow{x_t}\right) \tag{17}$$

where $\overrightarrow{x^*}$ indicates the best-so-far solution, $\overrightarrow{x_{r1}}$ and $\overrightarrow{x_{r1}}$ are two randomly-selected solutions from the current population. RV_3 includes a number generated randomly between 0 and 1. RV_4^n is a vector initialized randomly between 0 and 1. The best-so-far solution and the current solution are used to generate rays in a new location for the second scattering phase using the following formula:

$$\overrightarrow{x_{t+1}} = 2\cos(\pi \times r_1)\left(\overrightarrow{x^*}\right)\left(\overrightarrow{x_t}\right) \tag{18}$$

where r_1 is a number between 0 and 1 that was generated at random. π indicates the proportion of a circle's perimeter to its diameter. According to the following formula, switching between the first and second scattering phases is accomplished based on a predetermined probability P_e:

$$\overrightarrow{x_{t+1}} = \begin{cases} Eq.(17) if R < P_e \\ Eq.(18) Otherwise \end{cases} \tag{19}$$

where R is a number chosen at random from 0 to 1. The final scattering phase is based on creating a new solution using the current one and a solution chosen at random from the population using the following formula:

$$\overrightarrow{x_{t+1}} = \left(\overrightarrow{x_{r1}^p} + |RV_5| \times \left(\overrightarrow{x_{r2}} - \overrightarrow{x_{r3}}\right)\right) \times \overrightarrow{U} + \left(1 - \overrightarrow{U}\right) \times \overrightarrow{x_t} \tag{20}$$

where RV_5 is a random number based on the normal distribution, and $\overrightarrow{U}$ is a vector generated randomly values between 0 and 1. Exchanging (19) and (20) is based on computing the difference in fitness values between each solution and the best-so-far solution and normalizing this difference between 0 and 1 according to (21). If this difference is smaller than a threshold value R_1 which is assigned a random value between 0 and 1, (19) will be applied; otherwise, (20) is executed as modeled in (22):

$$F' = \left|\frac{F - F_b}{F_b - F_w}\right| \tag{21}$$

$$\overrightarrow{x_{t+1}} = \begin{cases} Eq.(19) if R < P_s | F' < R_1 \\ \quad Eq.(20) Otherwise \end{cases} \tag{22}$$

where R and R_1 are randomly generated numbers between 0 and 1. The flowchart of LSO is shown in Fig. 1.

3. The proposed algorithm: BHLSO, and BMLSO

In this section, the binary variant of the LSO, in addition to the improved one will be discussed in this section in detail. The main steps of this section are organizationally surveyed as follows: (1) the initialization step shows the way of distributing the individuals within the search space of the problem before running the optimization process, (2) afterward, the fitness function, and improving and repairing methodology are secondly shown, (3) the binary variant of the classical LSO is clearly presented, (4) the binary modified LSO (BMLSO) algorithm are clearly explained, and (5) finally, the improvement strategy and the proposed algorithm are clearly exposed to the readers.

3.1. Initialization, evaluation, and repairing strategy

At the start, a group of N rays with d dimensions for each one will be created and accurately distributed within the search space of the problem according to the following formula:

$$x_{i,j} = \begin{cases} 1 \ if r > 0.5 \\ 0 \ otherwise \end{cases} \tag{23}$$

Afterward, each solution will be evaluated to see its quality for tackling KP01 and MKP problems using the following formula:

$$\mathrm{f}(x) = \sum_{z=1}^{d} w_z * x_z \leq c \tag{24}$$

where w_z indicates the weight of the z^{th} item, and x_z shows if the same item is selected (including a value of 1), or not selected (including a value of 0). c includes the knapsack capacity. After evaluating each solution, the infeasible solution that maximizes the profits but doesn't subject to the knapsack capacity constraints couldn't be used as the best-so-far solution x^*. Therefore, it is discarded by using a penalty function (PF) that gives it a negative fitness value not to be selected as the best-so-far solution. In our work, those infeasible solutions will be repaired using a repair algorithm shown in algorithm 1 to be competitors (feasible ones) with the other solutions within the population. Afterward, those repaired solutions will be improved to search for better outcomes as shown in algorithm 1.

Algorithm 1 Repairing and improving Algorithm (RA)

Input: infeasible solution x_i
1. // Repairing algorithm
2. **While**($f(x_i)$ > c)
3. Remove the item with the lowest $\frac{\mathrm{pr}_z}{\mathrm{w}_z}$ from this solution
4. **End while**
5. // Improving algorithm
6. **While**($f(x_i) \leq$ c)
7. insert the items with the highest $\frac{\mathrm{pr}_z}{\mathrm{w}_z}$ in the knapsack
8. **End while**

Return x_i

3.2. Transfer functions

The LSO produces continuous values during the optimization process, which need to be converted into binary values to tackle knapsack problems. To do that, the sigmoid and x-shaped transfer function is used. In our experiments, the sigmoid transfer was described mathematically in eq. (26) and depicted in Fig. 2 could come true better outcomes with the continuous optimization algorithms when tackling the KP10 as shown in our papers [16,18], it is used in this research to see the performance of the LSO under this function. This function is defined as follows:

$$F(\overrightarrow{a}) = \frac{1}{1 + e^{-2*\overrightarrow{a}}} \tag{25}$$

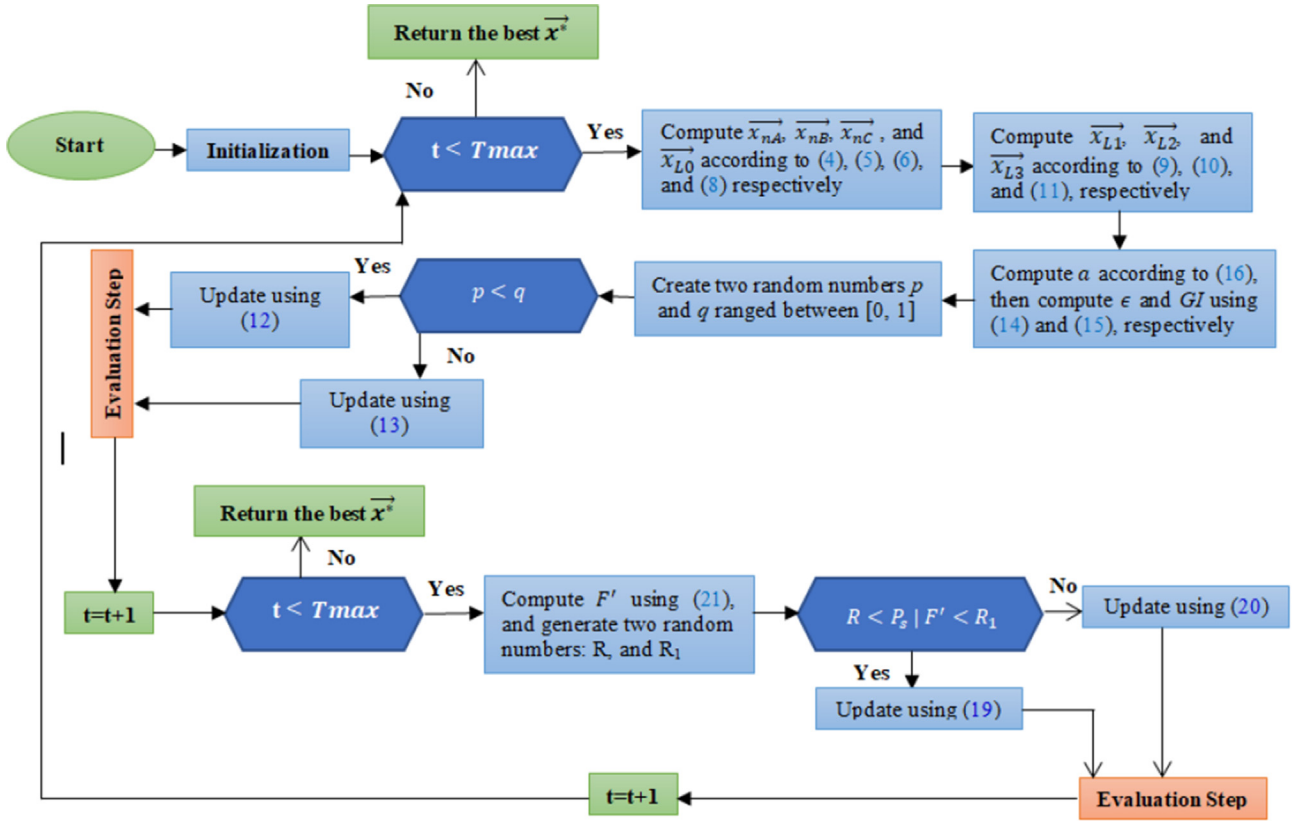


Fig. 1 Flowchart of LSO.

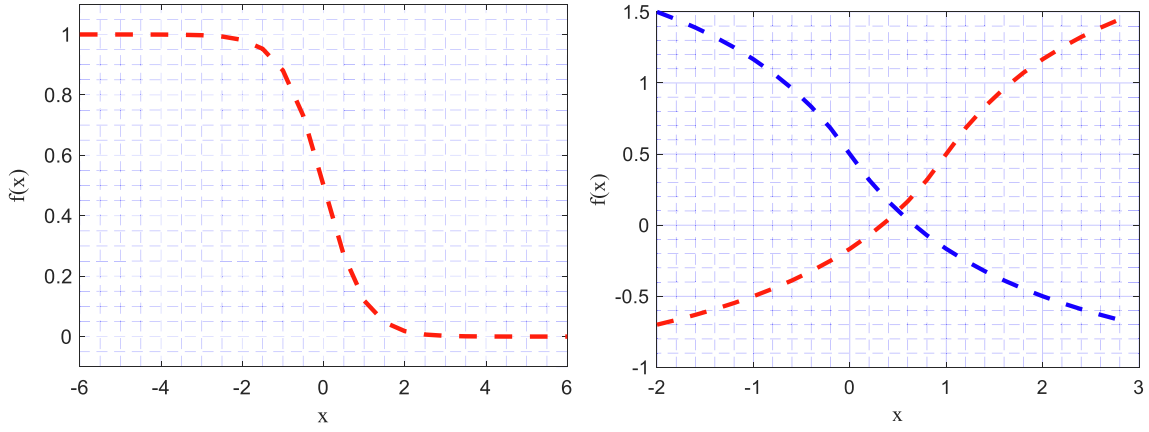
$$F_{bin} = \begin{cases} 1 if F(\vec{a}) \geq rand \\ 0 otherwise \end{cases} \tag{26}$$

On the other side, the x-shaped transfer function proved its efficiency compared with the others when solving the 0–1 KP [31]. As a result, this research is checked with our proposed algorithm and some of the other optimization algorithms to see its effectiveness. This function produces two binary solutions of the continuous solution to promote the exploration and exploitation capabilities for reaching better outcomes. Afterward, those two solutions are compared with each other, and the one with the highest fitness values is returned to be inserted in the next generation. This transfer function is mathematically defined as follows:

$$F_1(\vec{a}) = \frac{-\vec{a}}{1 + |-\vec{a}| * 0.5} + 0.5 \tag{27}$$

$$F_{bin1} = \begin{cases} 1 if F_1(\vec{a}) \geq rand \\ 0 otherwise \end{cases} \tag{28}$$

$$F_2(\vec{a}) = \frac{\vec{a} - 1}{1 + |\vec{a} - 1| * 0.5} + 0.5 \tag{29}$$



a) S-shaped Transfer function *b)* X-shaped Transfer function

Fig. 2 Depiction of X-shaped and S-shaped curves.

$$F_{bin2} = \begin{cases} 1 if F_2(\overrightarrow{a}) \geq rand \\ 0 otherwise \end{cases} \quad (30)$$

$$F_{bin} = \begin{cases} F_{bin1} iff(F_{bin1}) > f(F_{bin2}) \\ F_{bin2} otherwise \end{cases} \quad (31)$$

3.3. Binary variant of classical LSO

The classical LSO has been recently proposed for tackling continuous optimization problems; hence, in its current form, it cannot be applied to discrete optimization problems, like knapsack problems. Therefore, in this section, the classical LSO will be adapted using the transfer functions to be applicable to the knapsack problem. In the beginning, the initialization step is discretely performed by randomly assigning a 0 or 1 value for N solutions, which are then evaluated to identify the quality of each one using eq. (24) as the objective function. Afterwards, the optimization process of LSO will be fired to generate new continuous solutions, which are transformed into discrete ones using the transfer function discussed before. Unfortunately, some discrete solutions might not satisfy the constraints of the knapsack problem; therefore, those solutions will be repaired using algorithm 1 to become feasible to tackle this problem. This procedure will be repeated until the maximum number of iterations is reached. Finally, the steps of the binary LSO are stated in Algorithm 2.

