SUSTAINABLE MACHINE INTELLIGENCE JOURNAL

An Improved Binary Quadratic Interpolation Optimization for 0-1 Knapsack Problems

Sara Salem

Faculty of Computers and Informatics, Zagazig University, Zagazig, Sharqiyah, 44519, Egypt; ss8436754@gmail.com.

* Correspondence: ss8436754@gmail.com.

Event	Date
Received	06-05-2023
Revised	25-07-2023
Accepted	30-08-2023
Published	29-09-2023

Abstract: This paper presents a new binary optimization technique for solving the 0–1 knapsack problem. This algorithm is based on converting the continuous search space of the recently proposed quadratic interpolation optimization (QIO) into discrete search space using various V-shaped and S-shaped transfer functions; this algorithm is abbreviated as BQIO. To further improve its performance, it is effectively integrated with a uniform crossover operator and a swap operator to explore the discrete binary search space more effectively. This improved variant is called BIQIO. Both BQIO and BIQIO are assessed using 20 well-known knapsack instances and compared to four recently published metaheuristic algorithms to reveal their effectiveness. The comparison among algorithms is based on three performance metrics: the mean fitness value, Friedman mean rank and computational cost. The first two metrics are used to observe the accuracy of the results, while the last metric is employed to show the efficiency of each algorithm. The results of this comparison reveal the superiority of BIQIO over the classical BQIO and four rival optimizers.

Keywords: Quadratic interpolation optimization; 0–1 knapsack problem; Crossover operator; Transfer functions.

1. Introduction

The 0–1 knapsack problem (KP01) and its variants are both considered to be a subset of the NP-hard discrete optimization problems. The KP01 plays an essential part in a number of different problems, including those pertaining to resource allocation, production planning, project selection, computer science, and cutting stock [1]. There are two distinct categories of approaches to solving KP01: (a) deterministic algorithms and (b) metaheuristic techniques. Exact algorithms, such as dynamic programming and branch-and-bound, can provide precise and optimal solutions; however, their performance is significantly degraded with increasing the dimension size. Therefore, the metaheuristic algorithms were proposed as a strong alternative for solving those problems, regardless of their dimensions [2]. Several metaheuristic algorithms were used in the literature for solving this problem, some of them will be discussed in the next sections.

For the purpose of resolving KP01, a binary version of the Aquila optimizer (BAO) has been presented in [3]. In this variant, eight transfer functions were tested to see which one performed best in terms of boosting BAO's efficiency. Another form of BAO was proposed in the same paper, to enhance its exploration and exploitation operators by using the crossover operator and mutation operator. In a similar vein, Yildizdan developed a binary form of the artificial jellyfish search (AJS) to address the same issue; this variant is known as Bin_AJS [4]. To adapt AJS for use in the discrete search space, several transfer functions were studied to reveal the effect they had on AJS's efficiency. The experimental results have shown its advantages over other optimizers. The reptilian search algorithm (RSA) has been modified in [5] to solve KP01 using a binary form dubbed BRSA, which takes advantage of a number of transfer functions. Additionally, a stochastic repair and improvement mechanism was incorporated with BRSA to further enhance its performance. In [6], the binary artificial bee colony algorithm (ABC) with differential evolution (DE) was integrated to present a new variant namely BABC-DE to solve KP01.

The marine predators algorithm (MPA) was modified in [7] to handle 0-1 KP and other discrete issues. The binary extension of MPA (BMPA) was created by mapping continuous values to binary with the use of various transfer functions. In [8], the binary slime mould algorithm (BSMA) was enhanced to more precisely solve KP01. Similarly, the binary variation of the standard EO has been developed in [9] for solving KP01 using various transfer functions. For solving KP01, a binary monarch butterfly optimization (BMBO) strategy was proposed in [10]. In this approach, three distinct individual allocation strategies were evaluated for their potential to boost performance. The infeasible solutions are reworked while the feasible ones are optimized using a novel repair operator based on a greedy technique. The KP01 problem was addressed by presenting a novel binary bat algorithm (NBBA) in [11]. This method incorporated the best features of the local search scheme (LSS) with the binary bat algorithm (BBA). By allowing bats to improve their exploration capacity and LSS to increase their exploitation inclinations, the bat algorithm safeguards the BBA-LSS from becoming trapped in local optimum solutions.

The monarch butterfly optimization (MBO) was fortified in [12] through the implementation of chaotic maps to improve its global optimization capability and the Gaussian mutation operator to eliminate premature convergence of the optimization process by enhancing some poor individuals; this variant was called CMBO. CMBO, an enhanced variant, was implemented to solve the large-scale KP01 problem. There are several other metaheuristic algorithms for KP01, such as the whale optimization algorithm [13], rice optimization algorithm [14], gradient-based optimizer [15], harmony search algorithm [16], binary dragonfly algorithm [17], Archimedes optimization algorithm [18], cuckoo search algorithm [19], and migrating birds optimization [20].

