Solving knapsack problems using a binary gaining sharing knowledge-based optimization algorithm

Prachi Agrawal¹ · Talari Ganesh¹ · Ali Wagdy Mohamed^{2,3}

Received: 13 September 2020 / Accepted: 20 March 2021 / Published online: 4 April 2021
© The Author(s) 2021

Abstract

This article proposes a novel binary version of recently developed Gaining Sharing knowledge-based optimization algorithm (GSK) to solve binary optimization problems. GSK algorithm is based on the concept of how humans acquire and share knowledge during their life span. A binary version of GSK named novel binary Gaining Sharing knowledge-based optimization algorithm (NBGSK) depends on mainly two binary stages: binary junior gaining sharing stage and binary senior gaining sharing stage with knowledge factor 1. These two stages enable NBGSK for exploring and exploitation of the search space efficiently and effectively to solve problems in binary space. Moreover, to enhance the performance of NBGSK and prevent the solutions from trapping into local optima, NBGSK with population size reduction (PR-NBGSK) is introduced. It decreases the population size gradually with a linear function. The proposed NBGSK and PR-NBGSK applied to set of knapsack instances with small and large dimensions, which shows that NBGSK and PR-NBGSK are more efficient and effective in terms of convergence, robustness, and accuracy.

Keywords Gaining sharing knowledge-based optimization algorithm · 0–1 Knapsack problem · Population reduction technique · Metaheuristic algorithms · Binary variables

Introduction

In combinatorial optimization, the knapsack problem is one of the most challenging and NP-hard problems. It has been studied in the last few years and come up with various real-world applications such as resource allocation, selection of portfolios, assignment, and reliability problems [25]. Let the number of items d with profits p_k ; ($k = 1, 2, \dots, d$) and weights w_k ; ($k = 1, 2, \dots, d$) are packed in a knapsack of

maximum capacity $w_{\max}$. x_k ; ($k = 1, 2, \dots, d$) represents the selection of item whether it is selected in a knapsack or not. Therefore, x_k takes only two values 0 and 1; 0 means that k th item is not selected, and 1 represents the selection in a knapsack, and the selection of each item is at most once. The mathematical model of 0–1 knapsack problem (0-1KP) is given as:

Inputs: Number of items d

$p_k : [k] \rightarrow \mathbb{N}$, $w_k : [k] \rightarrow \mathbb{N}$, $w_{\max} \in \mathbb{N}$:

$$\text{Objective function: } \max f = \sum_{k=1}^d p_k x_k \quad (1)$$

$$\text{Constraints: } \sum_{k=1}^d w_k x_k \leq w_{\max} \quad (2)$$

$$x_k = 0 \text{ or } 1; \quad k = 1, 2, \dots, d. \quad (3)$$

The main aim of the knapsack problem is to maximize the profits of items, such that the total weights of selected items must be less than or equal to the capacity of the knapsack. In reality, 0-1KP are non-differentiable, discontinuous, and high-dimensional problems; therefore, it is not possible to apply the classical approach such as branch and bound

✉ Ali Wagdy Mohamed
aliwagdy@gmail.com

Prachi Agrawal
Prachiagrawal202@gmail.com

Talari Ganesh
ganimsc2007@gmail.com

¹ Department of Mathematics and Scientific Computing,
National Institute of Technology Hamirpur, Hamirpur
177005, Himachal Pradesh, India

² Operations Research Department, Faculty of Graduate Studies
for Statistical Research, Cairo University, Giza 12613, Egypt

³ Wireless Intelligent Networks Center (WINC), School of
Engineering and Applied Sciences, Nile University, Giza,
Egypt

method [17] and dynamic programming [7]. For example, due to consideration of high dimensions for 0-1KP, choosing the global optimal solution from the exhaustive set of feasible solution is not realistic. Hence, to overcome these difficulties, numerous metaheuristic algorithms have been developed and studied in the last 3 decades. In metaheuristic algorithms, there is no need for continuity and differentiability of the objective functions. There are various algorithms that have been developed to solve the complex optimization problems such as genetic algorithm, ant colony optimization, differential evolution, particle swarm optimization algorithm, etc., and applied to various real-world problems such as two-agent multi-facility customer order scheduling [23], earthquake casualty prediction [11], task scheduling [39], flow shop scheduling problem [15,16], etc.

Many metaheuristic algorithms have been proposed to solve 0-1KP in recent years. Shi [33] modified ant colony optimization algorithm to solve the classical 0-1KP, whereas Lin [22] used a genetic algorithm to obtain the solutions of knapsack problem with uncertain weights. Li and Li [21] proposed a binary particle swarm optimization algorithm with a multi-mutation to tackle the knapsack problem. A schema guiding evolutionary algorithm has been proposed for the knapsack problem by Liu and Liu [24]. Truong et al. [34] profounded chemical reaction optimization algorithm for solving 0-1KP with the greedy strategy to repair the infeasible solutions. Researchers pay great attention to develop the binary and discrete versions of various algorithms such as binary artificial fish swarm algorithm [3], adaptive binary Harmony Search algorithm [36], binary monkey algorithm [40], binary multi-verse optimizer [1], and discrete shuffled frog leaping algorithm [5] for solving the 0-1KP. Many algorithms have been developed to solve only knapsack problems which solve only low-dimensional knapsack problems. The real-world issues consider very high dimensions, and it is challenging to handle high-dimensional problems. Zou et al. [42] proposed a novel global harmony search algorithm with genetic mutation for obtaining the solution of knapsack problems. Moosavian [32] proposed Soccer league competition algorithm to tackle with the high dimensions of knapsack problems.

Out of these metaheuristic algorithms, gaining sharing knowledge based optimization algorithm (GSK), is recently developed human-based algorithm over continuous space [30]. GSK is based on the ideology of how human acquires and shares knowledge during their life-time. It depends on the two essential stages: junior or beginners gaining and sharing stage and senior or experts gaining and sharing stage. To enhance their skills, persons gain knowledge from their networks and share the acquired knowledge with the other persons in both stages.

GSK algorithm is applied to the continuous optimization problems, and the obtained results prove its robustness,

```

for  $t=1:NP$ 
  for  $k=1:d$ 
    if  $\text{rand} \leq k_r$  (knowledge ratio)
      if  $f(x_t) > f(x_R)$ 
         $x_{tk}^{new} = (x_t + k_f * ((x_{t-1} - x_{t+1}) + (x_R - x_t)))$ 
      else
         $x_{tk}^{new} = (x_t + k_f * ((x_{t-1} - x_{t+1}) + (x_t - x_R)))$ 
      end
    else  $x_{tk}^{new} = x_{tk}^{old}$ 
    end
  end

```

Fig. 1 Pseudocode for junior gaining sharing knowledge stage

```

for  $t=1:NP$ 
  for  $k=1:d$ 
    if  $\text{rand} \leq k_r$  (knowledge ratio)
      if  $f(x_t) > f(x_{middle})$ 
         $x_{tk}^{new} = (x_t + k_f * ((x_{pbest} - x_{pworst}) + (x_{middle} - x_t)))$ 
      else
         $x_{tk}^{new} = (x_t + k_f * ((x_{pbest} - x_{pworst}) + (x_t - x_{middle})))$ 
      end
    else  $x_{tk}^{new} = x_{tk}^{old}$ 
    end
  end

```

Fig. 2 Pseudocode for senior gaining sharing knowledge stage

efficiency, and ability to find optimal solutions for the problems. The GSK algorithm has shown its significant capability in solving two different sets over continuous space, the CEC2017 benchmark suite (30 unconstrained problems with dimensions 10, 30, 50, and 100) [2] in addition to CEC2011 benchmark suite (22 constrained problems with dimensions from 1 to 140) [12]. Moreover, it outperforms the most famous 10 metaheuristics such as differential evolution, particle swarm optimization, genetic algorithm, grey wolf optimizer, teaching learning based optimization, ant colony optimization, stochastic fractal search, animal migration optimization, and many others which reflects in turn its outstanding performance compared with other metaheuristics. The manuscript proposes a novel binary gaining sharing knowledge-based optimization algorithm (NBGSK) to solve the binary optimization problems. NBGSK algorithm has two requisites: binary junior or beginners gaining and sharing stage, and binary senior or experts gaining and sharing stage. These two stages enable NBGSK to explore the search space and intensify the exploitation tendency efficiently and effectively. The proposed NBGSK algorithm is applied to the NP-hard 0-1KP to check the performance of NBGSK and, the obtained solutions are compared with existing results from literature [32].