Algorithm 2: Binary light spectrum optimizer (BLSO)

Input: a population of ***N*** *rays, the maximum Iteration* ***T***
Initialization
While *(t < T)*
  for *each i ray*
    Convert x_i to binary-one using the adequate transfer function
    Evaluate the fitness value using eq. (24) and applying Algorithm 1
    update the best x^*
    determine normal lines $\overrightarrow{x_{nA}}$, $\overrightarrow{x_{nB}}$, &$\overrightarrow{x_{nC}}$
    determine direction vectors $\overrightarrow{x_{L0}}$, $\overrightarrow{x_{L1}}$,$\overrightarrow{x_{L2}}$, &$\overrightarrow{x_{L3}}$
    update the refractive index k^r
    update a, ϵ, &GI
    Generate two random numbers: p, q between 0 and 1
%%%%Generating new ColorFul ray: Exploration phase
    if $p \leq q$
    update using Eq. (12)
    Else
    update using Eq. (13)
    end if
    Convert x_i to binary-one using the adequate transfer function
    evaluate the fitness value using eq. (24) and applying Algorithm 1
    update the best x^*
  %%%%Scattering phase: exploitation phase
    Update using Eq. (22)
  end for
end while
Return *the best-so-far solution:* $\boldsymbol{x}^*$

3.4. Binary modified light spectrum optimizer (BMLSO)

Unfortunately, the classical BLSO has some disadvantages, including low population diversity, sticking to local minima, and slow convergence speed, which prevent it from fulfilling strong outcomes for discrete optimization problems like knapsack problems. Therefore, the exploration and exploitation operators of the classical BLSO are improved in a new variant, namely the modified LSO (MLSO), to improve its ability to tackle the knapsack problems. The modified operators are described in detail in the following sections.

3.4.1. Modified exploration operator

In this stage, the equations of the classical LSO used to generate the new colorful ray after determining the direction of the rays are updated to improve its ability to explore several discrete solutions, which might involve the near-optimal solution to the knapsack problem. This modified exploration operator generates the new candidate solution according to the following two equations, which are exchanged with each other based on the probability p. If $p < 0.8$, the current solution is updated using eq. (32); otherwise, it is updated by eq. (33)

$$\overrightarrow{x(t+1)} = \overrightarrow{r_3}.\overrightarrow{x_t} + \epsilon.\overrightarrow{r}_4.\left(\overrightarrow{x_{L1}} - \overrightarrow{x_{L3}}\right) \quad (32)$$

$$\overrightarrow{x(t+1)} = \overrightarrow{r_5}.\overrightarrow{x(t)} + \epsilon.\overrightarrow{r_6}.\left(\overrightarrow{x_{L2}} - \overrightarrow{x_{L3}}\right) \quad (33)$$

where $\overrightarrow{x(t)}$ is the current candidate solution, r_3 and r_5 are vectors of random numbers generated uniformly between $[0, 1]$, $\overrightarrow{r_4}$ and $\overrightarrow{r_6}$ are generated randomly using the normal distribution. But, within our research, generating $\overrightarrow{r_4}$ and $\overrightarrow{r_6}$ uniformly between 0 and 1 could come true better outcomes. ϵ refers to a scaling factor calculated as:

$$\epsilon = a \times \overrightarrow{r_6} \quad (34)$$

$\overrightarrow{r_6}$ are generated randomly using the normal distribution. Also, generating this vector within our research uniformly between 0 and 1 could come true better outcomes based on our experiments. a is a distance control factor computed using the following equation:

$$\mathrm{a} = 1 - \left(\frac{t}{\mathrm{T}}\right) \quad (35)$$

T is the maximum iteration.

3.4.2. Modified exploitation operator

Regarding the exploitation capability of the modified LSO, it is managed based on the q and z factors that determine if the current candidate solution will be updated between the best-so-far and its local best one or around the local best one only. Specifically, If q is smaller than GI, the new solution is updated between the local best solution of the current individual and the global best solution as follows:

$$\overrightarrow{x_{t+1}} = \sqrt{\left(\overrightarrow{x^*}\right)^2 + \left(\overrightarrow{x_t^p}\right)^2 + \left(2\cos\left(\theta^r\right)\left(\overrightarrow{x^*}\right)\left(\overrightarrow{x_t^p}\right)\right)} \quad (36)$$

where θ^r is a random angle between $[0, 2\pi]$, $\overrightarrow{x_t^p}$ indicates the last best solution obtained by the current solution. However, if the

probability z is greater than GI, then the newly updated solution will be updated as:

$$\overrightarrow{x_{t+1}} = \overrightarrow{x_t^p} - RV_4^n \times GI\left(\frac{ub - lb}{2}\right) \tag{37}$$

where RV_4^n is a vector generating using the normal distribution with a mean of 0 a standard deviation of one. *lb* and *ub* are respectively the lower and upper bounds of the dimensions in the optimization problem. The binary variant of this modified LSO is listed in Algorithm 3 which describes the steps of adapted MLSO for tackling the knapsack problem, namely BMLSO.

Algorithm 3: the BMLSO algorithm

Input: a population of **N** *rays, the maximum Iteration* **T**

Initialization
While (t < T)
  for each i ray
  Convert x_i to binary-one using the adequate transfer function
  evaluate the fitness value using eq. (24) and applying Algorithm 1
  update the best x^*
  Compute normal lines $\overrightarrow{x_{nA}}$, $\overrightarrow{x_{nB}}$, &$\overrightarrow{x_{nC}}$
  Specify direction vectors $\overrightarrow{x_{L0}}$, $\overrightarrow{x_{L1}}$,$\overrightarrow{x_{L2}}$, &$\overrightarrow{x_{L3}}$
  Generate the refractive index k^r
  update a, andϵ
  Update p, q, and z randomly between 0 and 1
  if $p \leq 0.8$
    update the current using (32)
  Else
    update the current using (33)
  end if
  if $q < GI$
    update the current using eq. (36)
  end if
  if $z > GI$
    update the current using eq. (37)
  end if
  end for
$t++$
end while
Return x^*

3.5. Simulated binary crossover.

The Crossover operators are considered a critical component for the evolutionary algorithms used to generate offspring solutions from two or more parents [46]. Those operators are classified into two types, *i.e.,* the mean-centric operators and the parent-centric operators. The simulated binary crossover (SBX) belongs to the parent-centric operators, which generate the offspring solutions around their parents, contradicted the mean-centric operators, which produce the offspring around the centroid of the parents [47]. The SBX is mathematically described as follows to generate the offspring solutions:

$$\begin{aligned} \boldsymbol{x}^1 &= 0.5[\boldsymbol{x}_1(\boldsymbol{t}) + \boldsymbol{x}_2(\boldsymbol{t}) + \boldsymbol{\beta}(\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t}))] \\ \boldsymbol{x}^2 &= 0.5[\boldsymbol{x}_1(\boldsymbol{t}) + \boldsymbol{x}_2(\boldsymbol{t}) - \boldsymbol{\beta}(\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t}))] \end{aligned} \tag{38}$$

where $\boldsymbol{x}^1$ indicates the first generated offspring and $\boldsymbol{x}^2$ is the second one. $\boldsymbol{x}_1(\boldsymbol{t})$***and***$\boldsymbol{x}_2(\boldsymbol{t})$ are two parents at iteration t. $\boldsymbol{\beta}$ is a spreading factor to determine the absolute difference in the generated offspring to that of the parents. To control the distribution of $\boldsymbol{\beta}$, a parameter denoting as $\boldsymbol{\eta_c}$ is used, making the probability of generating the offsprings near the parents is so high if it has a large value.

3.6. Swarm and evolutionary-based improvement strategy (SEI)

In our method, the behaviors of the swarm algorithms and the SBX will be integrated in a novel way to produce a method that has some merits of the swarm confined to the exploration capability and the convergence speed of the SBX. Regarding the swarm behaviors, similar to the whale optimization algorithm, a novel way is suggested to explore the search space to find promising regions, which include the optimal solutions. This way is divided into three steps:

- The first step is used to explore the search space as an attempt to finding better outcomes and mathematically defined as follows:

$$\overrightarrow{x}(t+1) = \overrightarrow{x_r}(t) - \overrightarrow{A}.\overrightarrow{D} \tag{39}$$

$$\overrightarrow{\boldsymbol{D}} = \left|\overrightarrow{\boldsymbol{C}}.\overrightarrow{\boldsymbol{x}}_{\boldsymbol{r}} - \overrightarrow{\boldsymbol{x}}(\boldsymbol{t})\right| \tag{40}$$

$$\overrightarrow{A} = 2.\overrightarrow{a}.\overrightarrow{rand} - \overrightarrow{a} \tag{41}$$

$$\overrightarrow{a} = cc - cc * \frac{t}{T} \tag{42}$$

  where $\overrightarrow{\boldsymbol{x}}_{\boldsymbol{r}}$ is a solution selected from the population at iteration t. $\overrightarrow{\boldsymbol{C}}$ is a uniform random vector including values generated between 0 and 2**.** $\overrightarrow{A}$ is a distance control factor to determine the absolute difference of the current solution to that of the randomly selected one. cc is a fixed value.

- The second step is suggested to promote the exploitation capability of an optimization algorithm, which is mathematically described as follows:

$$\overrightarrow{x}(t+1) = \overrightarrow{x^*}(t) - \overrightarrow{A}.\overrightarrow{D} \tag{43}$$

$$\overrightarrow{D} = \left|\overrightarrow{C}.\overrightarrow{x}(t) - \overrightarrow{x}(t)\right| \tag{44}$$

  The first step and the second step are exchanged with each other to update the current dimension in the current solution based on the distance control factor $\overrightarrow{A}$. If the absolute current value in the $\overrightarrow{A}$ vector is greater than a predefined constant value b, then the first step will be applied to explore this dimension's search space. Otherwise, the second step will be applied to search around the best-so-far solution for a better solution.

- The third step is based on exploiting the solution around the search space of the problem as an attempt to find better outcomes. The difference between this step and the previous one that the previous step was calculating the distance between the current solution and the same one multiplied

by a random value created between 0 and 2. This will make the value of C controlling in the exploitation capability. To be more specific, when C is greater than 1.5, or smaller than 0.5 will make the updated solution near the best-so-far solution and this will promote the exploitation capability. In the third step, the current solution will be updated under the absolute distance between the best-so-far and the current one. Mathematically, this step is defined as follows:

$$\vec{x}(t+1) = \vec{x^*}(t) + \cos(2\pi l).e^l.\vec{D} \tag{45}$$

$$\vec{D} = \left|\vec{x^*}(t) - \vec{x}(t)\right| \tag{46}$$

where l is a randomly generated value between [-1, 1]. As a trying to accelerate the optimization process's convergence speed, the third step is randomly exchanged under a probability of 0.5 with the SBX used two parents to generate the offsprings: the best-so-far solution and the current one, which may accelerate the convergence speed toward the optimal solution. The first step and second step, and the third step and the SBX is exchanged with each other to update the current solution with a predefined probability P. Ultimately, the SEI strategy is described in algorithm 4.

Algorithm 4: The proposed SEI strategy

```
for each i solution
    r: choosing a random number between 0 and 1.
    if (r < P)
        for each j dimension in the i solution
        if (|A| > b)
            Update x_{i,j}(t+1) using eq. (43)
        else
            Update x_{i,j}(t+1) using eq. (39)
        end if
        end for
    else
        r_1: choosing a random number between 0 and 1.
        if (r_1 < 0.5)
            Update the current solution using the SBX
        Else
            Update the current solution using eq. (45)
        end if
        end if
        Convert x_i to binary-one using the adequate transfer function.
        evaluate the fitness value using eq. (24) and applying Algorithm 1
    update the best x*
end for
t++
```

3.7. Binary hybrid MLSO (BHLSO)

To further improve the performance of BMLSO when solving knapsack problems, especially multidimensional knapsack problems, an additional powerful binary variant based on improving MLSO using the novel SEI strategy is presented here; this variant is named BHLSO. The main advantage of this variant is that it mixes genetic operators' characteristics with swarm characteristics to explore and exploit several regions within the search space for better solutions. Finally, the steps of this variant are clearly listed in Algorithm 5.

Algorithm 5: the BHLSO algorithm

```
Input: a population of N rays, the maximum Iteration T
Initialization
While (t < T)
  for each i ray
    Convert x_i to binary-one using the adequate transfer function
    evaluate the fitness value using eq. (24) and applying Algorithm 1
    update the best x* and local best x^l dispersion
    Compute normal lines x_nA, x_nB, &x_nC
    Specify direction vectors x_L0, x_L1, x_L2, &x_L3
    Generate the refractive index k^r
    update a, and ϵ
    Update p, q, and z randomly between 0 and 1
    if p ≤ 0.8
      update the current using eq. (32)
    Else
      update the current using eq. (33)
    end if
    if q < GI
      update the current using eq. (36)
    end if
    if z > GI
      update the current using eq. (37)
    end if
    end for
    Update the current population using Algorithm 2
  t++
end while
Return x*
```

3.8. Advantage and disadvantage

This section is added to report the advantages and disadvantages of the three binary variants of LSO presented in this study. In summary, Table 1 demonstrates the advantages and disadvantages of each proposed binary variant.

4. Experimental results

In this section, the proposed algorithm: BHLSO is validated on KP01 and multidimensional knapsack problems to see its effectiveness in tackling those two different optimization problems. Generally, this section is organized as follows:

Section 4.1: shows the performance metrics.

Section 4.2: exposes the first experiments conducted to compare BLSO, BMLSO, and BHLSO.

Section 4.2: exposes the second experiments conducted on KP01.

Section 4.3: discusses the third experiment conducted on MKP.

Table 1 Advantages and disadvantages of three proposed binary variant: BLSO, BMLSO, and BHLSO.