In this paper, a binary variant of a recently proposed metaheuristic algorithm known as the quadratic interpolation optimization (BQIO), which is inspired by the generalized quadratic interpolation method is presented for solving the 0-1 knapsack optimization problems; this variant is called BQIO. This variant is assessed using the best transfer function from V-shaped and S-shaped families to strengthen its performance when tackling those problems. In addition, to further enhance its performance, it is integrated with the uniform mutation operator and swap operator to present a new variant, namely BIQIO. Both BQIO and BIQIO are investigated using 20 well-known KP01 instances and compared to four metaheuristic algorithms, including binary equilibrium optimizer, binary marine predators algorithm, binary flower pollination algorithm, and binary generalized normal distribution optimization. The experimental findings report that BIQIO is more effective.

The remainder of this work is structured as follows: Section 2 covers the QIO's mathematical model; Section 3 covers the proposed algorithms; Section 4 presents the results and discusses them; and Section 5 covers the conclusion and future work.

2. Quadratic Interpolation Optimization

This algorithm, known as Quadratic Interpolation Optimization, has been recently developed for solving continuous optimization problems [21]. This algorithm was inspired by the generalized quadratic interpolation (GQI) technique. This technique is used in the QIO algorithm as a searching mechanism for tackling several optimization problems. Similar to the other metaheuristic algorithms, this algorithm involves two phases: exploration and exploitation, which are described in detail within the next two sections.

2.1. Exploration strategy

This strategy is used by the QIO algorithm for two purposes: Avoiding falling into local minima and preserving the population diversity. In this strategy, the GQI method is used to update each solution in the population, according to the next formulas:

$$\vec{v}_i^{t+1} = \vec{x}_i^{*}(t) + w_1 \cdot \left(\overrightarrow{x_{r3}} - \vec{x}_i^{*}(t)\right) + round\left(0.5 \cdot (0.05 + r_1)\right) \cdot \log \frac{r_2}{r_3} \tag{1}$$

where r_1, r_2, and r_3 are three different variables including numbers generated at random between 0 and 1. $\overrightarrow{x_{r3}}$ is a random solution from the current population, $\vec{x}_i^{*}(t)$ is estimated by the GQI function as defined in the following formula:

$$\vec{x}_i^{*}(t) = GQI\left(\vec{x}_i^{t}, \overrightarrow{x_{r1}}, \overrightarrow{x_{r2}}, f(\overrightarrow{x_{r1}}), f(\overrightarrow{x_{r2}})\right) \tag{2}$$

where GQI is the GQI function, $\overrightarrow{x_{r1}}$ and $\overrightarrow{x_{r2}}$ are random solutions from the current population, $f(\cdot)$ is the fitness function, and $\vec{x}_i^{t}$ is the ith solution. Regarding w_1, it is mathematically generated according to the following formula:

$$w_1 = 3n_1 b \tag{3}$$

$$b = 0.7 \cdot a + 0.15 \cdot a \cdot \left(\cos\left(\frac{5\pi t}{T_{max}}\right) + 1\right) \tag{4}$$

$$a = \cos\left(\frac{\pi t}{2T_{max}}\right) \tag{5}$$

where n_1 is a normal distribution-based random number, t is the current function evaluation, T_{max} is the maximum function evaluation.

2.2. Exploitation strategy

The QIO algorithm performs the exploitation operator using the GQI method to exploit the regions around the best-so-far solution for accelerating the convergence in the right direction of the near-optimal solution. The exploitation capabilities of QIO algorithm are performed according to the following formula for each solution in the population:

$$\vec{v}_i^{t+1} = \overrightarrow{x_{best}^{*}}(t) + w_2 \cdot \left(\overrightarrow{x_{best}} - round(1 + r_4) \cdot \frac{(U-L)}{(U_{rD}-L_{rD})} \cdot \vec{x}_{i,rD}^{t}\right) \tag{6}$$

where $\overrightarrow{x_{best}}$ is the best-so-far solution, r_4 is a random number between 0 and 1, U is the upper bound, and L is the lower bound, $\overrightarrow{x_{i,rD}^{t}}$ is a vector including random dimensions selected from the ith solution, and w_2 is generated according to the following formula:

$$w_2 = 3 \cdot \left(1 - \frac{t-1}{T_{max}}\right) n_2 \quad (7)$$

where n_2 is a normal distribution-based random number. Regarding $\overrightarrow{x_{best}^{*}}(t)$, it is computed using the GQI function as shown in the following formula:

$$\overrightarrow{x_{best}^{*}}(t) = GQI(\overrightarrow{x_{best}}, \overrightarrow{x_{r1}}, \overrightarrow{x_{r2}}, f(\overrightarrow{x_{best}}), f(\overrightarrow{x_{r1}}), f(\overrightarrow{x_{r2}})) \quad (8)$$

Finally, the pseudocode of the classical QIO is explained in algorithm 1.