Fig. 3 Flowchart of GSK algorithm

As the population size is one of the most important parameters of any metaheuristic algorithm; therefore, choosing the appropriate size of the population is a very critical task. A large number of population size extend the diversity, but use more numbers of function evaluations. On the other hand, by considering a small number of population size, it may trap into local optima. From the literature, there are following observation to choose the population size:

- The population size may be different for every problem [9].
- It can be based on the dimension of the problems [31].
- It may be varied or fixed throughout the optimization process according to the problems [6,18].

Mohamed et al. [29] proposed adaptive guided differential evolution algorithm with population size reduction technique which reduces the population-size gradually. Furthermore,

Table 1 Results of binary junior gaining and sharing stage of Case 1 with $k_f = 1$

	x_{t-1}	x_{t+1}	x_R	Results	Modified results
Subcase (a)	0	0	0	0	0
	0	0	1	1	1
	1	1	0	0	0
	1	1	1	1	1
Subcase (b)	1	0	0	1	1
	1	0	1	2	1
	0	1	0	−1	0
	0	1	1	0	0

GSK algorithm is a population-based optimization algorithm, and its mechanism depends on the size of the population. Similarly, to enhance the performance of NBGSK algorithm, the linear population size reduction mechanism

is applied, that decreases the population size linearly, which is marked as PR-NBGSK. To check the performance of PR-NBGSK, it is employed on the 0-1KP with small and large dimensions and compared the results with NBGSK, binary bat algorithm [28], and different binary versions of particle swarm optimization algorithm [27,35].

The organization of the paper is as: the second section describes the GSK algorithms; third section describes the proposed novel binary GSK algorithm. The population reduction scheme is elaborated in fourth section and the numerical experiments and their comparison are given in fifth section that is followed by final section which contains the concluding remarks.

GSK algorithm

A constrained optimization problem is formulated as:

$$\min f(X); \quad X = [x_1, x_2, \dots, x_d]$$

s.t.

$$g_t(X) \leq 0; \quad t = 1, 2, \dots, m$$

$$X \in [L_k, U_k]; \quad k = 1, 2, \dots, d,$$

where, f denotes the objective function; $X = [x_1, x_2, \dots, x_d]$ are the decision variables; $g_t(X)$ are the inequality constraints; and L_k, U_k are the lower and upper bounds of decision variables, respectively, and d represents the dimension of individuals. If the problem is in maximization form, then consider minimization = – maximization.

In the recent years, a novel human-based optimization algorithm, Gaining sharing knowledge-based optimization algorithm (GSK) [30], has been developed. It follows the concept of gaining and sharing knowledge throughout human life-time. GSK mainly relies on the two important stages:

1. Junior gaining and sharing stage (early–middle stage)
2. Senior gaining and sharing stage (middle–later stage).

In the early middle stage or junior gaining and sharing stage, it is not possible to acquire knowledge from social media or friends. An individual gains knowledge from their known persons such as family members, relatives, or neighbours. Due to lack of experience, these people want to share their thoughts or gained knowledge with other people which may or may not be from their networks, and they do not have much experience to differentiate others in good or bad category.

Contrarily, in the middle later years stage or senior gaining sharing stage, individuals gain knowledge from their large networks such as social media friends and colleagues. These people have much experience or great ability to categorize

Table 2 Results of binary junior gaining and sharing stage of Case 2 with $k_f = 1$

	x_{t-1}	x_t	x_{t+1}	x_R	Results	Modified results
Subcase (c)	1	1	0	0	3	1
	1	0	0	0	1	1
	0	1	1	1	0	0
	0	0	1	1	– 2	0
Subcase (d)	0	0	0	0	0	0
	0	1	0	0	2	1
	0	0	1	0	– 1	0
	0	0	0	1	– 1	0
	1	0	1	0	0	0
	1	0	0	1	0	0
	0	1	1	0	1	1
	0	1	0	1	1	1
	1	1	1	0	2	1
	1	0	1	1	– 1	0
	1	1	0	1	2	1
	1	1	1	1	1	1

```

1 Start
2 Initialize the value of parameters ( $d, MaxNFE, NP, k_r, p$ )
3  $K = \text{rand}$ ;
4 Create discrete binary population using equation (7)
5 Evaluate  $f(x_t)$ .
6 while  $NFE < MaxNFE$ 
7  $d_{junior} = d * \left(1 - \frac{NFE}{MaxNFE}\right)^K$ ;  $d_{senior} = d - d_{junior}$ ;
8 Apply Binary Junior gaining sharing stage
9 Apply Binary Senior gaining sharing stage
10 Update the population
11 Select the global best solution
12 End
13 End

```

Fig. 4 Pseudocode for NBGSK

people into good or bad classes. Thus, they can share their knowledge or skills with the most suitable persons, so that they can enhance their skills. The dimensions of junior and senior stages will be calculated and it will depend on the knowledge factor. The process mentioned above of GSK can be formulated mathematically in the following steps:

Step 1 In the first step, the number of persons is assumed (number of population size NP). Let x_t ($t = 1, 2, \dots, NP$) be the individuals of a population. $x_{tk} = (x_{t1}, x_{t2}, \dots, x_{td})$, where d is branch of knowledge assigned to an individual. and f_t ($t = 1, 2, \dots, NP$) is the corresponding objective function values.

To obtain a starting solution for the optimization problem, the initial population must be obtained. The initial population is

Table 3 Results of binary senior gaining and sharing stage of Case 1 with $k_f = 1$

	x_{pbest}	x_{pworst}	x_{middle}	Results	Modified results
Subcase (a)	0	0	0	0	0
	0	0	1	1	1
	1	1	0	0	0
	1	1	1	1	1
Subcase (b)	1	0	0	1	1
	1	0	1	2	1
	0	1	0	− 1	0
	0	1	1	0	0

Table 4 Results of binary senior gaining and sharing stage of Case 2 with $k_f = 1$

	x_{pbest}	x_t	x_{pworst}	x_{middle}	Results	Modified results
Subcase (c)	1	1	0	0	3	1
	1	0	0	0	1	1
	0	1	1	1	0	0
	0	0	1	1	− 2	0
Subcase (d)	0	0	0	0	0	0
	0	1	0	0	2	1
	0	0	1	0	− 1	0
	0	0	0	1	− 1	0
	1	0	1	0	0	0
	1	0	0	1	0	0
	0	1	1	0	1	1
	0	1	0	1	1	1
	1	1	1	0	2	1
	1	0	1	1	− 1	0
	1	1	0	1	2	1
	1	1	1	1	1	1

1 Start
2 Initialize parameters ($d, MaxNFE, k_r, p, NP_{min}, NP_{max}$)
3 $K = rand$;
4 Create discrete binary population using equation (7)
5 Evaluate $f(x_t)$.
6 while $NFE < MaxNFE$
7 $d_{junior} = d * \left(1 - \frac{NFE}{MaxNFE}\right)^K$; $d_{senior} = d - d_{junior}$;
8 Apply Binary Junior gaining sharing stage
9 Apply Binary Senior gaining sharing stage
10 Update the population;
11 Reduce the population size using equation (14)
12 Select the global best solution
13 End
14 End

Fig. 5 Pseudocode for PR-NBGSK

created randomly within the boundary constraints as:

$$x_{tk}^0 = L_k + rand_k * (U_k - L_k), \quad (4)$$

where $rand_k$ denotes uniformly distributed random number in the range 0 and 1.