	BLSO	BMLSO	BHLSO
Advantage	• Easy to implementStrong performance for low-dimensionality • Low computational cost	• Easy to implement • Strong performance for low dimensionality • Low computational cost • Strong performance for high dimensionality • Self-adaptive	• Easy to implement • Strong performance for low and high dimensionality • Strong performance for low dimensionality
Disadvantages	• Stuck into local minima • Lack of population diversity • Slow convergence speed • Low performance for high-dimensionality • Parameter adjustment problem	• Low performance for MKP	• Relative high computational cost • Parameter adjustment problem

4.1. Performance Metrics, parameter settings, and datasets descriptions.

The proposed algorithms and the others are statistically compared with each other based on five statistical metrics: best, average, worst, standard deviation (SD), success rate (SR), success iteration (SI), and CPU Time. Success rate (SR) indicates the percentage of achieving the optimal solution by each algorithm based on the following formula:

$$\mathrm{SR}(\%) = \frac{optimalvaluenumberinallruns}{numberofruns} * 100 \quad (47)$$

Regarding SI, it is used to see the average of the success iterations, which each algorithm could come true the optimal solution within.

With regards to the instances used for both KP01, 45 instances with various item sizes ranging between 4 and 2000 are used to check the performance of the proposed algorithm. These instances are specifically used due to its wide utilization in the literature [48–51]. In Table 2, those instances are described based on four characteristics: the instance name (IN), the number of items (D), the desired optimal solution (Opt), and the knapsack capacity (Capacity). In the same context, for MKP, the proposed algorithm and the other ones are compared with each other on the WEISH benchmark Due to the common use of this benchmark in recent researches [20,32,52], which includes 30 instances with items sizes within the range of 30 and 90 and 5 knapsacks for each instance. This benchmark is descriptively presented in Table 3 in terms of the number of items, the number of the used knapsack (KN), and the optimal solution.

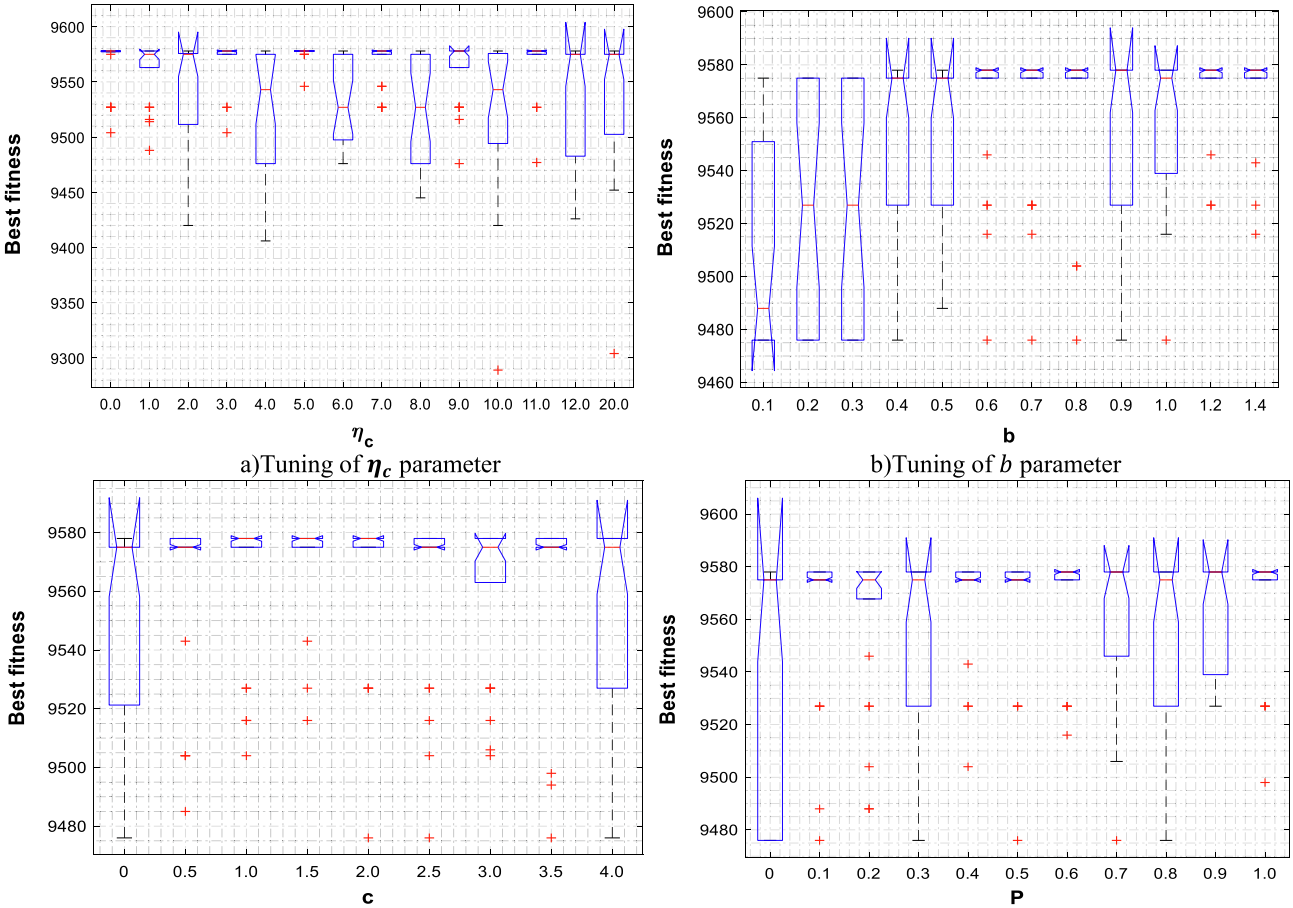
Table 2 Description of the KP01 instances.

IN	Capacity	D	Opt	IN	Capacity	D	Opt	IN	Capacity	D	Opt
KP1	269	10	295	KP8a	1,863,633	8	3,924,400	KP3_100	997	100	2397
KP2	878	20	1024	KP8b	1,822,718	8	3,813,669	KP3_200	997	200	2697
KP3	20	4	35	KP8c	1,609,419	8	3,347,452	KP3_500	2517	500	7117
KP4	11	4	23	KP8d	2,112,292	8	4,187,707	KP3_1000	4990	1000	14,390
KP5	375	15	481.069368	KP8e	2,493,250	8	4,955,555	KP3_2000	9819	2000	28,919
KP7	50	7	107	KP12b	3,259,036	12	6,473,019				
KP8	10,000	23	9767	KP12c	2,489,815	12	5,170,626				
KP9	80	5	130	KP12d	3,453,702	12	6,941,564				
KP6	60	10	52	KP12a	2,805,213	12	5,688,887				
KP10	879	20	1025	KP12e	2,520,392	12	5,337,472				
KP11	577	30	1437	KP1_100	995	100	9147				
KP12	655	35	1689.0	KP1_200	1008	200	11,238				
KP13	819	40	1821	KP1_500	2543	500	28,857				
KP14	907	45	2033	KP1_1000	5002	1000	54,503				
KP15	882	50	2440	KP1_2000	10,011	2000	110,625				
KP16	1050	55	2651	KP2_100	995	100	1514				
KP17	1006	60	2917	KP2_200	1008	200	1634				
KP18	1319	65	2817	KP2_500	2543	500	4566				
KP19	1426	70	3223	KP2_1000	5002	1000	9052				
KP20	1433	75	3614	KP2_2000	10,011	2000	18,051				

Table 3 Descriptions of MKP instances.

Instance(inst)	Opt	D	kN	Instance(inst)	Opt	D	KN
weish01	4554	30	5	weish16	7289	60	5
weish02	4536	30	5	weish17	8633	60	5
weish03	4115	30	5	weish18	9580	70	5
weish04	4561	30	5	weish19	7698	70	5
weish05	4514	30	5	weish20	9450	70	5
weish06	5557	40	5	weish21	9074	70	5
weish07	5567	40	5	weish22	8947	70	5
weish08	5605	40	5	weish23	8344	80	5
weish09	5246	40	5	Weish24	10,220	80	5
weish10	6339	50	5	Weish25	9939	80	5
weish11	5643	50	5	Weish26	9584	90	5
weish12	6339	50	5	weish27	9819	90	5
weish13	6159	50	5	weish28	9492	90	5
weish14	6954	60	5	weish29	9410	90	5
weish15	7486	60	5	weish30	11,191	90	5

Afterward, the most critical factors that affect the proposed algorithm's performance are getting to the optimal values of its parameters. Therefore, in this section, extensive experiments have been performed to find the optimal parameters for the proposed algorithm's four parameters, and they are η_c, b, c, and P. After running the proposed algorithm: BHLSO 25 independent runs with various values for each parameter and depicted the findings in Fig. 3, it is evident from this figure that the best values for those four parameters: η_c, b, c, and P are 5.0, 1.2, 1.5, and 0.6, respectively.



a)Tuning of η_c parameter

b)Tuning of b parameter

c)Tuning of c parameter

c)Tuning of P parameter

Fig. 3 Tuning of the proposed algorithm's parameters.

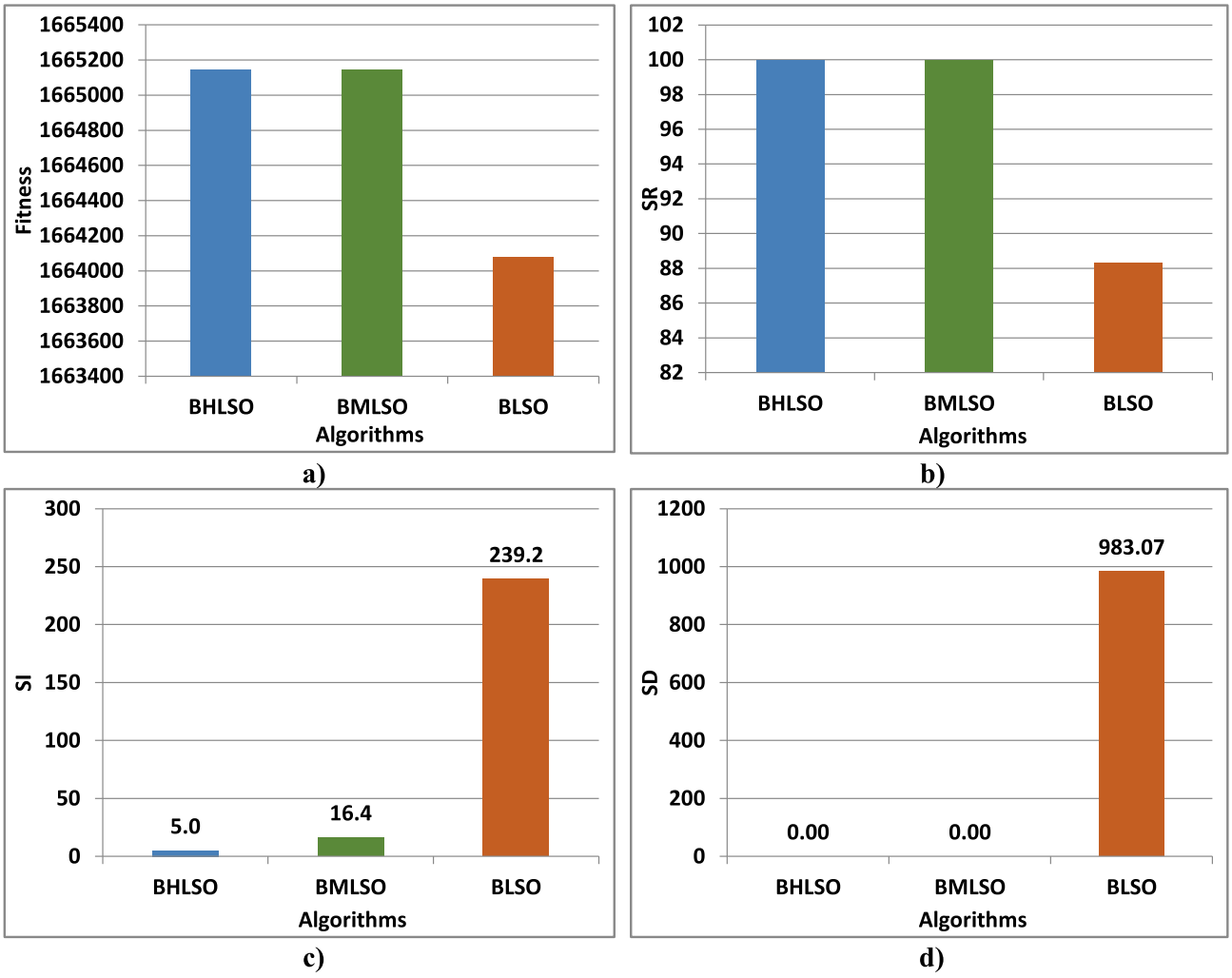


Fig. 4 Comparison between BHLSO, BMLSO, and BLSO in terms of various performance metrics over the instances: KP1- KP12e.

To check the effectiveness of the proposed algorithm, it is compared with a number of the robust recent optimization algorithms using their parameters in the cited paper. Those algorithms are the binary whale optimization algorithm (BWOA) [53], the binary Harris hawks algorithm (BHHA) [54], the binary Sine Cosine algorithm(BSCA) [55], the binary slap swarm algorithm(BSSA) [56], the binary marine predators algorithm (BMPA) [18], the binary equilibrium optimizer (BEOA) [16], elitism genetic algorithm (εGA) [57], and differential evolution (DE) [58]. All algorithms are executed 25 independent runs for each instance with $T = 1000$ and $N = 20$ for the KP01, and $T = 5000$ and $N = 20$ for the MKP To ensure a fair comparison among the algorithms, Ultimately, it is worth mentioning that our experiments have been conducted on a device with 8 GB of RAM, Intel® Core™ i3-2330 M CPU @ 2.20 GHz, and 32-bit windows 10. Those experiments are coded using Java programming language.

