Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 10(10): 23306–23336.
DOI: 10.3934/math.20251034
Received: 08 August 2025
Revised: 15 September 2025
Accepted: 30 September 2025
Published: 14 October 2025

Research article

Mathematical modeling and improved memetic algorithm for the extended simplified discounted {0-1} knapsack problem with randomness

Zhouxi Qin[1], **Dazhi Pan**[1,2,*], **Ke Yang**[1] **and Dapeng Gao**[1]

[1] School of Mathematical Sciences, China West Normal University, Nanchong 637009, China

[2] Sichuan Colleges and Universities Key Laboratory of Optimization Theory and Applications, Nanchong 637009, China

* **Correspondence:** Email: pdzzj@cwnu.edu.cn.

Abstract: In this paper, we first extended the extended simplified discount {0-1} backpack problem (ESD {0-1} KP) and proposed the extended simplified discounted {0-1} knapsack problem with randomness (ESD {0-1} KP-R) model. Compared with the ordinary ESD {0-1} KP model, it increases the size and randomness of the term set, which is more suitable for the actual concept of "discount" and has stronger generalization. Then we used the improved memetic algorithm to solve the ESD {0-1} KP-R model. First, a greedy strategy with a relax variable was first designed to obtain the initial solution. Second, designed a crossover strategy with fuzzy sets to generate a good offspring population. Third, in order to overcome the shortcoming of memetic algorithms falling into local optimization, we designed a population diversity adjustment strategy with an information vector. This strategy combines the parent and child populations into a set of candidate solutions, and then divides all the solutions in the set into four categories according to the fitness and diversity of the solutions. Different selection methods can be used to adjust the diversity of the population while ensuring the quality of the solutions selected. In addition, based on the profit density for combinations of terms, three kinds of neighborhood structures were designed. They were used to improve the algorithm's searching ability, explore the neighborhood of local solutions, and jump out of the local optimum, respectively. The local search was performed by the variable neighborhood search algorithm. Finally, the effectiveness of the proposed improvement strategy and the feasibility of the proposed algorithm for solving the ESD {0-1} KP-R were demonstrated through experimental analysis on four types of instances.

Keywords: ESD {0-1} KP-R; memetic algorithm; population diversity adjustment strategy; relaxed greedy initialization; fuzzy crossover strategy; local search
Mathematics Subject Classification: 90C27, 90C59

1. Introduction

The 0-1 knapsack problem (0-1 KP) is the most basic knapsack problem (KP) [1]. The discounted {0-1} knapsack problem (D {0-1} KP) is an extended form of the 0-1 KP. It maximizes profit by simulating the phenomenon of discounted sales in real business. Since Guder [2] proposed this problem, the D {0-1} KP problem has been more widely studied: Rong et al. [3] combined the core idea and dynamic programming methods to solve the D {0-1} KP, and He Yizhao et al. [4] established two new mathematical models for the D {0-1} KP and solved it based on the distinguished person retention strategy. Zuhua Dai et al. [5] proposed a greedy repair method based on the density of items, and further constructed the PSO based group greedy repair and optimization algorithm for solving D0-1KP (PSO-GGRDKP). Due to the excessive non-normal coding of the D {0-1} KP model and the limited acceleration methods for solving it, Yang Yang et al. [6] established a model of the simplified discounted {0-1} knapsack problem (SD {0-1} KP) by combining the distinguished person retention strategy, greedy strategy, and high-precision penalty function method to improve the genetic algorithm to solve the problem. Thereafter, Zhang Qin et al. [7] extended the items in the SD {0-1} KP set from two to three to propose the ESD {0-1} KP problem, which increased the number of items in the set and made the discount sale more relevant to reality. Lin Hong et al. [8], on the basis of the ESD {0-1} KP model, added a virtual item with a profit and weight of 0, relaxing the constraints, proving theoretically that the ESD {0-1} KP is equivalent to the multiple-choice knapsack problem (MCKP), and proposing a new greedy strategy operator (NGSOR). However, in real-world discount scenarios, promotional bundles come in a wide variety rather than being fixed to include only three items per bundle. Considering that the number of items in the set of the ESD {0-1} KP model is small and the number of items in the set is fixed, this paper further extends the ESD {0-1} KP model and proposes the ESD {0-1} KP-R problem, which is closer to the actual sale.

When the range of the items, sets, profit, and weight in the ESD {0-1} KP-R instance increase, the exact algorithms such as dynamic programming and branch bounding lack practicality. The intelligent algorithms are the mainstream algorithms for solving these problems. At present due to the advantages of having fewer parameters and faster solution speeds, researching how to use intelligent algorithms to solve the ESD {0-1} KP-R is a worth study. The genetic algorithm (GA) is an intelligent algorithm proposed by Professor Holland with reference to the law of biological evolution—"the survival of the fittest" [9], which is widely used in combinatorial optimization, numerical optimization, machine learning, neural networks, and other fields [4], but Rudolph [10] pointed out that the standard GA does not have global convergence. To address this significant limitation of the GA algorithm, Moscato et al. [11] proposed the memetic algorithm (MA). In a broad sense, a "meme" refers to an intelligent algorithm capable of self-replication through imitation and thus propagation within a meme pool [12]. The MA can be regarded as a hybrid approach combining global search based on GA and heuristic-driven local search [13]; however, premature convergence remains a typical drawback [14]. This study aims to enhance the memetic algorithm for solving the ESD {0-1} KP-R problem. The main contributions are as follows:

- We have proposed the ESD {0-1} KP-R by extending the ESD {0-1} KP in both size and randomness.
- We have developed a mathematical model for the problem along with meaningful parametric constraints.