Step 2 At first, the dimensions of junior and senior stage should be computed through the following formula:

$$d_{junior} = d * \left(\frac{Gen^{max} - G}{Gen^{max}} \right)^K \quad (5)$$

$$d_{senior} = d - d_{junior}, \quad (6)$$

where $K (> 0)$ denotes the knowledge rate, which governs the experience rate. d_{junior} and d_{senior} represent the dimension for the junior and senior stages, respectively. Gen^{max} is the maximum number of generations, and G denotes the generation number.

Step 3 Junior gaining sharing knowledge stage: During which the early aged people gain knowledge from their small networks and share their views with the other people who may or may not belong to their group. Thus, individuals are updated through as:

1. According to objective function values, the individuals are arranged in ascending order as:

$$x_{\text{best}}, \dots, x_{t-1}, x_t, x_{t+1}, \dots, x_{\text{worst}}.$$

2. For every x_t ($t = 1, 2, \dots, \text{NP}$), select the nearest best (x_{t-1}) and worst x_{t+1} to gain the knowledge, and also select randomly (x_R) to share the knowledge. Therefore, to update the individuals, the pseudocode is presented in Fig. 1 in which $k_f (> 0)$ is the knowledge factor.

Step 4 Senior gaining sharing knowledge stage: This stage comprises the impact and effect of other people (good or bad) on the individual. The updation of the individual can be determined as follows:

1. The individuals are classified into three categories (best, middle, and worst) after sorting individuals into ascending order (based on the objective function values).

best individual = $100p\%$ (x_{pbest}), middle individual = $d - 2 \cdot 100p\%$ (x_{middle}), worst individual = $100p\%$ (x_{pworst}).

2. For every individual x_t , choose two random vectors of the top and bottom $100p\%$ individual for gaining part and the third one (middle individual) is chosen for the sharing part, where $p \in [0, 1]$ is the percentage of best and worst classes. Therefore, the new individual is updated through the following pseudocode presented in Fig. 2.

The flowchart of GSK algorithm is shown in Fig. 3.

Proposed novel binary GSK algorithm (NBGSK)

To solve problems in binary space, a novel binary gaining sharing knowledge-based optimization algorithm (NBGSK) is suggested. In NBGSK, the new initialization, dimensions of stages, and the working mechanism of both stages (junior and senior gaining sharing stages) are introduced over binary space, and the remaining algorithms remain the same as the previous one. The working mechanism of NBGSK is presented in the following subsections:

Binary initialization

The initial population is obtained in GSK using Eq. (4) and it must be updated using the following equation for binary population:

$$x_{tk}^0 = \text{round}(\text{rand}(0, 1)), \quad (7)$$

where the round operator is used to convert the decimal number into the nearest binary number.

Evaluate the dimensions of stages

Before proceeding further, the dimensions of junior (d_{junior}) and senior (d_{senior}) stage should be computed using number of function evaluation (NFE) as:

$$d_{\text{junior}} = d \times \left(1 - \frac{\text{NFE}}{\text{MaxNFE}}\right)^K \quad (8)$$

$$d_{\text{senior}} = d - d_{\text{junior}}, \quad (9)$$

where $K (> 0)$ denotes the knowledge rate and it is randomly generated and MaxNFE denotes the maximum number of function evaluations.

Binary junior gaining and sharing step

A binary junior gaining and sharing step is based on the original GSK with $k_f = 1$. The individuals are updated in original GSK using the pseudo code (Fig. 1) which contains two cases. These two cases are defined for binary stage as follows:

Case 1 When $f(x_R) < f(x_t)$: there are three different vectors (x_{t-1} , x_{t+1} , x_R), which can take only two values (0 and 1). Therefore, total 2^3 combinations are possible, which are listed in Table 1. Furthermore, these eight combinations can be categorized into two different subcases [(a) and (b)] and each subcase has four combinations. The results of every possible combinations are presented in Table 1.

Subcase (a) If x_{t-1} is equal to x_{t+1} , the result is equal to x_R .
Subcase (b) When x_{t-1} is not equal to x_{t+1} , then the result is same as x_{t-1} by taking -1 as 0 and 2 as 1.

The mathematical formulation of Case 1 is as follows:

$$x_{tk}^{\text{new}} = \begin{cases} x_R; & \text{if } x_{t-1} = x_{t+1} \\ x_{t-1}; & \text{if } x_{t-1} \neq x_{t+1}. \end{cases} \quad (10)$$

Case 2 When $f(x_R) \geq f(x_t)$: There are four different vectors (x_{t-1} , x_t , x_{t+1} , x_R), which consider only two values (0 and 1). Thus, there are total 2^4 combinations possible that are presented in Table 2. Moreover, the 16 combinations can be divided into two subcases [(c) and (d)] in which (c) and (d) have 4 and 12 combinations, respectively.

Subcase (c) If x_{t-1} is not equal to x_{t+1} , but x_{t+1} is equal to x_R , the result is equal to x_{t-1} .

Subcase (d) If any of the conditions arise $x_{t-1} = x_{t+1} \neq x_R$ or $x_{t-1} \neq x_{t+1} \neq x_R$ or $x_{t-1} = x_{t+1} = x_R$, the result is equal to x_t by considering -1 and -2 as 0, and 2 and 3 as 1.

The mathematical formulation of Case 2 is as:

$$x_{tk}^{\text{new}} = \begin{cases} x_{t-1}; & \text{if } x_{t-1} \neq x_{t+1} = x_R \\ x_{t-1}; & \text{Otherwise.} \end{cases} \quad (11)$$

Binary senior gaining and sharing stage

The working mechanism of binary senior gaining and sharing stage is same as the binary junior gaining and sharing stage with value of $k_f = 1$. The individuals are updated in original senior gaining sharing stage using pseudocode (Fig. 2), which contains two cases. The two cases further modified for binary senior gaining sharing stage in the following manner:

Case 1 When $f(x_{\text{middle}}) < f(x_t)$: it contains three different vectors (x_{pbest} , x_{middle} , x_{pworst}), and they can assume only binary values (0 and 1), and thus, total eight combinations are possible to update the individuals. These total eight combinations can be classified into two subcases [(a) and (b)] and each subcase contains only four different combinations. Table 3 represents the obtained results of this case.

Subcase (a) If x_{pbest} is equal to x_{pworst} , the result is equal to x_{middle} .

Subcase (b) On the other hand, if x_{pbest} is not equal to x_{pworst} , the results are equal to x_{pbest} with assuming -1 or 2 equivalent to their nearest binary value (0 and 1, respectively).

Case 1 can be mathematically formulated in the following way:

$$x_{tk}^{\text{new}} = \begin{cases} x_{\text{middle}}; & \text{if } x_{\text{pbest}} = x_{\text{pworst}} \\ x_{\text{pbest}}; & \text{if } x_{\text{pbest}} \neq x_{\text{pworst}}. \end{cases} \quad (12)$$

Case 2 When $f(x_{\text{middle}}) \geq f(x_t)$: it consists of four different binary vectors (x_{pbest} , x_{middle} , x_{pworst} , x_t), and with the values of each vector, total 16 combination are presented. The 16 combinations are also divided into two subcases [(c) and (d)]. The subcases (c) and (d) further contains 4 and 12 combinations, respectively. The subcases are explained in detail in Table 4.

Subcase (c) When x_{pbest} is not equal to x_{pworst} , and x_{pworst} is equal to x_{middle} , then the obtained results are equal to x_{pbest} .