4.2. Experiment 1: Comparison between BLSO, BMLSO, and BHLSO

This section reports the findings of several experiments conducted to observe the performance of the classical BLSO, BHLSO, and BMLSO as shown in Fig. 4. In general sense, this figure shows the average of various performance metrics: Fitness, SI, SR, and SD of the instances from KP1 to KP12e for various proposed algorithms to show which one is the best. Inspecting this figure affirms that BHLSO and BMLSO are better than the classical BLSO for all reported performance metrics, and hence the modified variant and hybrid one have an effective performance compared to the classical one. Therefore, our improvements to the classical LSO positively affect its performance. This figure also affirms that BHLSO has a faster convergence to the optimal solutions because it could fulfill the lowest SI in relative to BMLSO (See Fig. 4(c)). It is worth mentioning that the performance of both BHLSO and BMLSO is competitive for the other performance metrics, so both of these will be used in the next experiments to be compared with the other rival algorithms.

4.3. Experiment 2: 0–1 knapsack problem

In this section, our experiments conducted to check the efficacy of BHLSO over the KP01 problems are shown in detail. After running each algorithm 25 independent runs on the instances from KP1 to KP10, the best, average, worst, SD, SR(%),

Table 4 Comparison of the instances from KP1 and KP10.

Name	Opt		BHLSO	BMLSO	BMPA [18]	BSSA [56]	BWOA [53]	BSCA [55]	BEOA [16]	εGA [57]	DE [59]
KP1	295	Best	**295**	**295**	**295**	**295**	**295**	**295**	**295**	293	**295**
		Average	**295**	**295**	**295**	**295**	**295**	**295**	**295**	294	**295**
		Worst	**295**	**295**	**295**	**295**	**295**	**295**	**295**	**295**	**295**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	25	**100**
		SI	**1.6**	2.65	4.1	5	41.45	4.9	4.1	3750	3
KP2	1024	Best	**1024**	**1024**	**1024**	**1024**	**1024**	**1024**	**1024**	769	**1024**
		Average	**1024**	**1024**	**1024**	**1024**	**1024**	**1024**	**1024**	867	**1024**
		Worst	**1024**	**1024**	**1024**	**1024**	**1024**	**1024**	**1024**	999	**1024**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	64	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	0	**100**
		SI	**1.15**	2.15	1.85	13	5.65	9.7	7.65	5000	3
KP3	35	Best	**35**	**35**	**35**	**35**	**35**	**35**	**35**	33	**35**
		Average	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
		Worst	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	95	**100**
		SI	**0**	**0**	**0**	**0**	**0**	**0**	**0**	250	**0**
KP4	23	Best	**23**	**23**	**23**	**23**	**23**	**23**	**23**	19	**23**
		Average	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
		Worst	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	85	**100**
		SI	0.5	**0.15**	**0.15**	0.4	0.85	0.2	0.4	750	**1**
KP5	481.069368	Best	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	425	**481.069368**
		Average	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	474	**481.069368**
		Worst	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**	**481.069368**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	17	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	85	**100**
		SI	**0**	**0**	**0**	0.45	0.35	0.25	**0**	751	**0**
KP6	52	Best	**52**	**52**	**52**	**52**	**52**	**52**	**52**	**52**	**52**
		Average	**52**	**52**	**52**	**52**	**52**	**52**	**52**	**52**	**52**
		Worst	**52**	**52**	**52**	**52**	**52**	**52**	**52**	**52**	**52**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
		SI	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	**0**
KP7	107	Best	**107**	**107**	**107**	**107**	**107**	**107**	**107**	105	**107**
		Average	**107**	**107**	**107**	**107**	**107**	**107**	**107**	**107**	**107**
		Worst	**107**	**107**	**107**	**107**	**107**	**107**	**107**	**107**	**107**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	85	**100**
		SI	**0**	**0**	**0.05**	**0.25**	**4.85**	**1.05**	**0.05**	751	**0**
KP8	9767	Best	**9767**	**9767**	**9767**	**9767**	**9767**	**9767**	**9767**	9733	**9767**
		Average	**9767**	**9767**	**9767**	9766.8	9764.55	9766.4	**9767**	9743	**9767**
		Worst	**9767**	**9767**	**9767**	9763	9763	9762	**9767**	9753	**9767**
		SD	**0**	**0**	**0**	0.872	1.564	1.241	**0**	6	**0**
		SR(%)	**100**	**100**	**100**	95	20	75	**100**	0	**100**
		SI	**30.85**	107.15	42.5	280	780	452.95	49.2	5000	60
KP9	130	Best	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**
		Average	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**
		Worst	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
		SI	**0**	**0**	**0**	0.1	0.1	**0**	**0**	**0**	**0**
KP10	1025	Best	**1025**	**1025**	**1025**	**1025**	**1025**	**1025**	**1025**	771	**1025**
		Average	**1025**	**1025**	**1025**	1024.7	**1025**	**1025**	**1025**	856	**1025**
		Worst	**1025**	**1025**	**1025**	1019	**1025**	**1025**	**1025**	991	**1025**
		SD	**0**	**0**	**0**	1.308	**0**	**0**	**0**	61	**0**
		SR(%)	**100**	**100**	**100**	95	**100**	**100**	**100**	0	**100**
		SI	**1.35**	1.7	1.4	55	5.3	6.2	6.3	5000	4

Table 5 Summary of table 4.

	BHLSO	BMLSO	BMPA [18]	BSSA [56]	BWOA [53]	BSCA [55]	BEOA [16]	εGA [57]	DE [59]
Avg. Fitness	**1293.907**	**1293.907**	**1293.907**	1293.857	1293.662	1293.847	**1293.907**	1258.043	1293.9069
Avg. SD	**0**	**0**	**0**	0.218	0.1564	0.1241	**0**	15.15978	0
Avg. SR(%)	**100**	**100**	**100**	99	92	97.5	**100**	57.5	100
Avg. SI	**3.545**	11.38	5.005	35.42	83.855	47.525	6.77	2125.31	7.13

Table 6 Comparison of the instances from KP11 and KP20.

Name	Opt		BHLSO	BMLSO	BMPA [18]	BSSA [56]	BWOA [53]	BSCA [55]	BEOA [16]	εGA [57]	DE [59]
KP11	1437	Best	**1437**	**1437**	**1437**	**1437**	**1437**	**1437**	**1437**	1026	**1437**
		Average	**1437**	**1437**	**1437**	1436.7	**1437**	**1437**	**1437**	1169	**1437**
		Worst	**1437**	**1437**	**1437**	1431	**1437**	**1437**	**1437**	1402	**1437**
		SD	**0**	**0**	**0**	**1.308**	**0**	**0**	**0**	87	**0**
		SR(%)	**100**	**100**	**100**	**95**	**100**	**100**	**100**	0	**100**
		SI	**1.5**	4.3	2.3	89	1.7	1.55	47.1	5000	16
KP12	1689	Best	**1689**	**1689**	**1689**	**1689**	**1689**	**1689**	**1689**	1222	**1689**
		Average	**1689**	**1689**	**1689**	1688.7	**1689**	**1689**	**1689**	1322	**1689**
		Worst	**1689**	**1689**	**1689**	1686	**1689**	**1689**	**1689**	1573	**1689**
		SD	**0**	**0**	**0**	0.900	**0**	**0**	**0**	94	**0**
		SR(%)	**100**	**100**	**100**	90	**100**	**100**	**100**	0	**100**
		SI	**3.05**	8.65	7.3	264	22.85	31.05	92.7	5000	44
KP13	1821	Best	**1821**	**1821**	**1821**	**1821**	**1821**	**1821**	**1821**	1236	**1821**
		Average	**1821**	**1821**	**1821**	1815.45	**1821**	**1821**	**1821**	1342	**1821**
		Worst	**1821**	**1821**	**1821**	1785	**1821**	**1821**	**1821**	1532	**1821**
		SD	**0**	**0**	**0**	9.075	**0**	**0**	**0**	69	**0**
		SR(%)	**100**	**100**	**100**	55	**100**	**100**	**100**	0	**100**
		SI	**5.4**	17.9	6.9	500	49.5	38.25	240.2	5000	73
KP14	2033	Best	**2033**	**2033**	**2033**	**2033**	**2033**	**2033**	**2033**	1349	**2033**
		Average	**2033**	**2033**	**2033**	2029.05	**2033**	**2033**	**2033**	1481	**2033**
		Worst	**2033**	**2033**	**2033**	1995	**2033**	**2033**	**2033**	1683	**2033**
		SD	**0**	**0**	**0**	10.210	**0**	**0**	**0**	95	**0**
		SR(%)	**100**	**100**	**100**	85	**100**	**100**	**100**	0	**100**
		SI	**2.15**	5	2.95	467.2	1.9	1.2	158.8	5000	35
KP15	2440	Best	**2444**	**2444**	**2444**	**2444**	**2444**	**2444**	**2444**	1572	2440
		Average	**2444**	**2444**	**2444**	2438.21	**2444**	**2444**	**2444**	1747	2442
		Worst	**2444**	**2444**	**2444**	2421	**2444**	**2444**	**2444**	2126	**2444**
		SD	**0**	**0**	**0**	6.308	**0**	**0**	**0**	121	2
		SR(%)	**100**	**100**	**100**	20	**100**	**100**	**100**	0	50
		SI	**3.5**	14.9	5	807	210.75	364.6	67.55	5000	546
KP16	2651	Best	**2651**	**2651**	**2651**	**2651**	**2651**	**2651**	**2651**	1710	**2651**
		Average	**2651**	**2651**	**2651**	2624.85	**2651**	**2651**	**2651**	1862	**2651**
		Worst	**2651**	**2651**	**2651**	2523	**2651**	**2651**	**2651**	2058	**2651**
		SD	**0**	**0**	**0**	31.399	**0**	**0**	**0**	89	**0**
		SR(%)	**100**	**100**	**100**	15.0	**100**	**100**	**100**	0	**100**
		SI	3.8	24.25	6	921	1.85	**1.8**	427.5	5000	169
KP17	2917	Best	**2917**	**2917**	**2917**	**2917**	**2917**	**2917**	**2917**	1771	**2917**
		Average	**2917**	**2917**	**2917**	2880.9	**2917**	**2917**	**2917**	2013	**2917**
		Worst	**2917**	**2917**	**2917**	2819	**2917**	**2917**	**2917**	2233	**2917**
		SD	**0**	**0**	**0**	33.621	**0**	**0**	**0**	126	**0**
		SR(%)	**100**	**100**	**100**	35	**100**	**100**	**100**	0	**100**
		SI	1.65	9.25	3	929	1.75	**1.3**	431.3	5000	90
KP18	2817	Best	**2818**	**2818**	**2818**	2801	**2818**	**2818**	**2818**	1803	2817
		Average	**2818**	**2818**	**2818**	2763.2	**2818**	**2818**	**2818**	1946	2817
		Worst	**2818**	**2818**	**2818**	2675	**2818**	**2818**	**2818**	2097	**2818**
		SD	**0**	**0**	**0**	34.599	**0**	**0**	**0**	77	**0**
		SR(%)	**100**	**100**	**100**	0	**100**	**100**	**100**	0	70
		SI	**1.95**	13.45	2.25	950	44	67.65	248.75	5000	440

Table 6 (*continued*)

Name	Opt		BHLSO	BMLSO	BMPA [18]	BSSA [56]	BWOA [53]	BSCA [55]	BEOA [16]	εGA [57]	DE [59]
KP19	3223	Best	**3223**	**3223**	**3223**	**3223**	**3223**	**3223**	**3223**	1998	3220
		Average	**3223**	**3223**	3222.9	3165.85	3221.5	3221.4	3222	2246	3222
		Worst	**3223**	**3223**	3221	3078	3221	3221	3221	2471	**3223**
		SD	**0**	**0**	0.436	40.566	0.866	0.800	1.322	109	1
		SR(%)	**100**	**100**	95	5	25	20	65	0	75
		SI	**88**	170.2	120	1000	580	868	446.6	5000	728
KP20	3614	Best	**3614**	**3614**	**3614**	**3614**	**3614**	**3614**	**3614**	2317	**3614**
		Average	**3614**	**3614**	**3614**	3545.7	**3614**	3613.15	**3614**	2455	**3614**
		Worst	**3614**	**3614**	**3614**	3388	**3614**	3597	**3614**	2707	**3614**
		SD	**0**	**0**	**0**	53.101	**0**	3.705	**0**	135	**0**
		SR(%)	**100**	**100**	**100**	5.0	**100**	95	**100**	0	**100**
		SI	**3.2**	13.15	6	985.2	57.7	91.8	599.65	5000	269

Table 7 Summary of table 6.

	BHLSO	BMLSO	BMPA [18]	BSSA [56]	BWOA [53]	BSCA [55]	BEOA [16]	εGA [57]	DE [59]
Avg. Fitness	**2464.7**	**2464.7**	2464.69	2438.861	2464.55	2464.455	2464.6	1758.3	2464.365
Avg. SD	**0**	**0**	0.0436	22.1087	0.0866	0.4505	0.1322	100.270	0.361042
Avg. SR(%)	**100**	**100**	99.5	40.5	92.5	91.5	96.5	0	89.5
Avg. SI	**11.42**	28.105	16.17	691.24	97.2	146.72	276.015	5000	241.09

Table 8 Comparison of the instances from KP11 and KP20.