- We have employed a binary solution representation and a randomized repair strategy for infeasible solutions.
- We have designed a greedy strategy with a relaxation variable for initial solutions and a fuzzy-set-based crossover for offspring generation.
- We have introduced a dynamic population diversity adjustment mechanism to prevent premature convergence and enhance local search via variable neighborhood search.
- We have combined parent and offspring populations during an update using an information vector, achieving at least 20% expected average optimization efficiency improvement, especially in large-scale instances.

2. Literature review

The backpack problem has been widely studied as a typical combinatorial optimization problem, and many different types of backpack problems have been derived based on the {0-1} backpack problem. The discount {0-1} backpack problem is one of them [15]. It can be defined as: We are given a set of n items and a backpack with capacity C. Each set contains 3 items, and the items in the i-th set are labeled as $3i, 3i+1, 3i+2$, with profit coefficients $p_{3i}, p_{3i+1}, p_{3i+2}$, weight coefficients $w_{3i}, w_{3i+1}, w_{3i+2}$, and they satisfy the conditions: $p_{3i+2} = p_{3i} + p_{3i+1}$, $w_{3i+2} < w_{3i} + w_{3i+1}$, $w_{3i+2} > max\{w_{3i}, w_{3i+1}\}$. It is required that at most one item from each set of items be removed and placed into the backpack. Find the scheme for placing the items into the backpack so that the coefficient of profit is maximized while the weight of the items in the backpack does not exceed the capacity of the backpack [16]. The current solutions for the D {0-1} KP can be mainly categorized into exact and intelligent algorithms. Among them, the exact algorithms are mainly based on dynamic programming methods, such as Rong et al. [3], which decomposed the problem into several easy-to-solve sub-problems through the kernel idea, and then solved it by dynamic programming algorithms. He et al. [4] first gave a recursive formula based on the principle of weight minimization, and then solved the problem by using dynamic programming methods.

Intelligent algorithms are the dominant algorithms currently used in the study of the D {0-1} KP. These algorithms illustrate the performance of the algorithms through experimental comparisons with standard examples. We can give some examples and locate our algorithms based on the methods proposed in these literatures. Zhu et al. [17] proposed a discrete differential algorithm based on the two encoding mechanisms for transforming real vectors into integer vectors, and two new algorithms for solving the D {0-1} KP. Feng et al. [18] proposed an algorithm called monarch butterfly optimization (MBO) for solving the D {0-1} KP problem by generating offspring (positional updates) via the migration operator, and using it to adjust the positions of other butterflies. Yang Yang et al. [19] modified the greedy repair and optimization algorithm (GROA) in the first genetic algorithm (FirEGA) to obtain a new greedy repair optimization algorithm (NGROA) for dealing with non-normal coding individuals. Liu et al. [20] proposed the chaotic crow search algorithm based on the differential evolution strategy (DECCSA) and solved the coding problem of the D{0-1}KP by using a hybrid coding approach and greedy repair and optimization strategy (GROS). He et al. [21] proposed a ring theory-based evolutionary algorithm (RTEA) which generates individuals by utilizing the global exploration operator (R-GEO) and local development operator (R-LDO), and applied a greedy selection strategy to generate a new population. Xu et al. [22] designed the Lagrange interpolation-based learning monkey algorithm (LSTMA) for the existing algorithms in solving the D {0-1} KP with large scale and high complexity. Wu et al. [23] proposed an optimization algorithm based on the hybrid teaching-learning-based optimization algorithm (HTLBO), where a two-tuple consisting of a quaternary vector and a real vector is used to represent an individual in the HTLBO so that the TLBO can solve the D{0-1}KP efficiently. Truong [24] proposed a moth-flame optimization algorithm to solve the D {0-1} KP and used a new coding scheme and a greedy repair procedure to help the algorithm converge quickly. Hao et al. [25] proposed a discrete hybrid multi-verse optimization algorithm (DHMVO) with quaternary coding to solve the D {0-1} KP. Zhang et al. [26] proposed a modified ant colony optimization (MACO) algorithm for solving the D {0-1} KP and used a hybrid greedy optimization operator based on profit density and the profit of items to enhance the repaired solutions constructed by ants. Sulaiman et al. [27] proposed a fitness-based acceleration coefficient binary particle swarm optimization (FACBPSO) method so that the values of acceleration coefficients were based on the fitness of each particle, and this modification sped up the convergence and reduced the computation time. Zhang et al. [28] proposed an enhanced group theory-based optimization algorithm (EGTOA) and, based on it, a new method for solving the D {0-1} KP was given. Zhang et al. [29] addressed the 0-1 knapsack problem with complex discount relationships by proposing an efficient hybrid algorithm that combined simulated annealing and adaptive differential evolution. Dai et al. [30] proposed three discrete jaya algorithm integrated with reinforcement learning (DJAYA-RL) by discretely improving the JAYA algorithm through Q-learning.