Subcase (d) If any case arises other than (c), then the obtained results is equal to x_t by taking -2 and -1 as 0 and 2 and 3 as 1.

The mathematical formulation of Case 2 is given as:

$$x_{tk}^{\text{new}} = \begin{cases} x_{\text{pbest}}; & \text{if } x_{\text{pbest}} \neq x_{\text{pworst}} = x_{\text{middle}} \\ x_t; & \text{Otherwise.} \end{cases} \quad (13)$$

The pseudocode for NBGSK is shown in Fig. 4.

Population reduction on NBGSK (PR-NBGSK)

As the population size is one of the most important parameters of optimization algorithm, it may not be fixed throughout the optimization process. For exploration of the solutions of optimization problem, first, the number of population size must be large, but to rectify the quality of solutions and enhance the performance of algorithm, decrement in the population size is required.

Mohamed et al. [29] used non-linear population reduction formula for differential evolution algorithm to solve the global numerical optimization problem. Based on the formula, we used the following framework to reduce the population size gradually:

$$NP_{G+1} = \text{round} \left[(NP_{\min} - NP_{\max}) * \left(\frac{NFE}{\text{Max NFE}} \right) + NP_{\max} \right], \quad (14)$$

where NP_{G+1} denotes the modified (new) population size in next generation, $NP_{\min}$ and $NP_{\max}$ are the minimum and maximum population size, respectively, NFE is current number of function evaluation, and Max NFE is the assumed maximum number of function evaluations. Taking into consideration $NP_{\min}$ is assumed as 12, we need at least two elements in best and worst partitions. The main advantage to apply population reduction technique to NBGSK is to discard the infeasible or worst solutions from the initial phase of the optimization process without influencing the exploration capability. In the later stage, it emphasizes the exploitation tendency by deleting the worst solutions from the search space.

Note: in this study, population size reduction technique is combined with proposed NBGSK, which is named as PR-NBGSK and the pseudocode for PR-NBGSK is drawn in Fig. 5.

Table 5 Numerical values in PR-NBGSK and NBGSK

Parameters	Considered values
$NP_{\min}$	12
$NP_{\max}$	200
k_f	1
k_r	0.9
p	0.1
δ	10^2
λ	-10^2

Numerical experiments and comparisons

To investigate the performance of proposed algorithms PR-NBGSK and NBGSK, 0-1KP are considered. The first set consists of 10 small scale problems which are taken from the literature [32] and the second one composed with 10 large scale problems.

First, to solve the constrained optimization problem, different types of constraint handling techniques are used [10,26]. Deb introduced an efficient constraint handling technique which is based on the feasibility rules [13]. Most commonly used approach to handle the constraints is penalty function method, in which the infeasible solutions are punished with some penalty to violate the constraints. Bahreininejad [4] introduced ALM for the water cycle algorithm and solved the real-world problems. In ALM, a constrained optimization problem is converted into an unconstrained optimization problem with some penalty to the original objective function. The original optimization problem is transformed into the following unconstrained

optimization problem:

$$\max = f(X) + \delta \sum_{t=1}^m \{g_t(X)\}^2 - \lambda \sum_{t=1}^m \{g_t(X)\}, \quad (15)$$

where $f(X)$ is the original objective function given in the problem, δ is quadratic penalty parameter, $\sum_{t=1}^m \{g_t(X)\}^2$ represents the quadratic penalty term, and λ is the Lagrange multiplier.

The ALM is similar to the penalty approach method in which the penalty parameter is chosen as large as possible. In ALM, δ and λ are chosen in such a way that λ can remain small to maintain the strategic distance from ill condition. The advantage of ALM is that it decreases the chances of ill conditioning that happened in the penalty approach method.

After applying the ALM to the constrained optimization problems, the problems are solved and compared with binary bat algorithm [28], V-shape transfer function used in PSO (VPSO) [27], S-shaped transfer function used in PSO (SPSO) [27], probability binary PSO (BPSO) [35], and the algorithms

Table 6 Data for small-scale problems $F_1 - F_{10}$

Problem	Dim (d)	Profits (p), weights (w), Capacity($w_{\max}$)
F_1	10	$p = (55, 10, 47, 5, 4, 50, 8, 61, 85, 87)$, $w = (95, 4, 60, 32, 23, 72, 80, 62, 65, 46)$, $w_{\max} = 269$
F_2	20	$p = (44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63)$, $w = (92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58)$, $w_{\max} = 878$
F_3	4	$p = (9, 11, 13, 15)$, $w = (6, 5, 9, 7)$, $w_{\max} = 20$
F_4	4	$p = (6, 10, 12, 13)$, $w = (2, 4, 6, 7)$, $w_{\max} = 11$
F_5	15	$p = (0.125126, 19.330424, 58.500931, 35.029145, 82.284005, 17.410810, 71.050142, 30.399487, 9.140294, 14.731285, 98.852504, 11.908322, 0.891140, 53.166295, 60.176397)$, $w = (56.358531, 80.874050, 47.987304, 89.596240, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204, 16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575)$, $w_{\max} = 375$
F_6	10	$p = (20, 18, 17, 15, 15, 10, 5, 3, 1, 1)$ $w = (30, 25, 20, 18, 17, 11, 5, 2, 1, 1)$, $w_{\max} = 60$
F_7	7	$p = (70, 20, 39, 37, 7, 5, 10)$, $w = (31, 10, 20, 19, 4, 3, 6)$, $w_{\max} = 50$
F_8	23	$p = (981, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857)$ $w = (983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959)$, $w_{\max} = 10000$
F_9	5	$p = (33, 24, 36, 37, 12)$, $w = (15, 20, 17, 8, 31)$, $w_{\max} = 80$
F_{10}	20	$p = (91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44)$ $w = (84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92)$ $w_{\max} = 879$

Table 7 Results of small-scale 0-1KP

	Algorithms	Best	Mean	St dev.	Max NFE	SR (%)
F1	BBA	295	293.98	2.025249	10,000	62
	ABHS [32]	295	295	0	10,000	100
	VPSO	295	295	0	2200	100
	BHS [32]	295	295	1.2	10,000	78
	SPSO	295	294.58	0.882714	10000	72
	NGHS [32]	295	295	0	10,000	100
	BPSO	295	295	0	1800	100
	SLC [32]	295	295	0	2269	100
	NBGSK	295	295	0	400	100
	PR-NBGSK	295	295	0	400	100
F2	BBA	985	918.34	34.36218	10,000	0
	ABHS [32]	1024	1024	0	10,000	100
	VPSO	1024	1023.76	1.187692	10000	96
	BHS [32]	1024	1023.52	1.63	10,000	92
	SPSO	1024	1000.8	12.98193	10000	8
	NGHS [32]	1024	1024	0	10,000	100
	BPSO	1024	1023.16	2.103059	10000	86
	SLC [32]	1024	1024	0	6035	100
	NBGSK	1024	1024	0	7200	100
	PR-NBGSK	1024	1024	0	5368	100
F3	BBA	35	35	0	400	100
	ABHS [32]	35	35	0	10,000	100
	VPSO	35	35	0	400	100
	BHS [32]	35	34.86	0.98	10,000	98
	SPSO	35	35	0	400	100
	NGHS [32]	35	35	0	10,000	100
	BPSO	35	35	0	400	100
	SLC [32]	35	35	0	2042	100
	NBGSK	35	35	0	400	100
	PR-NBGSK	35	35	0	400	100
F4	BBA	23	23	0	400	100
	ABHS [32]	23	23	0	10,000	100
	VPSO	23	23	0	400	100
	BHS [32]	23	22.98	0.14	10,000	98
	SPSO	23	23	0	400	100
	NGHS [32]	23	23	0	10,000	100
	BPSO	23	23	0	400	100
	SLC [32]	23	23	0	2080	100
	NBGSK	23	23	0	400	100
	PR-NBGSK	23	23	0	400	100
F5	BBA	481.0694	433.2606	25.99674	10,000	10
	ABHS [32]	481.07	481.07	0	10,000	100
	VPSO	481.07	481.07	0	5000	100
	BHS [32]	481.07	476.5	13.28	10,000	88