Name	Opt		BHLSO	BMLSO	BMPA [18]	BSSA [56]	BWOA [53]	BSCA [55]	BEOA [16]	εGA [57]	DE [59]
KP8a	**3,924,400**	Best	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	3,865,959	**3,924,400**
		Average	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	3,921,478	**3,924,400**
		Worst	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**	**3,924,400**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	12,737	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	95	**100**
		SI	**0**	**0**	**0**	**0**	**0**	**0**	**0**	251	**0**
KP8b	**3,813,669**	Best	**3,813,669**	**3,813,669**	**3,813,669**	**3,813,669**	**3,813,669**	**3,813,669**	**3,813,669**	3,710,150	**3,813,669**
		Average	**3,813,669**	**3,813,669**	3,813,669	3812119.4	**3,813,669**	**3,813,669**	**3,813,669**	3,786,166	**3,813,669**
		Worst	**3,813,669**	**3,813,669**	**3,813,669**	3,782,677	**3,813,669**	**3,813,669**	**3,813,669**	**3,813,669**	**3,813,669**
		SD	**0**	**0**	**0**	6754.550	**0**	**0**	**0**	32,897	**0**
		SR(%)	**100**	**100**	**100**	95	**100**	**100**	**100**	50	**100**
		SI	**0**	18.35	20.2	117.25	139.4	40.35	16.3	2500	11
KP8c	**3,347,452**	Best	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	3,311,541	**3,347,452**
		Average	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	3,339,952	**3,347,452**
		Worst	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**	**3,347,452**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	10,787	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	55	**100**
		SI	**0**	1.55	1.75	3.05	5.05	1.8	2.1	2250	4
KP8d	**4,187,707**	Best	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	4,072,147	**4,187,707**
		Average	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	4,179,168	**4,187,707**
		Worst	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**	**4,187,707**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	27,336	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	90	**100**
		SI	**0**	**0**	**0**	**0**	**0.05**	**0**	**0**	501	**0**
KP8e	**4,955,555**	Best	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	4,856,657	**4,955,555**
		Average	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	4,945,665	**4,955,555**
		Worst	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**	**4,955,555**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	29,669	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	90	**100**
		SI	**0**	0.05	0.05	0.25	0.3	**0**	0.1	501	**0**

(*continued on next page*)

Table 8 (*continued*)

Name	Opt		BHLSO	BMLSO	BMPA [18]	BSSA [56]	BWOA [53]	BSCA [55]	BEOA [16]	εGA [57]	DE [59]
KP12a	**5,688,887**	Best	**5,688,887**	**5,688,887**	**5,688,887**	**5,688,887**	**5,688,887**	**5,688,887**	**5,688,887**	5,636,813	**5,688,887**
		Average	**5,688,887**	**5,688,887**	**5,688,887**	5686942.1	5685593.3	5688238.7	**5,688,887**	5,659,643	**5,688,887**
		Worst	**5,688,887**	**5,688,887**	**5,688,887**	5,682,404	5,681,360	5,682,404	**5,688,887**	5,682,404	**5,688,887**
		SD	**0**	**0**	**0**	2970.884	3301.7	1944.900	**0**	18,838	**0**
		SR(%)	**100**	**100**	**100**	70	50	90	**100**	0	**100**
		SI	**0**	76.75	28.15	146.2	363.1	121	25.4	5000	31
KP12b	**6,473,019**	Best	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	6,414,452	6,473,019
		Average	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	6,472,950	6,480,692
		Worst	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**	**6,498,597**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	18,157	11,721
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	55	70
		SI	**0**	**0**	**0**	0.2	150.65	350	**0**	2251	300
KP12c	**5,170,626**	Best	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	5,108,920	**5,170,626**
		Average	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	5,167,541	**5,170,626**
		Worst	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**	**5,170,626**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	13,449	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	95	**100**
		SI	**0**	**0**	**0**	0.05	0.1	**0**	**0**	251	**0**
KP12d	**6,992,404**	Best	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	6,881,630	6,941,564
		Average	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	6,971,842	6,989,862
		Worst	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**	**6,992,404**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	30,215	11,080
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	10	5
		SI	**0**	**0**	**0**	950	750	950	0.05	4500	950
KP12e	**5,337,472**	Best	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	5,289,570	**5,337,472**
		Average	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	5,308,731	**5,337,472**
		Worst	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**	**5,337,472**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	23,467	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	40	**100**
		SI	**0**	**0**	0.35	1.1	6.5	1.75	0.45	3000	**1**

Bold values indicate the best results.

and SI are introduced in Table 4. Inspecting this table shows that all algorithms are competitive in terms of the best, average, worst, SD, and SR(%). Also, this table confirms the superiority of the proposed algorithm in terms of SI and this shows that our proposed algorithm has a higher convergence speed in comparison to the other algorithms. To confirm the superiority of BHLSO on those instances, the average of the fitness values, SD, SR(%), and SI on all the instances are computed and introduced in Table 5. From this table, it is clear that BHLSO, BMLSO, BEOA, and BMPA could come true the same values for the average of the fitness values, SD, and SR(%), but our proposed could outperform all for the SI with a value of 3.545.

Likewise for the instances from KP11 to KP20, and the instances from KP8a to KP12e, the algorithms are running on those instances, and the performance metrics outcomes on each one are presented in Table 6 and 8, and the average of the same metrics are in Table 7 and 9, respectively. From those tables, it is observed that BHLSO, BMLSO, BSCA, DE, and BEOA are competitive in terms of the average of the fitness values, SD, and SR(%), while BHLSO could outperform all in terms of SI.

Table 10 shows the outcomes of the algorithms on large-scale instances with various correlations between the profits and weights to see their abilities in estimating the optimal solution under different conditions. Checking this table show that the performance of all compared algorithms is competitive with BHLSO and BMLSO even the instances with dimensions up to 200. But, the efficiency of BHLSO and BMLSO is competitive with DE and BMPA and superior to the other

Table 9 Summary of table 8.

	BHLSO	BMLSO	BMPA [18]	BSSA [56]	BWOA [53]	BSCA [55]	BEOA [16]	εGA [57]	DE [59]
Avg. Fitness	**4,991,677**	**4,991,677**	**4,991,677**	4,991,327	4,991,348	4,991,612	**4,991,677**	4975313.535	4989632.24
Avg. SD	**0**	**0**	**0**	972.5434	330.17	194.49	**0**	21755.277	2280.163
Avg. SR(%)	**100**	**100**	**100**	96.5	95	99	**100**	58	87.5
Avg. SI	**0**	9.67	5.05	121.81	141.515	146.49	4.44	2100.51	129.7

Table 10 Comparison of the instances from KP1_100 and KP3_2000.

Instance	Opt		BHLSO	BMLSO	BMPA[18]	BSSA[56]	BWOA[53]	BSCA[55]	BEOA[16]	DE [59]
KP1_100	9147	Best	**9147**	**9147**	**9147**	**9147**	**9147**	**9147**	**9147**	**9147**
		Average	**9147**	**9147**	**9147**	**9147**	**9147**	**9147**	**9147**	**9147**
		Worst	**9147**	**9147**	**9147**	**9147**	**9147**	**9147**	**9147**	**9147**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
KP1_200	11,238	Best	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**
		Average	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**
		Worst	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**	**11,238**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
KP1_500	28,857	Best	**28,857**	**28,857**	**28,857**	**28,857**	**28,857**	**28,857**	**28,857**	**28,857**
		Average	**28,857**	**28,857**	**28,857**	28821.1	28854.7	28850.1	28855.85	**28,857**
		Worst	**28,857**	**28,857**	**28,857**	28,709	28,834	28,834	28,834	**28,857**
		SD	**0**	**0**	**0**	41.244	6.900	10.540	5.013	**0**
		SR(%)	**100**	**100**	**100**	25	90	70	95	**100**
KP1_1000	54,503	Best	**54,503**	**54,503**	**54,503**	54,011	54,460	54,394	**54,503**	54,389
		Average	**54,503**	**54,503**	**54,503**	53350.2	53842.9	53769.15	54459.8	54,479
		Worst	**54,503**	**54,503**	**54,503**	52,642	53,254	52,428	54,370	**54,503**
		SD	**0**	**0**	**0**	372.154	366.443	545.127	39.135	32
		SR(%)	**100**	**100**	**100**	0	0	5	30	45
KP1_2000	110,625	Best	**110,625**	**110,625**	**110,625**	106,409	101,162	100,649	110,582	108,500
		Average	**110606.4**	110599.9	110593.25	105187.65	93568.65	95780.5	109845.6	109,328
		Worst	**110,578**	**110,578**	**110,578**	102,734	89,332	92,409	109,047	110,547
		SD	**14.924**	15.396	16.982	908.683	2760.740	2340.451	433.217	469
		SR(%)	**15**	**15**	**5.0**	0	0	0	0	0
KP2_100	1514	Best	**1514**	**1514**	**1514**	**1514**	**1514**	**1514**	**1514**	**1514**
		Average	**1514**	**1514**	**1514**	**1514**	**1514**	**1514**	**1514**	**1514**
		Worst	**1514**	**1514**	**1514**	**1514**	**1514**	**1514**	**1514**	**1514**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
KP2_200	1634	Best	**1634**	**1634**	**1634**	**1634**	**1634**	**1634**	**1634**	**1634**
		Average	**1634**	**1634**	**1634**	**1634**	**1634**	**1634**	**1634**	**1634**
		Worst	**1634**	**1634**	**1634**	**1634**	**1634**	**1634**	**1634**	**1634**
		SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
KP2_500	4566	Best	**4566**	**4566**	**4566**	**4566**	4552	**4556**	**4566**	**4566**
		Average	**4566**	**4565.4**	**4559.05**	4559.9	4508.3	4530.3	4564.6	**4566**
		Worst	**4566**	**4559**	**4556**	4552	4446	4497	4559	**4566**
		SD	**0**	**1.96**	**4.141**	4.795	28.313	20.943	2.499	**0**
		SR(%)	**100**	**95**	**25**	35	0	0	85	**100**
KP2_1000	9052	Best	**9052**	**9052**	**9051**	**9041**	8473	8676	**9052**	9051
		Average	**9051.2**	**9051.2**	9051	9014.7	8212.85	8419.75	9050.75	9051
		Worst	**9051**	**9051**	**9051**	8975	7966	8249	9048	9051
		SD	**0.357**	**0.400**	**0**	19.756	135.720	132.221	1.299	0
		SR(%)	**15**	**20**	**0**	0	0	0	30	0
KP2_2000	18,051	Best	**18,050**	**18,050**	**18,050**	17,888	15,225	15,367	18,048	17,919
		Average	**18049.3**	**18049.1**	18048.2	17774.15	14710.65	15103.95	18032.15	18,015
		Worst	**18,046**	**18,046**	**18,046**	17,575	14,387	14,711	17,992	**18,046**
		SD	**1.187**	**1.411**	1.600	83.152	242.453	185.441	13.499	30
		SR(%)	**0**	**0**	**0**	0	0	0	0	0
KP3_100	2397	Best	**2397**	**2397**	**2397**	**2397**	**2397**	2397	**2397**	**2397**
		Average	**2397**	**2397**	**2397**	2396.9	2395.1	2396.75	**2397**	**2397**
		Worst	**2397**	**2397**	**2397**	2396	2390	2396	**2397**	**2397**
		SD	**0**	**0**	**0**	35.243	**2.587**	0.433	**0**	**0**
		SR(%)	**100**	**100**	**100**	50	30	75	**100**	**100**
KP3_200	2697	Best	**2697**	**2697**	**2697**	**2697**	**2697**	2697	**2697**	**2697**
		Average	**2697**	**2697**	**2697**	**2697**	2689.75	2662.2	**2697**	**2697**
		Worst	**2697**	**2697**	**2697**	**2697**	2672	2497	**2697**	**2697**
		SD	**0**	**0**	**0**	**0**	7.042	50.800	**0**	**0**
		SR(%)	**100**	**100**	**100**	**100**	30	15	**100**	**100**

(*continued on next page*)

Table 10 (*continued*)

Instance	Opt		BHLSO	BMLSO	BMPA[18]	BSSA[56]	BWOA[53]	BSCA[55]	BEOA[16]	DE [59]
KP3_500	7117	Best	**7117**	**7117**	**7117**	**7117**	5714	5917	**7117**	**7117**
		Average	**7117**	**7117**	**7117**	7100.25	5584.25	5732.35	**7117**	**7117**
		Worst	**7117**	**7117**	**7117**	7017	5503	5517	**7117**	**7117**
		SD	**0**	**0**	**0**	35.243	60.317	97.951	**0**	**0**
		SR(%)	**100**	**100**	**100**	50	0	0	**100**	**100**
KP3_1000	14,390	Best	**14,390**	**14,390**	**14,390**	14,287	9690	9678	**14,390**	14,290
		Average	**14,390**	**14,390**	**14,390**	14133.65	9374.95	9473.85	14324.8	14,363
		Worst	**14,390**	**14,390**	**14,390**	14,074	9142	9288	14,290	**14,390**
		SD	**0**	**0**	**0**	62.345	144.637	106.080	47.428	42
		SR(%)	**100**	**100**	**100**	0	0	0	25	40
KP3_2000	28,919	Best	**28,919**	**28,919**	**28,919**	28,217	16,100	16,519	**28,919**	28,319
		Average	**28,919**	**28,919**	**28,919**	27850.25	15853.5	16082.2	28723.6	28,573
		Worst	**28,919**	**28,919**	**28,919**	27,617	15,495	15,805	28,419	28,815
		SD	**0**	**0**	**0**	157.711	174.963	166.435	132.022	130
		SR(%)	**100**	**100**	**100**	0	0	0	5	0

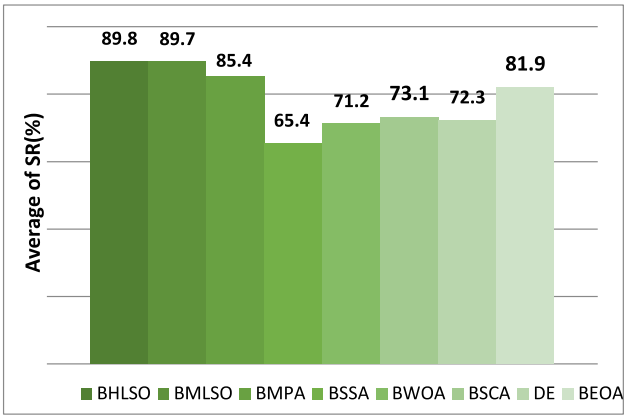


Fig. 5 Comparison in terms of SR(%).

for KP1_500 and KP1_1000. For KP1_2000, KP2_500 and KP2_2000; in another word, the improved variant could be competitive with BLSO and BMPA in terms of the best, worst, and SR(%) values and superior in terms of the average value and the SD. for the other instances, BHLSO, BLSO, and BMPA were competitive with each other and superior to the others. To check the efficiency of algorithms on all instances, Fig. 5 is presented to compute the average of the SR on all the instances. From this figure, BHLSO could occupy the first rank with a value of 89.8, followed by BMLSO as the second-best one, while BSSA comes in the last rank with an amount of 65.4.