Algorithm 1: The classical QIO

Input: N, T_{max}
Output: c

1. Initialize randomly the solutions $\vec{x}_i (i = 1,2, \dots \dots, N)$
2. Compute the objective value of those solutions
3. Identifying $\overrightarrow{x_{best}}$ that has the best objective value among all solutions
4. $t = 1$; //the current function evaluation
5. **while** the termination condition is not achieved **do**
6. **for** each $\vec{x}_i, i \in 1{:}N$
7. Selecting two solutions $\overrightarrow{x_{r1}} \neq \overrightarrow{x_{r2}} \neq \overrightarrow{x_{r3}} \neq \overrightarrow{x_i}$ from the current solutions
8. Generating random number r_2 in $(0,1)$
9. **If** $r_2 > 0.5$
10. Applying Eq. (2) to get $\overrightarrow{x_i^{*}}(t)$
11. Performing the exploration operator using Eq. (1)
12. **else**
13. Applying Eq. (8) to get $\overrightarrow{x_{best}^{*}}(t)$
14. Performing the exploration operator using Eq. (6)
15. **end**
16. $\boldsymbol{if}$ $(f(\vec{v}_i^{t+1}) < f(\vec{x}_i^{t}))$
17. $\vec{x}_i^{t} = \vec{v}_i^{t+1};$
18. **end**
19. update $\overrightarrow{x_{best}}$ if $\vec{v}_i^{t+1}$ is better
20. $t = t + 1$
21. **End for**
22. **End while**

3. The proposed algorithm: BIQIO

The classical QIO algorithm was proposed for tackling continuous problems; thereby it is unsuitable to directly solve KP01. Therefore, it is converted into a binary algorithm using eight well-known transfer functions (TFs), belonging to V-shaped and S-shaped transfer functions. The first four functions according to this reference belong to the S-shaped and are symbolized in this study as TF1, TF2, TF3, and TF4, while the other functions belong to the V-shaped and are symbolized as TF5, TF6, TF7, and TF8. These TFs are first applied to normalize the continuous solutions between 0 and 1 that are then converted randomly into 1 and 0 according to the following formula:

$$\vec{x}_{bin} = \begin{cases} 1 & if\ F\left(v_{ij}^{t+1}\right) \geq rand \\ 0 & otherwise \end{cases} \quad (9)$$

where $F(v_{ij}^{t+1})$ represents the normalized value of the jth dimension in the ith solution, which is obtained by one of the used transfer functions. After generating the binary solution $\vec{x}_{bin}$ the ith solution, it is evaluated using the following objective function to measure its quality:

$$Maximize\ f(\vec{x}_{bin}) = \sum_{k=1}^{n} x_{bin}^{k} * pr_k \quad (10)$$
$$Subject\ to\ \sum_{z=1}^{n} w_k * x_k \leq c$$

$$x_{bin}^{k} = 0 \; or \; 1, \; k = 0,1 \ldots \ldots n$$

where $\boldsymbol{pr_k}$ represents the profit of the **kth** item, $\boldsymbol{x_{bin}^{k}}$ is the binary value used to determine whether the **kth** item will be added to the knapsack or not, $\boldsymbol{n}$ is the number of items, $\boldsymbol{w_k}$ represents the weight of each item, and $\boldsymbol{c}$ is the capacity of the knapsack. After evaluating the binary solutions, they are compared with each other, and the solution with the highest objective value and subject to the required constraint is considered the best-so-far solution $\overrightarrow{\boldsymbol{x_{best}}}$. The steps of adapting the QIO algorithm for solving KP0 are listed in Algorithm 2.

Algorithm 2: The binary QIO (BQIO)

Input: N, T_{max}
Output: $\overrightarrow{\boldsymbol{x_{best}}}$

1. Initialize randomly the solutions $\vec{x}_i(i = 1,2, \ldots \ldots, N)$ with binary values
2. Compute the objective value of those solutions according to (10)
3. Identifying $\overrightarrow{\boldsymbol{x_{best}}}$ that has the best objective value among all solutions
4. $t = 1$; //the current function evaluation
5. $\vec{x}_i^{tb} = \vec{x}_i(i = 1,2, \ldots \ldots, N)$ % storing the binary solutions
6. **while** the termination condition is not achieved **do**
7. **for** each $\vec{x}_i, i \in 1{:}N$
8. Selecting two solutions $\overrightarrow{\boldsymbol{x_{r1}}} \neq \overrightarrow{\boldsymbol{x_{r2}}} \neq \overrightarrow{\boldsymbol{x_{r3}}} \neq \overrightarrow{\boldsymbol{x_i}}$ from the current solutions
9. Generating random number r_2 in $(0, 1)$
10. **If** $\boldsymbol{r_2 > 0.5}$
11. Applying Eq. (2) to get $\overrightarrow{\boldsymbol{x_i^*}}(\boldsymbol{t})$
12. Performing the exploration operator using Eq. (1)
13. **else**
14. Applying Eq. (8) to get $\overrightarrow{\boldsymbol{x_{best}^*}}(\boldsymbol{t})$
15. Performing the exploration operator using Eq. (6)
16. **End**
17. Generate the binary solution $\vec{x}_{bin}$ of $\vec{v}_i^{t+1}$ using (9)
18. $\boldsymbol{if}$ $(f(\vec{x}_{bin}) > f(\vec{x}_i^{tb}))$
19. $\vec{x}_i^{t} = \vec{v}_i^{t+1}$;
20. $\vec{x}_i^{tb} = \vec{x}_{bin}$
21. **end**
22. update $\overrightarrow{\boldsymbol{x_{best}}}$ with $\vec{x}_{bin}$ if the last is better
23. $\mathrm{t = t + 1}$
24. **End for**
25. **End while**