Table 7 continued

	Algorithms	Best	Mean	St dev.	Max NFE	SR (%)
F6	SPSO	475.4784	433.3258	19.53025	10000	0
	NGHS [32]	481.07	481.07	0	10,000	100
	BPSO	481.07	481.07	0	4200	100
	SLC [32]	481.07	481.07	0	4319	100
	NBGSK	481.07	481.07	0	6800	100
	PR-NBGSK	481.07	481.07	0	3222	100
	BBA	52	51.88	0.328261	10000	88
	ABHS [32]	52	52	0	10,000	100
	VPSO	52	52	0	6800	100
	BHS [32]	52	51.62	0.94	10,000	82
	SPSO	52	52	0	6600	100
	NGHS [32]	52	52	0	10,000	100
	BPSO	52	52	0	1200	100
	SLC [32]	52	52	0	1919	100
	NBGSK	52	52	0	5600	100
F7	PR-NBGSK	52	52	0	400	100
	BBA	107	106.92	0.395897	10,000	96
	ABHS [32]	107	107	0	10,000	100
	VPSO	107	107	0	6000	100
	BHS [32]	107	105.64	2.68	10,000	62
	SPSO	107	107	0	2000	100
	NGHS [32]	107	107	0	10,000	100
	BPSO	107	107	0	2000	100
	SLC [32]	107	107	0	2025	100
	NBGSK	107	107	0	2800	100
	PR-NBGSK	107	107	0	400	100
F8	BBA	9762	9743.02	7.914054	10,000	0
	ABHS [32]	9767	9767	0	10,000	100
	VPSO	9767	9766.52	0.99468	10,000	76
	BHS [32]	9767	9766.8	0.85	10,000	94
	SPSO	9753	9739.08	6.58954	10,000	0
	NGHS [32]	9767	9767	0	10,000	100
	BPSO	9767	9758.9	2.815772	10,000	2
	SLC [32]	9767	9767	0	4873	100
	NBGSK	9767	9764.22	2.426806	10,000	20
	PR-NBGSK	9767	9767	0	4144	100
F9	BBA	130	130	0	400	100
	ABHS [32]	130	130	0	10,000	100
	VPSO	130	130	0	400	100
	BHS [32]	130	129.76	1.68	10,000	98
	SPSO	130	130	0	400	100
	NGHS [32]	130	130	0	10,000	100
	BPSO	130	130	0	600	100
	SLC [32]	130	130	0	1994	100
	NBGSK	130	130	0	400	100
	PR-NBGSK	130	130	0	400	100

Table 7 continued

	Algorithms	Best	Mean	St dev.	Max NFE	SR (%)
F10	BBA	962	921.54	25.61585	10,000	0
	ABHS [32]	1025	1025	0	10,000	100
	VPSO	1025	1025	0	9600	100
	BHS [32]	1025	1024.64	1.42	10,000	94
	SPSO	1025	1002.4	11.53875	10,000	6
	NGHS [32]	1025	1025	0	10,000	100
	BPSO	1025	1024.28	1.969564	10,000	88
	SLC [32]	1025	1025	0	5017	100
	NBGSK	1025	1025	0	7800	100
	PR-NBGSK	1025	1025	0	3498	100

Table 8 Average computational time taken by all optimizers for small-scale problems

Problem	BBA	VPSO	SPSO	BPSO	NBGSK	PR-NBGSK
F1	0.69	0.28	0.27	0.28	0.76	0.27
F2	1.23	0.30	0.32	0.31	0.48	0.23
F3	0.44	0.26	0.27	0.28	0.95	0.22
F4	0.46	0.26	0.28	0.28	0.81	0.25
F5	0.97	0.29	0.30	0.30	0.47	0.26
F6	0.71	0.27	0.29	0.29	0.54	0.34
F7	0.58	0.28	0.28	0.28	0.85	0.55
F8	1.90	0.31	0.47	0.46	0.50	0.27
F9	0.93	0.28	0.51	0.52	0.45	0.32
F10	1.72	0.63	0.56	0.31	0.48	0.30

run on a personal computer Inter Core™ i5 @ 2.50GHz with 4 GB RAM on MATLAB R2015a. The parameters values used in NBGSK and PR-NBGSK are given in Table 5.

Small-scale problems

This section contains low-dimensional 0-1KP and the details of every problem are presented in Table 6, in which first two columns represent the name of problem and the dimensions respectively. Profits p_k , weights w_k , and the capacity of knapsack $w_{\max}$ are given in the third column of Table 6.

These problems $F_1 - F_{10}$ are taken from the literature and were solved using different algorithms to get the optimal solution. The problems F_1 and F_2 were solved by novel global harmony search algorithm [42] and the obtained optimal objective values are 295 and 1024, respectively.

Sequential combination tree algorithm was proposed by An and Fu [8] to solve the knapsack problem F_3 and the obtained optimal solution of F_3 is 35 at (1,1,0,1). This method is applicable only for low-dimensional problems.

The knapsack problem F_4 is solved using greedy-policy based algorithm [38] and the optimal objective value is 23 at (0,1,0,1).

Fig. 6 Box Plot for NFE used in ten problems of PR-NBGSK

To solve the knapsack problem F_5 with 15 decision variables, Yoshizawa and Hashimoto [37] applied the information of search space landscape and found the optimal objective value is 481.0694.

A method developed by Fayard and Plateau [14] was applied on F_6 and obtained optimal solution as 50 at (0,0,1,0,1,1,1,1,0,0).

Fig. 7 The convergence graph for small-scale 0-1KP

Table 9 Data for large-scale 0-1KP

Problems	Dim	Capacity	Max NFE
F_{11}	100	1100	15,000
F_{12}	500	4000	20,000
F_{13}	1000	10,000	30,000
F_{14}	1200	14,000	40,000
F_{15}	1400	15,000	40,000
F_{16}	1600	18,000	50,000
F_{17}	1800	20,000	50,000
F_{18}	2000	22,000	50,000
F_{19}	2200	24,000	60,000
F_{20}	2500	26,000	60,000

The knapsack problem F_7 is solved using non-linear dimensionality reduction method by Zhao [20] and found the optimal solution as 107 and F_8 is solved by NGHS and found the optimal solution as 9767 [42].

The optimal solution found by DNA [41] algorithm of F_9 is 130 at (1,1,1,1,0).

This problem is taken from the literature and the problem is solved by NGHS [42] and found the optimal solution as 1025.

The solutions of above ten problems are obtained by PR-NBGSK and NBGSK algorithms, and to compare the results, the problems are solved by four state-of-the-art algorithms BBA, VPSO, SPSO, and BPSO.

Each algorithm performs over 50 independent runs and the obtained results are presented in Table 7 with the best, worst, average objective value, number of function evaluations, and success rate of each algorithm.