In addition, Table 11 presents the p-value obtained under the Wilcoxon rank-sum test for the large-scale instances to show the difference between the outcomes of the proposed algorithm and those of each rival algorithm. According to this

Table 11 Significance analysis under Wilcoxon rank-sum test for large-scale instances.

Instance	BMLSO	BMPA[18]	BSSA[56]	BWOA[53]	BSCA[55]	BEOA[16]	DE [59]
KP1_100	NaN	NaN	NaN	NaN	NaN	NaN	NaN
KP1_200	NaN	NaN	NaN	NaN	NaN	NaN	NaN
KP1_500	NaN	NaN	**2.7855E-06**	1.6245E-01	**9.2800E-03**	3.4211E-01	NaN
KP1_1000	NaN	NaN	**8.0065E-09**	**7.6609E-09**	**2.9661E-08**	**9.3664E-06**	**1.6585E-04**
KP1_2000	2.4275E-01	3.5772E-01	**5.8941E-08**	**5.8941E-08**	**5.8941E-08**	**1.4491E-07**	**5.8941E-08**
KP2_100	NaN	NaN	NaN	NaN	NaN	NaN	NaN
KP2_200	NaN	NaN	NaN	NaN	NaN	NaN	NaN
KP2_500	3.4211E-01	**2.5944E-06**	**2.4955E-05**	**7.9043E-09**	**7.6327E-09**	8.0359E-02	NaN
KP2_1000	6.9628E-01	8.0359E-02	**1.9383E-08**	**1.9447E-08**	**1.9447E-08**	7.0084E-01	6.9628E-01
KP2_2000	5.9936E-01	3.8016E-01	**3.4516E-08**	**3.4569E-08**	**3.4569E-08**	**9.9175E-08**	**5.8615E-08**
KP3_100	NaN	NaN	1.6245E-01	**7.4346E-06**	**1.9450E-02**	NaN	NaN
KP3_200	NaN	NaN	NaN	**9.2056E-06**	**3.4729E-07**	NaN	NaN
KP3_500	NaN	NaN	**3.9744E-04**	**7.9772E-09**	**7.9772E-09**	NaN	NaN
KP3_1000	NaN	NaN	**7.6751E-09**	**7.9919E-09**	**7.9919E-09**	**1.9413E-06**	**6.5160E-05**
KP3_2000	NaN	NaN	**7.9334E-09**	**8.0065E-09**	**7.9919E-09**	**2.6754E-08**	**7.8321E-09**

table, BHLSO has significantly different outcomes from the majority of its competitors in most large-scale instances. Furthermore, to check the speedup of the algorithms, Fig. 6 is introduced to show the average of the CPU time consumed by each algorithm on the instances from KP01 to KP12e. This figure reveals that both BHLSO and BMLSO need less CPU time than all competitors.

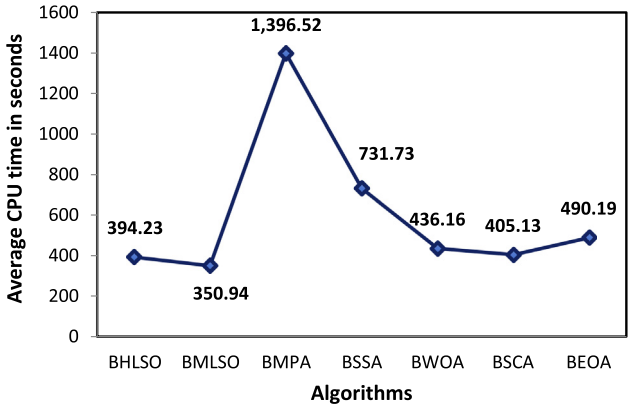


Fig. 6 Comparison of the CPU time on the datasets from KP1 to KP12e.

4.4. Experiment 3: Multi-dimensional knapsack experiments

In the previous section, it was proved that BMLSO and BHLSO could be superior compared to the other recent robust optimization algorithms for KP01 problems on 45 common instances. This significant success for KP01 motivates us to check their performance in this section for MKP. After running each algorithm 25 independent runs on the instances from weish01 to weish11, the best, avg and worst fitness values obtained within those runs for each instance are introduced in Table 12, and the average of the fitness values on all instances are given in Fig. 7. Inspecting this figure shows that our proposed algorithm could slightly outperform the other algorithm. Therefore, in the next table, more instances with a high number of items and knapsack dimensions are used to see the stability of the algorithms.

Table 12 Comparison of the instances from weish01 and weish11.

Id		BHLSO	BMLSO	BMPA[18]	BSSA[56]	BWOA[53]	BSCA[55]	MMVO[32]	BEOA[16]
weish01	**Best**	**4554.0000**	**4554.0000**	**4554.0000**	**4554.0000**	**4554.0000**	**4554.0000**	**4554.0000**	**4554.0000**
	Avg	**4554.0000**	4553.4000	4552.8000	4554.0000	4542.2000	4542.6800	**4554.0000**	**4554.0000**
	Worst	**4554.0000**	4549.0000	4549.0000	4554.0000	4534.0000	4516.0000	**4554.0000**	**4554.0000**
weish02	**Best**	**4536.0000**	**4536.0000**	**4536.0000**	**4536.0000**	**4536.0000**	**4536.0000**	**4536.0000**	**4536.0000**
	Avg	**4536.0000**	**4536.0000**	**4536.0000**	**4536.0000**	4528.2000	4523.0000	**4536.0000**	**4536.0000**
	Worst	**4536.0000**	**4536.0000**	**4536.0000**	**4536.0000**	4504.0000	4504.0000	**4536.0000**	**4536.0000**
weish03	**Best**	**4115.0000**	**4115.0000**	**4115.0000**	**4115.0000**	**4115.0000**	**4115.0000**	**4106.0000**	**4115.0000**
	Avg	4112.8400	4109.2400	4106.3600	**4114.9200**	4095.5600	4052.0000	4101.7600	4113.4000
	Worst	**4106.0000**	**4106.0000**	**4106.0000**	**4106.0000**	4052.0000	4096.6400	4052.0000	**4106.0000**
weish04	**Best**	**4561.0000**	**4561.0000**	**4561.0000**	**4561.0000**	**4561.0000**	**4561.0000**	**4561.0000**	**4561.0000**
	Avg	**4561.0000**	**4561.0000**	**4561.0000**	**4561.0000**	4545.3200	4559.8000	**4561.0000**	**4561.0000**
	Worst	**4561.0000**	**4561.0000**	**4561.0000**	**4561.0000**	4505.0000	4531.0000	**4561.0000**	**4561.0000**
weish05	**Best**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**
	Avg	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**
	Worst	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**	**4514.0000**
weish06	**Best**	**5557.0000**	**5557.0000**	**5557.0000**	**5557.0000**	**5557.0000**	**5557.0000**	5542.0000	**5557.0000**
	Avg	**5557.0000**	5551.7600	5548.6000	5555.2800	5514.8400	5518.1200	5530.1600	**5557.0000**
	Worst	**5557.0000**	5542.0000	5542.0000	5542.0000	5506.0000	5542.0000	5517.0000	**5557.0000**
weish07	**Best**	**5567.0000**	**5567.0000**	**5567.0000**	**5567.0000**	**5567.0000**	**5567.0000**	**5567.0000**	**5567.0000**
	Avg	**5567.0000**	**5567.0000**	**5567.0000**	**5567.0000**	5548.4400	5546.6400	**5567.0000**	**5567.0000**
	Worst	**5567.0000**	**5567.0000**	**5567.0000**	**5567.0000**	5452.0000	5452.0000	**5567.0000**	**5567.0000**
weish08	**Best**	**5605.0000**	**5605.0000**	**5605.0000**	**5605.0000**	**5605.0000**	**5605.0000**	**5605.0000**	**5605.0000**
	Avg	**5605.0000**	5604.0400	5603.8000	**5605.0000**	5595.3200	5553.5600	5603.9600	**5605.0000**
	Worst	**5605.0000**	5603.000	5603.0000	**5605.0000**	5506.0000	5452.0000	5603.0000	**5605.0000**
weish09	**Best**	**5246.0000**	**5246.0000**	**5246.0000**	**5246.0000**	**5246.0000**	**5246.0000**	**5246.0000**	**5246.0000**
	Avg	**5246.0000**	**5246.0000**	**5246.0000**	**5246.0000**	5226.9600	5241.9200	**5246.0000**	**5246.0000**
	Worst	**5246.0000**	**5246.0000**	**5246.0000**	**5246.0000**	5212.0000	5212.0000	**5246.0000**	**5246.0000**
Weish10	**Best**	**6323.0000**	**6303.0000**	**6303.0000**	**6323.0000**	**6323.0000**	**6323.0000**	**6323.0000**	**6323.0000**
	Avg	**6308.5200**	6302.9200	6303.0000	6307.0000	6288.9200	6287.7200	6307.7600	6306.9600
	Worst	**6303.0000**	6301.0000	**6303.0000**	**6303.0000**	6265.0000	6242.0000	6265.0000	**6303.0000**
Weish11	**Best**	**5643.0000**	**5643.0000**	**5643.0000**	**5643.0000**	**5643.0000**	**5643.0000**	**5643.0000**	**5643.0000**
	Avg	**5643.0000**	**5643.0000**	**5643.0000**	**5643.0000**	5526.4000	5626.3600	5591.2000	**5643.0000**
	Worst	**5643.0000**	**5643.0000**	**5643.0000**	**5643.0000**	5396.0000	5476.0000	5487.0000	**5643.0000**

Bold values indicate the best results.

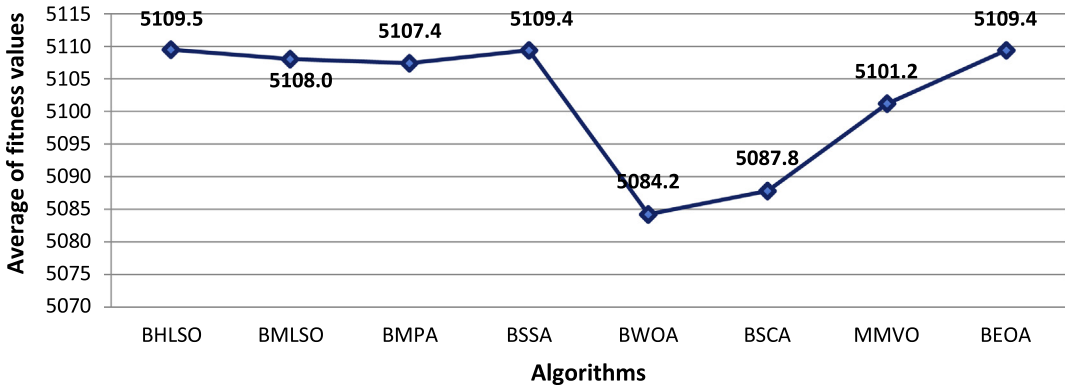


Fig. 7 Comparison in terms of the average of the fitness on the instances from WEISH01 to WEISH11.

Table 13 Comparison of the instances from weish12 and weish22.