However, BQIO still needs further improvements to strengthen its ability to find near-optimal solutions for the KP01 instances. Therefore, it is integrated with the uniform crossover operator, which uniform $\vec{\boldsymbol{x}}_{\boldsymbol{bin}}$ from the $\boldsymbol{ith}$ solution and $\overrightarrow{\boldsymbol{x_{best}}}$ based on a predefined crossover probability (CR) to generate a new binary solution, namely $\vec{\boldsymbol{x}}_{\boldsymbol{bin2}}$. $\vec{\boldsymbol{x}}_{\boldsymbol{bin2}}$ is further improved by selecting two positions with the condition that one of them includes 1 and the other includes 0, and swapping them to aid in achieving better outcomes. The proposed binary improved QIO (BIQIO) after adding those improvements is described in Algorithm 3.

Algorithm 3: The proposed BIQIO

Input: N, T_{max}
Output: $\overrightarrow{\boldsymbol{x_{best}}}$

1. Initialize randomly the solutions $\vec{x}_i(i = 1,2, \ldots \ldots, N)$ with binary values
2. Compute the objective value of those solutions according to (10)
3. Identifying $\overrightarrow{\boldsymbol{x_{best}}}$ that has the best objective value among all solutions
4. $t = 1$; //the current function evaluation
5. $\vec{x}_i^{tb} = \vec{x}_i(i = 1,2, \ldots \ldots, N)$ % storing the binary solutions
6. **while** the termination condition is not achieved **do**
7. **for** each $\vec{x}_i, i \in 1{:}N$
8. Selecting two solutions $\overrightarrow{\boldsymbol{x_{r1}}} \neq \overrightarrow{\boldsymbol{x_{r2}}} \neq \overrightarrow{\boldsymbol{x_{r3}}} \neq \overrightarrow{\boldsymbol{x_i}}$ from the current solutions
9. Generating random number r_2 in $(0, 1)$

If $r_2 > 0.5$
Applying Eq. (2) to get $\vec{x_i^*}(t)$
Performing the exploration operator using Eq. (1)
else
Applying Eq. (8) to get $\overrightarrow{x_{best}^*}(t)$
Performing the exploration operator using Eq. (6)
End
Generate the binary solution $\vec{x}_{bin}$ of $\vec{v}_i^{t+1}$ using (9)
Generating random number r_3 in $(0, 1)$
$if\ r_3 \leq \frac{t}{T_{max}}$
$\vec{x}_{bin2}$=Applying the uniform crossover operator between $\vec{x}_{bin}$ and $\overrightarrow{x_{best}}$ under Cr
Selecting two unique positions from $\vec{x}_{bin2}$ and swap them
$\vec{x}_{bin}$=$\vec{x}_{bin2}$
end
$\boldsymbol{if}\ (f(\vec{x}_{bin}) > f(\vec{x}_i^{tb}))$
$\vec{x}_i^t = \vec{v}_i^{t+1};$
$\vec{x}_i^{tb} = \vec{x}_{bin}$
end
update $\overrightarrow{x_{best}}$ with $\vec{x}_{bin}$ if the last is better
t = t + 1
End for
End while

4. Results and Discussion

The proposed BQIO and BIQIO are assessed using 20 well-known KP01 instances with a number of items ranging between 4 and 75 to observe the ability to estimate the near-optimal solutions under different scenarios. The properties of those instances are listed in Table 1. Those proposed algorithms are also compared to five well-known optimization algorithms, such as binary equilibrium optimizer (BEO), binary marine predators' algorithm (BMPA), binary flower pollination algorithm (BFPA), and binary generalized normal distribution optimization (BGNDO). The parameters of those algorithms are set as suggested in the cited papers. Regarding the parameters of the proposed BIQIO, it has only one parameter, namely CR, which is heuristically set to 0.9 in all experiments conducted in this paper. The maximum function evaluations and population size for all algorithms are set to 50000 and 100, respectively, to achieve a fair comparison. All algorithms are implemented in MATLAB over the same device.