The comparison is conducted on maximum number of function evaluations (NFE) used in each algorithms and the success rate (SR) of finding the optimal solutions in 50 runs. From Table 7, it can be seen that NBGSK and PR-NBGSK both provides exact solutions for each problem ($F_1 - F_{10}$). The SR of PR-NBGSK for every problem is 100%, whereas the mentioned algorithms SPSO, BPSO, and BBA also have less than 10% SR. Moreover, PR-NBGSK used very less number of function evaluations (presented in bold text), from the Table 7, 6 out of 10 problems (F_1 , F_3 , F_4 , F_6 , F_7 , F_9) PR-NBGSK used less than 1000 number of function evaluations, whereas the other algorithms used 10,000 NFE in most of the problems. Table 8 shows the average computational time taken by all algorithms. It describes that PR-NBGSK algorithm takes least computational time as compared to other algorithms. PR-NBGSK algorithm shows less time in solving 7 problems out of 10 problems. Figure 6 shows the box plot for NFE used in solving 10 knapsack problems by PR-NBGSK, which indicates that, over 50 runs, PR-NBGSK is able to find the optimal solution without more oscilla-

tions in NFE. Figure 7 presents the convergence graph of all algorithms for each problem, which shows that PR-NBGSK converges to the optimal solution in less NFE as compared to other algorithms. Therefore, PR-NBGSK and NBGSK have fast convergence speed to get the optimal solution as compared to the other state-of-the-art algorithms.

Large-scale problems

In the previous subsection, we have considered only low-dimensional 0-1KP which seems very easy to evaluate. Therefore, this part contains large-scale 0-1KP with randomly generated data. The data for 10 knapsack problems are generated randomly with the following information [36]: profit p_k is between 50 and 100; weights w_k is random integer between 5 and 20. The capacity and dimensions of the problems with maximum number of function evaluations are displayed in Table 9.

As the dimension of problems increases, the problems become more complex. The problems $F_{11} - F_{20}$ are solved using PR-NBGSK, NBGSK, BBA, VPSO, SPSO, and BPSO, and each algorithm performs over 30 independent runs. The obtained solutions of every problems are given in Table 10 with best, worst, average objective value, and their standard deviation. From Table 10, it can be observed that PR-NBGSK acquires the overwhelming performance over the other algorithms and presented the best objective value (bold text) in all problems. Besides, it can be easily observed from Table 9 that the results provided by NBGSK are better than all results provided by compared algorithms in all problems. BBA algorithm presents the worst results among all algorithms with high standard deviation, and it can be concluded that BBA is not suitable for these high-dimensional knapsack problems.

The box plots are displayed in Fig. 8 for all algorithms which demonstrates that the best, worst, and mean solutions obtained by PR-NBGSK are much better than the solutions of other compared algorithms. It also depicts that there is no disparity among the objective values in each run. It can be obviously seen from Table 10 that the standard deviations provided by both PR-NBGSK and NBGSK algorithms are very smaller than the standard deviations provided by other compared algorithm. However, the smallest standard deviation is provided by PR-NBGSK which proves the robustness of the algorithm. While, the other algorithms have more disparity between their objective value except NBGSK algorithm. Moreover, the average computational time taken by all algorithms has been calculated for all problems. Table 11 presents that PR-NBGSK algorithm takes very less time to solve large-scale problems. It has been observed that BBA algorithm consumes lot of time and as compared to other algorithms. VPSO and BPSO algorithms present good results in case of computational time; however, PR-NBGSK algorithm performs better in most of the problems. The con-

Table 10 Results of large-scale 0-1KP

Problems	Dim	Algorithms	Best	Worst	Mean	St dev.
F_{11}	100	BBA	5893	5049	5376.767	185.3112
		VPSO	7049	6743	6895	88.31019
		SPSO	6983	6804	6881.567	47.65514
		BPSO	6586	6113	6311.7	106.134
		NBGSK	7225	7159	7196.733	20.24323
		PR-NBGSK	7227	7181	7210.767	16.85779
F_{12}	500	BBA	24785	23095	23922.9	379.4236
		VPSO	25,741	24,690	25,323.5	223.1648
		SPSO	25,374	24,914	25,076.8	120.6504
		BPSO	25,240	24,658	24,897.8	148.3524
		NBGSK	27,731	27,074	27,404.23	187.5949
		PR-NBGSK	28,916	28,547	28,743.3	81.15806
F_{13}	1000	BBA	48340	46223	47068.57	589.1186
		VPSO	53,821	50,451	51,925.5	769.4977
		SPSO	60,468	59,853	60,156.63	151.3604
		BPSO	49,459	48,012	48,633.8	354.8921
		NBGSK	63,208	62,667	62,971.03	159.8755
		PR-NBGSK	64,978	64,455	64,684.93	138.7349
F_{14}	1200	BBA	58737	55387	56787.2	800.7263
		VPSO	65,788	61,863	63,451.33	919.0207
		SPSO	82,110	79,762	80,997.6	598.7184
		BPSO	59,607	57,844	58,445.1	440.6416
		NBGSK	86,076	85,644	85,841.4	112.2011
		PR-NBGSK	86431	85,986	86,273.17	115.5216
F_{15}	1400	BBA	68145	65349	66260.7	658.8381
		VPSO	75,029	70,202	72,863.5	1052.166
		SPSO	92,283	91,027	91,541.13	310.3932
		BPSO	68,918	67,282	67,931.47	425.5978
		NBGSK	95,654	94,632	95,136.33	276.0133
		PR-NBGSK	96,759	96,001	96,466.73	163.7351
F_{16}	1600	BBA	78353	73890	75386.17	832.4437
		VPSO	86,935	82,068	84,900.47	1148.607
		SPSO	110,755	108,089	109,577.2	669.1591
		BPSO	78,377	76,431	76,994.87	416.7714
		NBGSK	113,990	113,298	113,667.6	186.2691
		PR-NBGSK	114,568	113,910	114,273.3	154.8005
F_{17}	1800	BBA	86387	83213	84738.57	938.3026
		VPSO	95,959	92,059	93,726.33	1050.456
		SPSO	121,381	119,857	120,743.1	437.2429
		BPSO	87,079	85,141	86,042.8	495.8977
		NBGSK	125,597	124,482	125,177.9	294.6612
		PR-NBGSK	126,650	125,820	126,154.1	201.3681

Table 10 continued

Problems	Dim	Algorithms	Best	Worst	Mean	St dev.
F_{18}	2000	BBA	95646	91561	93022.43	939.3055
		VPSO	103,679	97,381	101,573.4	1274.78
		SPSO	132,427	130,959	131,746.4	383.074
		BPSO	95,748	93,270	93,967.3	553.7279
		NBGSK	136,871	135,880	136,,358.8	234.6086
		PR-NBGSK	138,138	137,319	137,694.8	212.1948
F_{19}	2200	BBA	106208	100862	103032.6	1272.034
		VPSO	116,886	111,407	114,289.3	1335.173
		SPSO	145,406	144,536	144,884	226.5195
		BPSO	105,199	102,858	104,071.8	691.0127
		NBGSK	150,898	149,663	150,411.6	260.4685
		PR-NBGSK	151,885	150,779	151,,373.1	296.9905
F_{20}	2500	BBA	119077	114587	117192.4	1078.569
		VPSO	131,367	124,172	127,857.6	1437.553
		SPSO	157,408	156,465	156,792.1	180.3911
		BPSO	117,959	116,648	117,282	375.5923
		NBGSK	164,893	163,531	164,211.9	372.026
		PR-NBGSK	166,570	165,495	166,006.4	272.6385

vergence graph of all algorithms are drawn in Fig. 9 for illustrating the performance of algorithms. From the figures, it can be noticed that both PR-NBGSK and NBGSK algorithms converge to the best solution as compared to other algorithms in all problems. Although the state-of-the-art algorithms converge faster than PR-NBGSK and NBGSK, they either prematurely converge or they are stagnated at early stage of the optimization process. Thus, it can be concluded that both PR-NBGSK and NBGSK are able to balance between the two contradictory aspects exploration capability and exploitation tendency.

Statistical analysis

To investigate the solution quality and the performance of the algorithms statistically [19], two non-parametric statistical hypothesis tests are conducted: Friedman test and multi-problem Wilcoxon signed-rank test.

In the Friedman test, the final rankings are obtained for different algorithms of all problems. The null hypothesis states that There is no significant difference among the performance of all algorithms, whereas the alternative hypothesis is There is a significant difference among the performance of all algorithms. The decision is made on the obtained p value; when the obtained p value is less than or equal to the assumed significance level 0.05, the null hypothesis is being rejected.