Id		BHLSO	BMLSO	BMPA[18]	BSSA[56]	BWOA[53]	BSCA[55]	MMVO[32]	BEOA[16]
Weish12	**Best**	**6302.0000**	**6302.0000**	**6302.0000**	**6302.0000**	6301.0000	**6302.0000**	**6302.0000**	**6302.0000**
	Avg	**6301.9200**	6301.1600	6301.0400	6301.4400	6283.3600	6267.4000	6301.0400	6301.4800
	Worst	**6301.0000**	6301.0000	6301.0000	6301.0000	6154.0000	6117.0000	6301.0000	6301.0000
Weish13	**Best**	**6159.0000**	**6159.0000**	**6159.0000**	**6159.0000**	6122.0000	**6159.0000**	6122.0000	**6159.0000**
	Avg	**6159.0000**	**6154.6400**	**6159.0000**	**6159.0000**	6040.7600	6119.3200	6053.5200	**6159.0000**
	Worst	**6159.0000**	6050.0000	**6159.0000**	**6159.0000**	6025.0000	5964.0000	6025.0000	**6159.0000**
Weish14	**Best**	**6923.0000**	**6923.0000**	**6923.0000**	**6923.0000**	**6923.0000**	**6923.0000**	6900.0000	**6923.0000**
	Avg	**6921.5200**	6908.3600	6910.7600	6916.9200	6868.1200	6880.0400	6876.9600	6915.7200
	Worst	**6900.0000**	6900.0000	6900.0000	6900.0000	6787.0000	6787.0000	6836.0000	6900.0000
Weish15	**Best**	**7486.0000**	**7486.0000**	**7486.0000**	**7486.0000**	7442.0000	**7486.0000**	7416.0000	**7486.0000**
	Avg	**7486.0000**	**7486.0000**	7483.6000	**7486.0000**	7378.4800	7445.6400	7411.0000	**7486.0000**
	Worst	**7486.0000**	**7486.0000**	7456.0000	**7486.0000**	7199.0000	7236.0000	7391.0000	**7486.0000**
Weish16	**Best**	**7289.0000**	**7289.0000**	7288.0000	7288.0000	7288.0000	7288.0000	7288.0000	7288.0000
	Avg	**7288.7667**	7278.8800	7280.7600	7274.4000	7239.7600	7246.4400	7258.3200	7282.4667
	Worst	**7288.0000**	7221.0000	7268.0000	7226.0000	7221.0000	7201.0000	7247.0000	7281.0000
Weish17	**Best**	**8633.0000**	**8633.0000**	**8633.0000**	8556.0000	8624.0000	8575.0000	8621.0000	**8633.0000**
	Avg	**8633.0000**	8628.2000	8567.8800	8455.5200	8555.0000	8447.6400	8575.8800	**8633.0000**
	Worst	**8633.0000**	8592.0000	8506.0000	8326.0000	8457.0000	8390.0000	8532.0000	**8633.0000**
Weish18	**Best**	**9580.0000**	9560.0000	9522.0000	9521.0000	9521.0000	9411.000	9485.0000	**9580.0000**
	Avg	**9577.4800**	9537.4800	9465.4800	9408.2000	9438.3200	9352.4000	9399.0400	9565.6000
	Worst	**9573.0000**	9521.0000	9403.0000	9247.0000	9336.0000	9560.0000	9390.0000	9521.0000
Weish19	**Best**	**7698.0000**	**7698.0000**	**7698.0000**	**7698.0000**	**7636.0000**	**7698.0000**	7629.0000	**7698.0000**
	Avg	**7698.0000**	7695.5200	7680.6400	7667.6000	7546.2400	7542.4000	7432.0000	**7698.0000**
	Worst	**7698.0000**	7636.0000	7636.0000	7620.0000	7330.0000	7194.0000	7577.5600	**7698.0000**
Weish20	**Best**	**9450.0000**	**9450.0000**	**9450.0000**	**9450.0000**	**9408.0000**	**9450.0000**	9400.0000	**9450.0000**
	Avg	**9450.0000**	9447.1200	9430.8800	9400.2800	9371.7600	9266.3600	9389.6000	9449.8000
	Worst	**9450.0000**	9433.0000	9430.8800	9343.0000	9244.0000	8844.0000	9380.0000	9445.0000
Weish21	**Best**	**9074.0000**	**9074.0000**	**9074.0000**	**9074.0000**	**9074.0000**	**9074.0000**	8972.0000	**9074.0000**
	Avg	**9074.0000**	9070.8800	9048.0400	9070.0000	8902.6800	8946.2000	8867.6800	**9074.0000**
	Worst	**9074.0000**	8996.0000	8972.0000	9024.0000	8714.0000	8582.0000	8790.0000	**9074.0000**
Weish22	**Best**	**8929.0000**	8912.0000	8886.0000	8912.0000	8908.0000	8912.0000	8908.0000	**8929.0000**
	Avg	**8929.0000**	8895.7200	8886.0000	8875.8800	8888.6400	8796.9200	8908.0000	8908.3200
	Worst	**8929.0000**	8886.0000	8886.0000	8723.0000	8886.0000	8275.0000	8908.0000	8886.0000

Bold values indicate the best results.

Inspecting Table 13 shows that BHLSO could be competitive and superior to all algorithms for all instances. To confirm the superiority of the proposed algorithm, *i.e*, BHLSO, Fig. 8 is given to introduce the average of the fitness values on the instances presented in Table 13. Based on this figure, BHLSO could relatively outperform the other algorithm, where BHLSO could fulfill an average value of 7956.2 as the best one, and the second-best one is BEOA with a value of 7952.1.

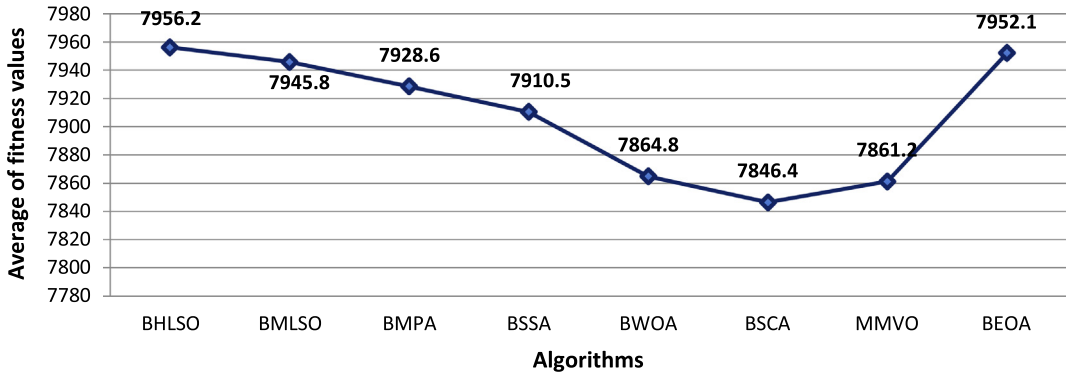


Fig. 8 Comparison in terms of the average of the fitness on the instances from WEISH12 to WEISH22.

Table 14 Comparison of the instances from weish23 and weish30.

Id		BHLSO	BMLSO	BMPA[18]	BSSA[56]	**BWOA**[53]	BSCA[55]	MMVO[32]	BEOA[16]
Weish23	**Best**	**8344.0000**	8303.0000	8263.0000	8303.0000	8249.0000	**8344.0000**	8224.0000	**8344.0000**
	Avg	**8329.2400**	8260.8800	8247.8400	8249.8800	8137.4800	8048.2800	8116.3200	8319.6000
	Worst	**8303.0000**	8245.0000	8233.0000	8213.0000	7862.0000	7534.0000	8054.0000	8250.0000
Weish24	**Best**	**10220.0000**	**10220.0000**	10167.0000	10091.0000	10118.0000	10167.0000	10118.0000	**10220.0000**
	Avg	**10220.0000**	10163.1600	10038.8400	9881.7200	9987.9600	9835.0400	10036.1600	10197.9200
	Worst	**10220.0000**	10116.0000	9917.0000	9818.0000	9818.0	9590.0000	9980.0000	10167.0000
Weish25	**Best**	**9939.0000**	**9939.0000**	9915.0000	9910.0000	9847.0000	**9939.0000**	9832.0000	**9939.0000**
	Avg	**9939.0000**	9930.4800	9861.3200	9796.7200	9756.1200	9699.1200	9796.7600	**9939.0000**
	Worst	**9939.0000**	9910.0000	9787.0000	9664.0000	9518.0000	9294.0000	9787.0000	**9939.0000**
Weish26	**Best**	**9578.0000**	9575.0000	9575.0000	9476.0000	9543.0000	9500.0000	9476.0000	9575.0000
	Avg	**9569.8000**	9493.6000	9482.7200	9452.2800	9364.6800	9407.4000	9383.9600	9555.9600
	Worst	**9516.0000**	9476.0000	9476.0000	9287.0000	9191.0000	9191.0000	9328.0000	9476.0000
Weish27	**Best**	**9781.0000**	9764.0000	9764.0000	9764.0000	9778.0000	9764.0000	9764.0000	**9781.0000**
	Avg	**9775.5600**	9743.1600	9517.0000	9589.3600	9563.0400	9348.1600	9717.0800	9764.6800
	Worst	9648.0000	**9671.0000**	9639.0000	9381.0000	9325.0000	9052.0000	9614.0000	9764.0000
Weish28	**Best**	**9454.0000**	**9454.0000**	**9454.0000**	**9454.0000**	**9454.0000**	**9454.0000**	9360.0000	**9454.0000**
	Avg	**9452.6400**	9421.3600	9371.7600	9325.1200	9204.8800	9152.9200	9142.7200	9430.7200
	Worst	**9443.0000**	9400.0000	9204.0000	9161.0000	8779.0000	8671.0000	9003.0000	9400.0000
Weish29	**Best**	**9378.0000**	9369.0000	9369.0000	9369.0000	9369.0000	9369.0000	9303.0000	9372.0000
	Avg	**9373.6800**	9367.8800	9287.3200	9274.4000	9148.7600	9014.0000	8987.2000	9369.2400
	Worst	**9369.0000**	9341.0000	9149.0000	9021.0000	8667.0000	8511.0000	8797.0000	**9369.0000**
Weish30	**Best**	**11191.0000**	11182.0000	11130.0000	11111.0000	11010.0000	11023.0000	10879.0000	11182.0000
	Avg	**11186.3200**	11166.6000	11036.5600	10849.1600	10865.1200	10693.8000	10720.0000	11172.0400
	Worst	**11165.0000**	11149.0000	10857.0000	10849.1600	10564.0000	10380.0000	10792.2400	11154.0000

Bold values indicate the best results.

Furthermore, more instances with a higher number of items and knapsack dimensions are used to validate the performance of the algorithms. After running the proposed algorithm on those instances and introducing the best, Avg, and Worst in Table 14, it was notified the superiority of the proposed algorithm in comparison to the other in terms of the investigated performance metrics in most instances. For each algorithm, the average of the fitness values introduced in Table 14 are calculated and depicted in Fig. 9. This figure shows that BHLSO is the best one with an amount of 9730.8, while BEOA is the second one with a value of 9718.9. Finally, Table 15 is presented to report the difference between the outcomes of BHLSO and the rival algorithms under the Wilcoxon rank-sum test for some MKP instances; this table demonstrates the significant difference between the outcomes of BHLSO against the rival algorithms for the vast majority of instances.

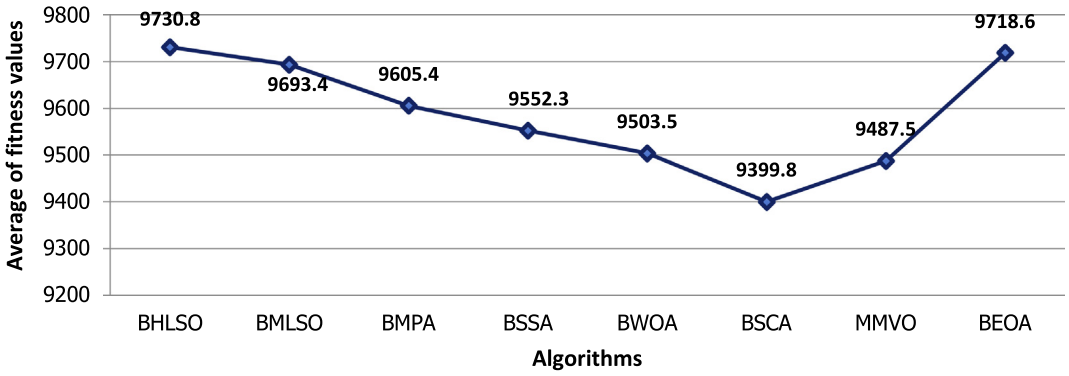


Fig. 9 Comparison in terms of the average of the fitness on the instances from WEISH23 to WEISH30.

Table 15 Significance analysis under Wilcoxon rank-sum test for some MKP instances.