Table 1: Properties of KP01 instances

Instances	Items	Knapsack capacity	Known optimum	Instances	Items	Knapsack capacity	Known optimum	Instances	Items	Knapsack capacity	Known optimum
K1	10	269	295	K8	23	10000	9767	K15	50	882	2440
K2	20	878	1024	K9	5	80	130	K16	55	1050	2651
K3	4	20	35	K10	20	879	1025	K17	60	1006	2917
K4	4	11	23	K11	30	577	1437	K18	65	1319	2817
K5	15	375	481.06937	K12	35	655	1689.0	K19	70	1426	3223
K6	10	60	52	K13	40	819	1821	K20	75	1433	3614
K7	7	50	107	K14	45	907	2033				

4.1. Performance analysis of various TFs with BQIO

In this section, the performance of eight transfer functions with the proposed BQIO will be investigated to find the most effective one. The proposed BQIO with each transfer function is executed 20 independent times. Then, the mean fitness values (Mean), the Friedman mean rank (FRK), and the computational cost (Time) are estimated and presented in Tables 2 and 3. Those tables illustrate that BQIO with TF8 could achieve competitive and superior outcomes in terms of Mean and FRK metrics for the majority of the solved instances. Therefore, this transfer function is considered for both BQIO and BIQIO in the experiments conducted in the next section.

Table 2: Performance observation of various TFs with BQIO (K1-K10)

Inst		*TF1*	*TF2*	*TF3*	*TF4*	*TF5*	*TF6*	*TF7*	*TF8*
K1	Mean	295.0000	295.0000	295.0000	295.0000	295.0000	295.0000	295.0000	295.0000
	FRK	4.500	4.500	4.500	4.500	4.500	4.500	4.500	4.500
	Time	1.082E-03	4.367E-04	7.523E-04	1.947E-03	4.411E-03	5.463E-04	9.301E-04	1.768E-03
K2	Mean	1024.0000	1024.0000	1024.0000	1024.0000	1024.0000	1024.0000	1024.0000	1024.0000
	FRK	4.500	4.500	4.500	4.500	4.500	4.500	4.500	4.500
	Time	8.849E-02	1.004E-02	9.117E-02	3.302E-02	1.911E-02	5.182E-03	4.808E-02	2.471E-03
K3	Mean	35.0000	35.0000	35.0000	35.0000	35.0000	35.0000	35.0000	35.0000
	FRK	4.500	4.500	4.500	4.500	4.500	4.500	4.500	4.500
	Time	4.799E-05	4.696E-05	4.496E-05	4.269E-05	4.625E-05	4.454E-05	4.457E-05	4.410E-05
K4	Mean	23.0000	23.0000	23.0000	23.0000	23.0000	23.0000	23.0000	23.0000
	FRK	4.500	4.500	4.500	4.500	4.500	4.500	4.500	4.500
	Time	5.736E-05	4.456E-05	4.463E-05	4.035E-05	4.301E-05	4.249E-05	4.285E-05	4.175E-05
K5	Mean	481.06937	481.06937	481.06937	481.06937	481.06937	481.06937	481.06937	481.06937
	FRK	4.500	4.500	4.500	4.500	4.500	4.500	4.500	4.500
	Time	4.774E-03	1.794E-02	2.694E-03	6.386E-02	2.942E-03	1.168E-01	9.296E-03	3.791E-02
K6	Mean	52.0000	52.0000	52.0000	52.0000	52.0000	52.0000	52.0000	52.0000
	FRK	4.500	4.500	4.500	4.500	4.500	4.500	4.500	4.500
	Time	4.264E-04	4.584E-05	4.744E-05	1.450E-03	4.592E-05	6.915E-04	4.626E-05	7.566E-04
K7	Mean	107.0000	107.0000	107.0000	107.0000	107.0000	107.0000	107.0000	107.0000
	FRK	4.500	4.500	4.500	4.500	4.500	4.500	4.500	4.500
	Time	4.369E-05	5.798E-05	4.587E-04	4.102E-04	5.548E-05	4.572E-05	3.619E-04	3.041E-04
K8	Mean	9766.7500	9767.0000	9766.8000	9767.0000	9767.0000	9766.8000	9767.0000	9767.0000
	FRK	4.800	4.400	4.600	4.400	4.400	4.600	4.400	4.400
	Time	1.548E-01	9.007E-02	2.028E-01	6.090E-02	3.321E-01	1.072E-01	1.136E-01	2.930E-01
K9	Mean	130.0000	130.0000	130.0000	130.0000	130.0000	130.0000	130.0000	130.0000
	FRK	4.500	4.500	4.500	4.500	4.500	4.500	4.500	4.500
	Time	4.58E-05	4.76E-05	4.58E-05	4.55E-05	4.30E-05	4.41E-05	4.63E-05	0.0003309
K10	Mean	1025.0000	1025.0000	1025.0000	1025.0000	1025.0000	1025.0000	1025.0000	1025.0000
	FRK	4.500	4.500	4.500	4.500	4.500	4.500	4.500	4.500
	Time	4.705E-02	5.166E-02	4.915E-02	3.311E-02	1.304E-01	4.689E-02	2.947E-02	8.882E-02