Multi-problem Wilcoxon signed-rank test was used to check the differences between all algorithms for all problems. It considers that S^+ denotes the sum of ranks for all problems which describes the first algorithm performs better than the

second one in a row, and S^- indicates the opposite of previous one. Larger the ranks indicate the larger performance discrepancy. The null hypothesis of this test narrates that There is no significant difference between the mean results of two sample and the alternative hypothesis is There is a significant difference between the mean results of two samples.

The three signs +, −, $\approx$ are assigned to compare the performance of two algorithms and described the following: Plus (+) : The results from the first algorithm are significantly better than the second one.

Minus (−) : The results from the second algorithm are significantly worse than the second one.

Approximate ($\approx$) : There is no significant difference between the two algorithms.

The p value is used for comparison and rejection of the null hypothesis that concludes; the null hypothesis is rejected if the obtained p value is less than or equal to the assumed significance level (5%).

In the following results, the p values are shown in bold, and the test are performed in SPSS 20.00. Table 12 lists the ranks according to the Friedman test. We can see that p value computed through the Friedman test is less than 0.05. Thus, we can conclude that there is a significant difference between the performances of the algorithms. The best rank was for PR-NBGSK, SLC, ABHS, and NGHS algorithms followed by NBGSK, respectively.

Table 13 summarizes the statistical analysis results of applying multiple-problem Wilcoxon's test between PR-NBGSK and other compared algorithms for $F_1 - F_{10}$ problems. From Table 13, we can see that PR-NBGSK obtains

Fig. 8 Box plot for objective function value of large-scale 0-1KP

Table 11 Average computational time taken by all optimizers for large-scale problems

Problem	BBA	VPSO	SPSO	BPSO	NBGSK	PR-NBGSK
F_{11}	10.75	0.81	0.81	0.76	1.03	0.76
F_{12}	51.20	2.94	2.99	2.60	2.87	2.43
F_{13}	135.92	8.14	14.63	12.58	8.46	9.28
F_{14}	250.85	12.72	19.90	17.31	13.14	12.70
F_{15}	249.25	14.68	19.63	16.87	16.14	14.53
F_{16}	332.04	20.88	22.35	18.11	23.23	17.96
F_{17}	514.65	25.32	23.85	20.22	25.44	20.48
F_{18}	526.52	24.68	28.41	23.96	40.89	26.12
F_{19}	614.68	31.57	37.39	31.48	37.92	30.80
F_{20}	729.70	36.51	38.05	32.88	42.15	32.40

higher S^+ values than S^- in all the cases with exception to SLC, ABHS, and NGHS, where S^+ and S^- are zero. Precisely, we can draw the following conclusions: PR-NBGSK outperforms SPSO, BHS, and BBA significantly in all functions. Thus, according to the Wilcoxon's test at $\alpha = 0.05$, the significance difference can be observed in 3 cases out of 9, which means that PR-NBGSK is significantly better than 3 algorithms out of 9 algorithms on 10 test functions at $\alpha = 0.05$. Alternatively, to be more precise, it is obvious from Table 13 that PR-NBGSK is inferior to, equal to, superior to other algorithms in 0, 63, and 27 out of the total 90 cases. Thus, it can be concluded that the performance of PR-NBGSK is almost better than the performance of compared algorithms in 30% of all cases, and it has the same performance as other compared algorithms in 70% of all problems.

Table 14 lists the ranks according to Friedman test. We can see that p value computed through Friedman test is less than 0.05. Thus, we can conclude that there is a significant difference between the performances of the algorithms. The best rank was for PR-NBGSK followed by NBGSK, respectively.

Table 15 summarizes the statistical analysis results of applying multiple-problem Wilcoxon's test between PR-NBGSK and other compared algorithms for $F_{11} - F_{20}$ problems. From Table 15, we can see that PR-NBGSK obtains higher S^+ values than S^- in all the cases. Precisely, we can draw the following conclusions: PR-NBGSK outperforms all algorithms significantly in all problems. Thus, according to the Wilcoxon's test at $\alpha = 0.05$, the significance difference can be observed in all five cases, which means that PR-NBGSK is significantly better than the five algorithms on ten test problems at $\alpha = 0.05$. Alternatively, to be more precise, it is obvious from Table 15 that PR-NBGSK is inferior to, equal to, superior to other algorithms in 0, 0, 50 out of the total 50 cases. Thus, it can be concluded that the performance of PR-NBGSK is better than the performance of compared algorithms in 100% of all cases. Accordingly, it can be deduced from these comparisons that the superiority of

the PR-NBGSK algorithm against the compared algorithms increases as the dimensions of the problems increase.

From the above discussion and results, it can be concluded that the proposed PR-NBGSK algorithm has better searching quality, efficiency, and robustness to solve low- and high-dimensional knapsack problems. The PR-NBGSK algorithm shows its overwhelming performance for all problems and proves its superiority from state-of-the-art algorithms. Moreover, the proposed binary junior and senior phase keeps the balance between the two main components of algorithms that is exploration and exploitation abilities and the population reduction rule helps to delete the worst solutions from the search space of PR-NBGSK. Besides, PR-NBGSK is very simple and easy to understand and implement in many languages.

Conclusions

This article presents a significant step and promising approach to solve the complex optimization problems in binary space. A novel binary version of gaining sharing knowledge-based optimization algorithm (NBGSK) is proposed to solve binary combinatorial optimization problems. NBGSK uses two binary vital stages: binary junior gaining and sharing stage and binary senior gaining and sharing stage, which are derived from the original junior and senior stages, respectively. Moreover, to enhance the performance of NBGSK and to get rid of worst and infeasible solutions, population size reduction technique applied to NBGSK and a new variant of NBGSK, i.e., PR-NBGSK is introduced. The proposed algorithms are employed to larger number of instances of 0-1 knapsack problems. The obtained results demonstrates that PR-NBGSK and NBGSK perform better or equal to state-of-the-art algorithms for low-dimensional 0-1 knapsack problems. For high-dimensional problems, PR-NBGSK outperforms the other mentioned algorithms, which also proven by statistical analysis of the solutions. Finally, the

Fig. 9 The convergence graph for large-scale 0-1KP

Table 12 Results of Friedman test for all algorithms across $F_1 - F_{10}$ problems

Algorithm	Mean ranking	Rank
PR-NBGSK	6.85	1
SLC	6.85	1
ABHS	6.85	1
NGHS	6.85	1
NBGSK	6.4	2
VPSO	6.2	3
BPSO	5.35	4
SPSO	4	5
BHS	2.85	6
BBA	2.8	7
Friedman p value	0	

Table 13 Wilcoxon test against PR-NBGSK for $F_1 - F_{10}$

	Algorithms	S^+	S^-	p value	+	$\approx$	–	Dec.
PR-NBGSK vs	SLC	0	0	1	0	10	0	$\approx$
	ABHS	0	0	1	0	10	0	$\approx$
	NGHS	0	0	1	0	10	0	$\approx$
	NBGSK	1	0	0.317	1	9	0	$\approx$
	VPSO	3	0	0.18	2	8	0	$\approx$
	BPSO	6	0	0.109	3	7	0	$\approx$
	SPSO	15	0	0.043	5	5	0	+
	BHS	45	0	0.008	9	1	0	+
	BBA	28	0	0.018	7	3	0	+

Table 14 Results of Friedman test for all algorithms across $F_{11} - F_{20}$ problems

Algorithm	Mean ranking	Rank
PR-NBGSK	6	1
NBGSK	5	2
SPSO	3.8	3
VPSO	3.2	4
BPSO	2	5
BBA	1	6
Friedman p value	0	

Table 15 Wilcoxon test against PR-NBGSK for $F_{11} - F_{20}$

	Algorithms	S^+	S^-	p value	+	$\approx$	–	Dec.
PR-NBGSK vs	NBGSK	55	0	0.005	10	0	0	+
	SPSO	55	0	0.005	10	0	0	+
	VPSO	55	0	0.005	10	0	0	+
	BPSO	55	0	0.005	10	0	0	+
	BBA	55	0	0.005	10	0	0	+

convergence graphs and presented box plots show that the PR-NBGSK is superior to other competitive algorithms in terms of convergence, robustness, and ability to find the optimal solutions of 0-1 knapsack problems.