Id	BMLSO	BMPA[18]	BSSA[56]	**BWOA**[53]	BSCA[55]	MMVO[32]	BEOA[16]
Weish12	**1.0047E-07**	**7.5684E-10**	**3.3136E-04**	**2.5822E-10**	**3.9032E-08**	**7.5684E-10**	**8.1299E-04**
Weish13	3.3706E-01	NaN	NaN	**4.4354E-11**	**1.1684E-03**	**5.6021E-11**	NaN
Weish14	**9.4553E-06**	**4.8209E-04**	**7.2080E-02**	**2.6695E-09**	**1.6175E-06**	**3.5624E-10**	**1.9452E-02**
Weish15	NaN	1.6143E-01	NaN	**2.3203E-11**	**2.4874E-03**	**2.2712E-11**	NaN
Weish16	**8.9362E-11**	**8.9362E-11**	**1.1395E-09**	**3.0407E-09**	**1.4008E-09**	**1.3739E-09**	**5.5491E-10**
Weish17	**2.0857E-02**	**3.6043E-10**	**9.1502E-11**	**8.8835E-11**	**6.7491E-11**	**9.6488E-11**	NaN
Weish18	**4.5387E-10**	**6.4895E-10**	**6.5341E-10**	**5.8761E-10**	**5.3312E-10**	**4.2180E-10**	**4.0822E-04**
Weish19	1.0000E + 00	**3.6157E-03**	**1.0774E-04**	**2.8912E-11**	**3.6498E-06**	**2.6958E-11**	3.3706E-01
Weish20	**9.2584E-05**	**3.7928E-09**	**3.5840E-10**	**7.3725E-11**	**1.3520E-07**	**4.4075E-11**	3.3706E-01
Weish21	3.3706E-01	**2.4542E-04**	1.6143E-01	**3.8247E-09**	**3.9469E-07**	**7.8574E-11**	NaN
Weish22	**6.9233E-11**	**4.4325E-12**	**3.4539E-11**	**1.4782E-11**	**1.1231E-10**	**4.4325E-12**	**3.3379E-10**
Weish23	**3.9455E-09**	**5.3525E-10**	**1.6497E-09**	**5.5921E-10**	**7.6380E-08**	**5.2362E-10**	3.9291E-01
Weish24	**2.3949E-10**	**9.6829E-11**	**8.4211E-11**	**9.1502E-11**	**7.9616E-11**	**8.6444E-11**	**2.3234E-04**
Weish25	**1.1387E-03**	**9.5135E-11**	**9.6942E-11**	**5.4516E-11**	**1.4338E-08**	**1.7562E-11**	NaN
Weish26	**3.1470E-09**	**2.9302E-10**	**3.8738E-10**	**1.0569E-09**	**5.2519E-10**	**4.4929E-10**	**2.4663E-05**
Weish27	**1.4688E-09**	**6.7460E-10**	**3.5848E-10**	**5.7057E-10**	**3.1029E-10**	**4.1942E-09**	**1.5991E-09**
Weish28	**5.4902E-05**	**9.8617E-10**	**3.2929E-08**	**5.1629E-07**	**3.2539E-08**	**1.2822E-10**	**8.6651E-03**
Weish29	**8.3170E-10**	**1.9015E-09**	**1.4908E-09**	**2.2418E-09**	**1.6401E-09**	**7.3636E-10**	**9.2677E-09**
Weish30	**3.9101E-08**	**8.5513E-10**	**8.5676E-10**	**7.8392E-10**	**8.5757E-10**	**7.8016E-10**	**1.6161E-07**

5. Conclusion and future work

Recently, a new physics-based *meta*-heuristic optimization algorithm known as light spectrum optimizer (LSO) has been proposed for solving continuous optimization problems. This algorithm could come true a significant success for these continuous problems, which motivates us to propose a binary variant of it, namely BLSO, for tackling both KP01 and MKP. The classical LSO was modified in this study to propose a new variant, namely MLSO, with better exploration and exploitation operators for overcoming the knapsack problems. To further promote the exploration and exploitation capability of the binary modified LSO (MBLSO), a novel method based on integrating both the swarm intelligence behaviors and SBX are proposed and integrated with the MLSO to produce a new binary variant abbreviated as BHLSO. To validate the performance of BLSO, BMLSO, and BHLSO, 45 KP01 instances and 30 MKP instances used commonly in the literature have been used in our experiments. In addition, the proposed binary variants of LSO are compared with a number of the recently-published algorithm to see their competitiveness. The empirical outcomes show the superiority of BHLSO for both KP01 and MKP in terms of final accuracy, CPU time, and convergence speed.

Future work involves applying LSO for tackling image segmentation problems, multi-objective optimization problems, parameter identification problem of photovoltaic systems and DNA fragment assembly problems,

Funding: The research is funded by Researchers Supporting Program at King Saud University, (RSPD2023R533).

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.

Acknowledgement

The authors give their appreciation to King Saud University for funding this work through the Researchers Supporting Project number (RSPD2023R533), King Saud University, Riyadh, Saudi Arabia.

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

References

[1] S. Khan et al, Solving the knapsack problem for adaptive multimedia systems, Stud. Inform. Univ. 2 (1) (2002) 157–178.
[2] E. Bas, A capital budgeting problem for preventing workplace mobbing by using analytic hierarchy process and fuzzy 0–1 bidimensional knapsack model, Expert Syst. Appl. 38 (10) (2011) 12415–12422.
[3] F. Taillandier, C. Fernandez, A. Ndiaye, Real estate property maintenance optimization based on multiobjective multidimensional knapsack problem, Comput. Aided Civ. Inf. Eng. 32 (3) (2017) 227–251.
[4] M. Engwall, A. Jerbrant, The resource allocation syndrome: the prime challenge of multi-project management?, Int J. Proj. Manag. 21 (6) (2003) 403–409.
[5] S. Müller, et al. Computation offloading in wireless multi-hop networks: Energy minimization via multi-dimensional knapsack problem. in 2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). 201IEEE.
[6] T. Karaboghossian, M. Zito, Easy knapsacks and the complexity of energy allocation problems in the smart grid, Optim. Lett. 12 (7) (2018) 1553–1568.
[7] P. Jacko, Resource capacity allocation to stochastic dynamic competitors: knapsack problem for perishable items and index-knapsack heuristic, Ann. Oper. Res. 241 (1–2) (2016) 83–107.
[8] Y. Li, Y. Tao, F. Wang, A compromised large-scale neighborhood search heuristic for capacitated air cargo loading planning, Eur. J. Oper. Res. 199 (2) (2009) 553–560.
[9] A. Koc et al, Prioritizing project selection, Eng. Econ. 54 (4) (2009) 267–297.
[10] E. Bas, Surrogate relaxation of a fuzzy multidimensional 0–1 knapsack model by surrogate constraint normalization rules and a methodology for multi-attribute project portfolio selection, Eng. Appl. Artif. Intel. 25 (5) (2012) 958–970.
[11] M. Tavana et al, A fuzzy hybrid project portfolio selection method using data envelopment analysis, TOPSIS and integer programming, Expert Syst. Appl. 42 (22) (2015) 8432–8444.
[12] M. Tavana, K. Khalili-Damghani, A.-R. Abtahi, A fuzzy multidimensional multiple-choice knapsack model for project portfolio selection using an evolutionary algorithm, Ann. Oper. Res. 206 (1) (2013) 449–483.
[13] S. Balev et al, A dynamic programming based reduction procedure for the multidimensional 0–1 knapsack problem, Eur. J. Oper. Res. 186 (1) (2008) 63–76.
[14] V. Boyer, D. El Baz, M. Elkihel, Solution of multidimensional knapsack problems via cooperation of dynamic programming and branch and bound, Eur. J. Ind. Eng. 4 (4) (2010) 434–449.
[15] Y. Vimont, S. Boussier, M. Vasquez, Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem, J. Comb. Optim. 15 (2) (2008) 165–178.
[16] M. Abdel-Basset, R. Mohamed, S. Mirjalili, A Binary Equilibrium Optimization Algorithm for 0–1 Knapsack Problems, Comput. Ind. Eng. (2020) 106946.
[17] S. Zhang, Y. Zhou, Q.J.E.S. Luo, A complex-valued encoding satin bowerbird optimization algorithm for global optimization. 12 (1) (2021) 191–205.
[18] M. Abdel-Basset et al, New Binary Marine Predators Optimization Algorithm for 0–1 Knapsack Problems, Comput. Ind. Eng. (2020) 106949.
[19] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.
[20] Y. He et al, An efficient binary differential evolution algorithm for the multidimensional knapsack problem, Eng. Comput. (2019) 1–17.
[21] D.-E.-S.-B. Mohammed, A binary multi-verse optimizer for 0–1 multidimensional knapsack problems with application in interactive multimedia systems, Elsevier (2019) 132.
[22] P. Wang et al, Complex-valued encoding metaheuristic optimization algorithm: A comprehensive survey. 407 (2020) 313–342.
[23] C. Han, G. Zhou, Y.J.I.A. Zhou, Binary symbiotic organism search algorithm for feature selection and analysis. 7 (2019) 166833–166859.
[24] Y. Gao et al, Quantum-Inspired Wolf Pack Algorithm to Solve the 0–1 Knapsack Problem, Math. Probl. Eng. 2018 (2018).
[25] Y. Feng et al, Opposition-based learning monarch butterfly optimization with Gaussian perturbation for large-scale 0–1 knapsack problem, Comput. Electr. Eng. 67 (2018) 454–468.
[26] Y. Feng et al, Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation, Memetic Computing 10 (2) (2018) 135–150.
[27] M. Abdel-Basset, D. El-Shahat, I. El-Henawy, Solving 0–1 knapsack problem by binary flower pollination algorithm, Neural Comput. & Applic. 31 (9) (2019) 5477–5495.
[28] X. Lai et al, Diversity-preserving quantum particle swarm optimization for the multidimensional knapsack problem, Expert Syst. Appl. (2020) 113310.
[29] A.C. Adamuthe, V.N. Sale, S.U. Mane, Solving single and multi-objective 01 Knapsack Problem using Harmony Search Algorithm, J. Sci. Res. 64 (1) (2020).
[30] J. García et al, A db-Scan Hybrid Algorithm: An Application to the Multidimensional Knapsack Problem, Mathematics 8 (4) (2020) 507.
[31] Z. Beheshti, A novel x-shaped binary particle swarm optimization, Soft. Comput. (2020) 1–30.
[32] M. Abdel-Basset et al, A binary multi-verse optimizer for 0–1 multidimensional knapsack problems with application in interactive multimedia systems, Comput. Ind. Eng. 132 (2019) 187–206.
[33] A. Baykasoğlu , F.B.J.E.S.w.A. Ozsoydan, An improved firefly algorithm for solving dynamic multidimensional knapsack problems. 2014. 41(8): p. 3712-3725.
[34] J. Branke, M. Orbayı, Ş. Uyar, The role of representations in dynamic knapsack problems, in: Workshops on Applications of Evolutionary Computation. 2006. Springer.
[35] F.B. Ozsoydan, A.J.F.G.C.S. Baykasoglu, A swarm intelligence-based algorithm for the set-union knapsack problem, 2019, 93, p. 560-569
[36] Y. Zhou, et al., A complex-valued encoding wind driven optimization for the 0-1 knapsack problem, 2017, 46(3), p. 684-702
[37] Y. Zhou, L. Li, M.J.N.P.L. Ma, A complex-valued encoding bat algorithm for solving 0–1 knapsack problem. 44 (2) (2016) 407–430.
[38] A. Baykasoğlu, F.B.J.I.S. Ozsoydan, Evolutionary and population-based methods versus constructive search strategies in dynamic combinatorial optimization. 420 (2017) 159–183.
[39] İ. Gölcük, F.B.J.E.S.w.A. Ozsoydan, Quantum particles-enhanced multiple Harris Hawks swarms for dynamic optimization problems. 2021, 167, p. 114202.
[40] B.C. Dean, M.X. Goemans, J.J.M.o.O.R. Vondrák, Approximating the stochastic knapsack problem: The benefit of adaptivity. 2008. 33(4): p. 945-964.

[41] A. Baykasoğlu, F.B. Ozsoydan, M.E.J.O.R. Senol, Weighted superposition attraction algorithm for binary optimization problems. 2020. 20(4): p. 2555-2581
[42] J.C. Hartman, T.C.J.C. Perry, and Cybernetics, Approximating the solution of a dynamic, stochastic multiple knapsack problem. 2006. 35(3): p. 535-550.
[43] F.B.J.C. Ozsoydan, I. Engineering, Artificial search agents with cognitive intelligence for binary optimization problems. 136 (2019) 18–30.
[44] F.B. Ozsoydan, Effects of dominant wolves in grey wolf optimization algorithm. 83 (2019) 105658.
[45] M. Abdel-Basset et al, Light Spectrum Optimizer: A Novel Physics-Inspired Metaheuristic Optimization Algorithm. 10 (19) (2022) 3466.
[46] S. Tsutsui, M. Yamamura, T. Higuchi, Multi-parent recombination with simplex crossover in real coded genetic algorithms, in Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1. 1999.
[47] I. Ono, H. Kita, S. Kobayashi, A real-coded genetic algorithm using the unimodal normal distribution crossover, in: Advances in Evolutionary Computing, Springer, 2003, pp. 213–237.
[48] 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.
[49] Y. Zhou, X. Chen, G. Zhou, An improved monkey algorithm for a 0–1 knapsack problem, Appl. Soft Comput. 38 (2016) 817–830.
[50] H. Wu, Y. Zhou, Q. Luo, Hybrid symbiotic organisms search algorithm for solving 0–1 knapsack problem, Int. J. Bio-Inspired Comput. 12 (1) (2018) 23–53.
[51] J. Cao et al, A modified artificial bee colony approach for the 0–1 knapsack problem, Appl. Intell. 48 (6) (2018) 1582–1595.
[52] P.C. Chu, J.E. Beasley, A genetic algorithm for the multidimensional knapsack problem, J. Heuristics 4 (1) (1998) 63–86.
[53] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[54] A.A. Heidari et al, Harris hawks optimization: Algorithm and applications, Futur. Gener. Comput. Syst. 97 (2019) 849–872.
[55] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120–133.
[56] S. Mirjalili et al, Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.
[57] M. Abdel-Basset, et al., IEGA: an improved elitism-based genetic algorithm for task scheduling problem in fog computing. 2021. 36(9): p. 4592-4631
[58] R. Storn, K.J.J.o.g.o. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. 1997. 11(4): p. 341-359.
[59] P. Kaelo, M.J.E.j.o.o.r. Ali, A numerical study of some modified differential evolution algorithms. 2006. 169(3): p. 1176-1184.