Table 3: Performance observation of various TFs with BQIO (K11-K16)

Inst		*TF1*	*TF2*	*TF3*	*TF4*	*TF5*	*TF6*	*TF7*	*TF8*
K11	Mean	1427.9500	1428.3500	1429.1000	1429.2000	1420.6500	**1432.2500**	1431.5000	1431.8000
	FRK	4.750	4.725	4.575	4.425	6.300	**3.675**	3.850	3.700
	Time	4.632E-01	4.706E-01	1.713E-01	4.802E-01	5.102E-01	4.104E-01	5.009E-01	**1.702E-01**
K12	Mean	1683.3500	1682.5000	1683.3500	1680.7000	1679.0500	1688.1000	1687.8500	**1688.5000**
	FRK	4.950	5.300	5.075	5.450	6.450	3.050	3.050	**2.675**
	Time	5.341E-01	5.439E-01	4.919E-01	5.703E-01	5.815E-01	3.595E-01	**2.565E-01**	6.381E-01
K13	Mean	1803.3000	1802.5000	1797.7500	1798.8000	1786.6000	1812.6500	1807.0500	**1814.4000**
	FRK	4.625	4.250	5.375	5.100	6.825	2.900	4.200	**2.725**
	Time	**6.333E-01**	6.580E-01	6.805E-01	6.595E-01	1.503E+00	1.468E+00	7.747E-01	1.673E+00
K14	Mean	2007.2500	2007.5500	2007.2000	2005.2000	1981.0500	2018.8500	2006.6000	**2018.7500**
	FRK	4.750	4.650	4.325	4.875	7.150	3.225	4.350	**2.675**
	Time	1.453E+00	**1.482E+00**	1.448E+00	1.483E+00	1.521E+00	1.514E+00	1.555E+00	1.744E+00
K15	Mean	2412.0000	2399.5500	2408.1500	2399.6000	2374.8000	2424.4000	2420.6000	**2424.5000**
	FRK	3.900	5.350	4.525	5.575	6.950	3.125	3.475	**3.100**
	Time	8.466E-01	**2.729E-01**	8.705E-01	9.095E-01	8.887E-01	8.714E-01	8.516E-01	9.167E-01
K16	Mean	2591.5000	2596.2000	2589.2500	2581.9500	2534.1000	2609.6500	2585.6000	**2614.0000**
	FRK	4.575	3.950	4.500	5.325	7.650	2.875	4.700	**2.425**
	Time	8.87E-01	**9.17E-01**	1.07E+00	9.96E-01	1.09E+00	1.03E+00	1.50E+00	1.25E+00

4.2. Performance evaluation of BIQIO with four rival optimizers

All algorithms are executed 20 independent times, and then the mean, FRK, and times are computed and reported in Table 4. In comparison to all the rival algorithms, this table shows that BIQIO is competitive in terms of Mean and FRK for 10 instances and superior for the other instances. Also, from this table, BQIO could be competitive with both BEO and BGNDO for the majority of the instances, while both BFPA and BMPA appear as the worst algorithms. Finally, those experiments show the effectiveness of BIQIO in comparison to the classical BQIO and four rival algorithms, so it could be considered as an alternative algorithm for solving the 0-1 knapsack instances.

Table 4: Performance observation of BQIO and BIQIO with four rival algorithms

Inst		BIQIO	BQIO	BEO	BMPA	BFPA	BGNDO		BIQIO	BQIO	BEO	BMPA	BFPA	BGNDO
K1	Mean	**295.00**	**295.00**	**295.00**	**294.55**	**295.00**	**295.00**	K11	**1437.00**	1425.55	1431.50	1379.00	1244.80	1418.00
	FRK	**3.45**	**3.45**	**3.45**	**3.75**	**3.45**	**3.45**		**1.53**	2.73	2.15	5.00	6.00	3.60
	Time	2.E-03	2.E-03	1.E-03	1.E-03	1.E-03	**8.E-04**		2.E-01	2.E-01	**2.E-02**	2.E-01	1.E-01	2.E-01
K2	Mean	**1024.00**	**1024.00**	**1024.00**	1017.85	930.25	**1024.00**	K12	**1689.00**	1681.55	1686.95	1625.35	1375.75	1673.40
	FRK	**2.70**	**2.70**	**2.70**	4.20	6.00	**2.70**		**1.55**	2.78	2.05	4.75	6.00	3.88
	Time	5.E-02	3.E-02	**1.E-03**	2.E-02	2.E-01	4.E-02		1.E-01	6.E-01	**3.E-02**	2.E-01	1.E-01	2.E-01
K3	Mean	**35.00**	**35.00**	**35.00**	**35.00**	**35.00**	**35.00**	K13	**1820.60**	1794.70	1812.05	1727.35	1475.95	1779.05
	FRK	**3.50**	**3.50**	**3.50**	**3.50**	**3.50**	**3.50**		**1.13**	3.10	2.13	4.80	6.00	3.85
	Time	7.E-05	6.E-05	2.E-04	6.E-05	7.E-05	**5.E-05**		2.E-01	6.E-01	**4.E-02**	2.E-01	1.E-01	2.E-01
K4	Mean	**23.00**	**23.00**	**23.00**	**23.00**	**23.00**	**23.00**	K14	**2031.70**	2007.00	2013.95	1871.45	1579.15	1952.05