Additionally, for the future research NBGSK and PR-NBGSK algorithms can be applied to multi-dimensional knapsack problems, and also, it may be enhanced by combining novel adaptive scheme for solving real-world problems. The Matlab source code of PR-NBGSK can be downloaded from <https://sites.google.com/view/optimization-project/files>.

Acknowledgements The authors would like to acknowledge the Editors and anonymous reviewers for providing their valuable comments and suggestions.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Abdel-Basset M, El-Shahat D, Faris H, Mirjalili S (2019) A binary multi-verse optimizer for 0–1 multidimensional knapsack problems with application in interactive multimedia systems. *Comput Ind Eng* 132:187–206
2. Awad N, Ali M, Liang JJ, Qu B, Suganthan P (2016) Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization. In: *Tech Rep*
3. Azad MAK, Rocha AMA, Fernandes EM (2014) A simplified binary artificial fish swarm algorithm for 0–1 quadratic knapsack problems. *J Comput Appl Math* 259:897–904
4. Bahreininejad A (2019) Improving the performance of water cycle algorithm using augmented lagrangian method. *Adv Eng Softw* 132:55–64
5. Bhattacharjee KK, Sarmah SP (2014) Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. *Appl Soft Comput* 19:252–263
6. Brest J, Maučec MS (2011) Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Comput* 15(11):2157–2174
7. Brocorne L, Hanafi S, Mansi R (2009) A dynamic programming algorithm for the bilevel knapsack problem. *Oper Res Lett* 37(3):215–218
8. Chen A, Yongjun F (2008) On the sequential combination tree algorithm for 0–1 knapsack problem. *J Wenzhou Univ (Natural Sci)* 2008:1
9. Cheng J, Zhang G, Neri F (2013) Enhancing distributed differential evolution with multicultural migration for global numerical optimization. *Inf Sci* 247:72–93
10. Coello CAC (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Comput Methods Appl Mech Eng* 191(11–12):1245–1287
11. Cui S, Yin Y, Wang D, Li Z, Wang Y (2020) A stacking-based ensemble learning method for earthquake casualty prediction. *Appl Soft Comput* 2020:56
12. Das S, Suganthan PN (2010) Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems. In: *Jadavpur University, Nanyang Technological University, Kolkata*, pp 341–359
13. Deb K (2000) An efficient constraint handling method for genetic algorithms. *Comput Methods Appl Mech Eng* 186(2–4):311–338
14. Fayard D, Plateau G (1975) Resolution of the 0–1 knapsack problem: comparison of methods. *Math Program* 8(1):272–307
15. Fu Y, Wang H, Wang J, Pu X (2020) Multiobjective modeling and optimization for scheduling a stochastic hybrid flow shop with maximizing processing quality and minimizing total tardiness. *IEEE Syst J* 2020:65
16. Fu Y, Zhou M, Guo X, Qi L (2019) Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm. *IEEE Trans Syst Man Cybern Syst* 50(12):5037–5048
17. Fukunaga AS (2011) A branch-and-bound algorithm for hard multiple knapsack problems. *Ann Oper Res* 184(1):97–119
18. Gao WF, Yen GG, Liu SY (2014) A dual-population differential evolution with coevolution for constrained optimization. *IEEE Trans Cybern* 45(5):1108–1121
19. García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *J Heuristics* 15(6):617
20. Jian-ying Z (2007) Nonlinear reductive dimension approximate algorithm for 0–1 knapsack problem. *J Inner Mongolia Normal Univ (Natural Sci Ed)* 2007:1
21. Li Z, Li N (2009) A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem. In: *2009 Chinese control and decision conference*, IEEE, pp 3042–3047
22. Lin FT (2008) Solving the knapsack problem with imprecise weight coefficients using genetic algorithms. *Eur J Oper Res* 185(1):133–145
23. Lin WC, Yin Y, Cheng SR, Cheng TE, Wu CH, Wu CC (2017) Particle swarm optimization and opposite-based particle swarm optimization for two-agent multi-facility customer order scheduling with ready times. *Appl Soft Comput* 52:877–884
24. Liu Y, Liu C (2009) A schema-guiding evolutionary algorithm for 0-1 knapsack problem. In: *2009 International association of computer science and information technology-Spring Conference*, IEEE, pp 160–164
25. Mavrotas G, Diakoulaki D, Kourentzis A (2008) Selection among ranked projects under segmentation, policy and logical constraints. *Eur J Oper Res* 187(1):177–192
26. Mezura-Montes E (2009) *Constraint-handling in evolutionary optimization*, vol 198. Springer, Berlin
27. Mirjalili S, Lewis A (2013) S-shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm Evol Comput* 9:1–14
28. Mirjalili S, Mirjalili SM, Yang XS (2014) Binary bat algorithm. *Neural Comput Appl* 25(3–4):663–681

29. Mohamed AK, Mohamed AW, Elfeky EZ, Saleh M (2018) Enhancing agde algorithm using population size reduction for global numerical optimization. In: International conference on advanced machine learning technologies and applications, Springer, pp 62–72
30. Mohamed AW, Hadi AA, Mohamed AK (2019) Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. *Int J Mach Learn Cybern* 2019:1–29
31. Mohamed AW, Sabry HZ (2012) Constrained optimization based on modified differential evolution algorithm. *Inf Sci* 194:171–208
32. Moosavian N (2015) Soccer league competition algorithm for solving knapsack problems. *Swarm Evol Comput* 20:14–22
33. Shi H (2006) Solution to 0/1 knapsack problem based on improved ant colony algorithm. In: 2006 IEEE international conference on information acquisition, IEEE, pp 1062–1066
34. Truong TK, Li K, Xu Y (2013) Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem. *Appl Soft Comput* 13(4):1774–1780
35. Wang L, Wang X, Fu J, Zhen L (2008) A novel probability binary particle swarm optimization algorithm and its application. *J Softw* 3(9):28–35
36. Wang L, Yang R, Xu Y, Niu Q, Pardalos PM, Fei M (2013) An improved adaptive binary harmony search algorithm. *Inf Sci* 232:58–87
37. Yoshizawa H, Hashimoto S (2000) Landscape analyses and global search of knapsack problems. In: Smc 2000 conference proceedings. 2000 IEEE international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no. 0, vol. 3, IEEE, pp 2311–2315
38. You W (2007) Study of greedy-policy-based algorithm for 0/1 knapsack problem. *Compu Modern* 4:10–16
39. Yuan H, Zhou M, Liu Q, Abusorrah A (2020) Fine-grained resource provisioning and task scheduling for heterogeneous applications in distributed green clouds. *IEEE/CAA J Autom Sin* 7(5):1380–1393
40. Zhou Y, Chen X, Zhou G (2016) An improved monkey algorithm for a 0–1 knapsack problem. *Appl Soft Comput* 38:817–830
41. Zhu Y, Ren LH, Ding Y, Kritaya K (2008) Dna ligation design and biological realization of knapsack problem. *Chin J Comput* 31(12):2207–2214
42. Zou D, Gao L, Li S, Wu J (2011) Solving 0–1 knapsack problem by a novel global harmony search algorithm. *Appl Soft Comput* 11(2):1556–1564

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.