	FRK	**3.50**	**3.50**	**3.50**	**3.50**	**3.50**	**3.50**		**1.05**	2.73	2.23	5.00	6.00	4.00
	Time	**5.8E-05**	7.0E-05	1.3E-04	6.4E-05	8.1E-05	8.1E-05		6.E-01	7.E-01	**4.E-02**	2.E-01	1.E-01	2.E-01
K5	Mean	**481.07**	**481.07**	**481.07**	469.03	473.69	**481.07**	K15	**2441.15**	2401.85	2429.95	2272.25	1871.10	2345.40
	FRK	**2.90**	**2.90**	**2.90**	4.15	5.25	**2.90**		**1.25**	2.95	1.95	4.85	6.00	4.00
	Time	1.1E-02	3.4E-02	2.9E-03	**2.4E-03**	2.2E-01	6.3E-03		2.E-01	8.E-01	**5.E-02**	3.E-01	2.E-01	3.E-01
K6	Mean	**52.00**	**52.00**	**52.00**	**51.95**	**52.00**	**52.00**	K16	**2643.25**	2592.75	2602.85	2430.35	2027.10	2504.00
	FRK	**3.48**	**3.48**	**3.48**	**3.63**	**3.48**	**3.48**		**1.10**	2.65	2.25	4.90	6.00	4.10
	Time	4.1E-03	4.4E-03	1.4E-04	1.6E-04	6.2E-03	**8.7E-05**		7.E-01	1.E+00	**5.E-02**	3.E-01	2.E-01	2.E-01
K7	Mean	**107.00**	**107.00**	**107.00**	**106.80**	**107.00**	**107.00**	K17	**2915.70**	2852.85	2874.95	2666.65	2124.95	2748.75
	FRK	**3.45**	**3.45**	**3.45**	**3.75**	**3.45**	**3.45**		**1.10**	2.68	2.23	5.00	6.00	4.00
	Time	1.0E-04	**6.4E-05**	1.4E-04	9.1E-05	5.6E-04	6.9E-05		3.5E-01	9.9E-01	**5.2E-02**	2.9E-01	1.7E-01	2.6E-01
K8	Mean	**9767.00**	9766.90	9766.15	9754.60	9758.40	**9767.00**	K18	**2814.95**	2738.20	2787.65	2562.50	2048.40	2643.70
	FRK	**2.33**	2.43	3.00	5.63	5.30	**2.33**		**1.00**	3.00	2.00	4.95	6.00	4.05
	Time	**4.E-02**	2.E-01	3.E-02	2.E-01	1.E-01	5.E-02		4.E-01	1.E+00	**6.E-02**	3.E-01	2.E-01	2.E-01
K9	Mean	**130.00**	**130.00**	**130.00**	**130.00**	**130.00**	**130.00**	K19	**3217.65**	3156.70	3186.55	2944.95	2339.10	3056.70
	FRK	**3.50**	**3.50**	**3.50**	**3.50**	**3.50**	**3.50**		**1.00**	2.85	2.15	5.00	6.00	4.00
	Time	6.E-05	6.E-05	3.E-04	6.E-05	5.E-05	**4.E-05**		1.E+00	1.E+00	**6.E-02**	3.E-01	2.E-01	3.E-01
K10	Mean	**1025.00**	**1025.00**	**1025.00**	1017.00	934.70	**1025.00**	K20	**3602.00**	3486.45	3526.75	3249.95	2581.00	3344.75
	FRK	**2.78**	**2.78**	**2.78**	3.90	6.00	**2.78**		**1.00**	2.70	2.35	4.80	6.00	4.15
	Time	3.E-02	4.E-03	**3.E-03**	3.E-01	3.E-01	2.E-02		1.E+00	1.E+00	**7.E-02**	3.E-01	2.E-01	3.E-01

5. Conclusion

In this article, we introduce a novel binary optimization strategy for the 1-in-0 knapsack problem. BQIO is an algorithm that uses a variety of V-shaped and S-shaped transfer functions to transform the continuous search space of the recently introduced quadratic interpolation optimization (QIO) into a discrete search space. Its performance is enhanced by the incorporation of a uniform crossover operator and a swap operator, which allow for more efficient exploration of the discrete binary search space. The name "BIQIO" describes this upgraded version. Twenty well-known knapsack examples are used to evaluate BQIO and BIQIO, and their performance is compared to that of four recently published metaheuristic methods. Mean fitness value, Friedman mean rank and computing cost are the three performance indicators used for the algorithms' comparison. The first two measures are intended to evaluate the precision of the outcomes, while the third is used to compare the effectiveness of various algorithms. The comparison shows that BIQIO is better than the classical BQIO and four other optimizers.

In the future, both BQIO and BIQIO will be employed for solving multidimensional knapsack problems, while the performance of the standard QIO will be assessed for solving several other optimization problems, including:

- 3-D Routing Planning for Unmanned Aircraft Vehicle
- DNA Fragment assembly problem
- Lost Target Search with Unmanned Aircraft Vehicle
- UAV-Assisted IoT Data Collection System

- Joint mining decision and resource allocation in an MEC-enabled wireless blockchain network.
- Tuning hyper-parameters of machine learning algorithms and deep neural networks
- Task Scheduling in Cloud Computing
- Energy efficiency in the IoT networks
- Placement Optimization for Multi-IRS-Aided Wireless Communications
- Energy-Efficient Trajectory Planning for Multi-UAV-Assisted MEC System

Supplementary Materials

Not applicable.

Funding

This research was conducted without external funding support.

Ethical approval

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

Conflicts of Interest

The authors declare that there is no conflict of interest in the research.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

All data used to support the findings of this study are available upon request.

6. References

[1]. Kellerer, H., et al., Introduction to NP-Completeness of knapsack problems. Knapsack problems, 2004: p. 483-493.

[2]. Abdel-Basset, M., D. El-Shahat, and A.K. Sangaiah, A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem. International Journal of Machine Learning and Cybernetics, 2019. 10: p. 495-514.

[3]. Baş, E., Binary Aquila Optimizer for 0–1 knapsack problems. Engineering Applications of Artificial Intelligence, 2023. 118: p. 105592.

[4]. Yildizdan, G. and E. Baş, A Novel Binary Artificial Jellyfish Search Algorithm for Solving 0–1 Knapsack Problems. Neural Processing Letters, 2023: p. 1-67.

[5]. Ervural, B. and H. Hakli, A binary reptile search algorithm based on transfer functions with a new stochastic repair method for 0–1 knapsack problems. Computers & Industrial Engineering, 2023. 178: p. 109080.

[6]. Cao, J., et al., A modified artificial bee colony approach for the 0-1 knapsack problem. Applied Intelligence, 2018. 48: p. 1582-1595.

[7]. Abdollahzadeh, B., et al., An enhanced binary slime mould algorithm for solving the 0–1 knapsack problem. Engineering with Computers, 2021: p. 1-22.

[8]. Abdel-Basset, M., R. Mohamed, and S. Mirjalili, A Binary Equilibrium Optimization Algorithm for 0–1 Knapsack Problems. Computers & Industrial Engineering, 2020: p. 106946.

[9]. Feng, Y., et al., Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. Neural computing and applications, 2017. 28: p. 1619-1634.
[10]. Rizk-Allah, R.M. and A.E. Hassanien, New binary bat algorithm for solving 0–1 knapsack problem. Complex & Intelligent Systems, 2018. 4: p. 31-53.
[11]. Feng, Y., et al., Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation. Memetic Computing, 2018. 10: p. 135-150.
[12]. Alamri, H.S., et al. Solving 0/1 knapsack problem using opposition-based whale optimization algorithm (OWOA).
[13]. Shu, Z., et al., A modified hybrid rice optimization algorithm for solving 0-1 knapsack problem. Applied Intelligence, 2022. 52(5): p. 5751-5769.
[14]. Abdel-Basset, M., et al., Recent metaheuristic algorithms with genetic operators for high-dimensional knapsack instances: A comparative study. Computers & Industrial Engineering, 2022. 166: p. 107974.
[15]. 16.Liu, K., et al., A hybrid harmony search algorithm with distribution estimation for solving the 0-1 knapsack problem. Mathematical Problems in Engineering, 2022. 2022.
[16]. Fang, L., Y. Yao, and X. Liang, New Binary Archimedes Optimization Algorithm and its application. Expert Systems with Applications, 2023: p. 120639.
[17]. Feng, Y., K. Jia, and Y. He, An improved hybrid encoding cuckoo search algorithm for 0-1 knapsack problems. Computational intelligence and neuroscience, 2014. 2014: p. 1-1.
[18]. Ulker, E. and V. Tongur, Migrating birds optimization (MBO) algorithm to solve knapsack problem. Procedia computer science, 2017. 111: p. 71-76.
[19]. Zhao, W., et al., Quadratic Interpolation Optimization (QIO): A new optimization algorithm based on generalized quadratic interpolation and its applications to real-world engineering problems. Computer Methods in Applied Mechanics and Engineering, 2023. 417: p. 116446.
[20]. Rodrigues, D., et al., Binary flower pollination algorithm and its application to feature selection. Recent advances in swarm intelligence and evolutionary computation, 2015: p. 85-100.
[21]. Ahmed, S., et al., Binary simulated normal distribution optimizer for feature selection: Theory and application in COVID-19 datasets. Expert Systems with Applications, 2022. 200: p. 116834.