3. Extended simplified discounted {0-1} knapsack problem with randomness

In this work, we further extend the ESD {0-1} KP problem [6] in two ways: i) the number of items in the set is increased; ii) the number of items is no longer fixed across sets. This is done by giving a random integer, and the number of items in each set is a random integer on the interval from 1 to this integer. This will increase the variety of combinations of items in a set, which is more relevant to the actual discount relationship. The extended problem is defined as follows.

Definition 3.1. *(Extended simplified discounted {0-1} knapsack problem with randomness) Let there be n sets* $E_1, E_2, \cdots, E_n$*, for any set* E_i*,* $i = 1, 2, \cdots, n$*, which contains* r_i *items* $(1 \leq r_i \leq r)$*, and note that the profit coefficients of these* r_i *items are* $P_i = (p_{i,1}, p_{i,2}, \cdots, p_{i,r_i})$*. The weight coefficients are* $W_i = (w_{i,1}, w_{i,2}, \cdots, w_{i,r_i})$*, and there are a total of* $(2^{r_i} - 1)$ *combinations to choose items from the set* E_i*. To simulate real-world discount scenarios, the discount rate is determined based on the quantity of purchased items [3]. Assuming that the number of items with the same combination of discount coefficients are equal, the discount coefficient of the combination is* $D_i = (d_{i,1}, d_{i,2}, \cdots, d_{i,r_i})$*,* $0 < d_{i,j} < 1$*,* $j = 1, 2, \cdots, r_i$*. The discount factors are uniformly distributed and satisfy* $d_{i,1} > d_{i,2} > \cdots > d_{i,r_i}$*. Given that the maximum load of the backpack is C, select some items (at least one) from each set to be put into the backpack. Require that the total weight of the items in the backpack does not exceed the capacity of the backpack C, and maximize the total profit of all items in the backpack.*

In order for the problem to make sense, the relevant parameters need to be fully conditional:

- *If the combination with the smallest weight is chosen, the total weight does not exceed the backpack's capacity.*
- *If the combination with the largest weight is chosen, the total weight is not less than the backpack's capacity.*

Note that $M_{i,1}, M_{i,2}, \cdots, M_{i,\Delta j}$ $(1 \leq j \leq r_i)$ denotes any combination of j items contained in the i-th set, Δj denotes the number of all combinations containing j items, so we have that $M_{i,q} \in \{M_{i,1}, M_{i,2}, \cdots, M_{i,\Delta j}\}$ holds, where $q \in \{1, 2, \cdots, \Delta j\}$, and the total weight of all set combinations can be expressed as:

$$\varXi = \sum_{i=1}^{n} \left(\sum_{k=1}^{j} w_{i,m_{q_k}} \cdot d_{i,j} \right), \tag{1}$$

where $m_{q_k} \in \{1, 2, \cdots, r_i\}$ denotes a subscript of the k-th item in the q-th combination Mi, q; and $w_{i,m_{q_k}} \in \{w_{i,1}, w_{i,2}, \cdots, w_{i,r_i}\}$ $(k = 1, 2, \cdots, j)$ denotes the weight of the k-th item of the q-th combination containing the j items in the i-th set.

Thereby, the above constraints can be expressed as follows:

$$\min\{\varXi\} \leq C \leq \max\{\varXi\}. \tag{2}$$

Based on the above definition, a mathematical model is developed for the ESD {0-1} KP-R:

$$(ESD\ \{0\text{-}1\}\ KP\text{-}R) \quad \begin{cases} \max & f(x) = \sum_{i=1}^{n} \sum_{j=1}^{r_i} x_{i,j} \cdot p_{i,j}, \\ s.t. & \sum_{i=1}^{n} \left(\sum_{j=1}^{r_i} w_{i,j} x_{i,j} \cdot d_{i,*} \right) \leq C, \end{cases} \tag{3}$$

where $X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,r_i}) \in \{0, 1\}^{r_i}$, $i = 1, 2, \cdots, n$, denotes whether the j-th $(j \in \{1, 2, \cdots, r_i\})$ item of set E_i is put in the backpack or not and $d_{i,*}$ denotes the discount coefficient corresponding to the currently selected combination of items in set E_i.

Beyond its theoretical interest, the ESD {0-1} KP-R model formulated in this paper has significant practical implications. A prominent application lies in E-commerce promotion and product selection. Online platforms often offer various discount strategies (e.g., “buy-one-get-one-free”, “volume discounts”) which alter the value-to-weight ratio of items in a non-linear manner. The core challenge for merchants is to select an optimal assortment of products to promote under a limited “knapsack” capacity, such as the number of front-page exposure spots or a fixed advertising budget, with the goal of maximizing total profit. Our proposed algorithm is designed to tackle precisely such complex decision-making scenarios efficiently. The insights gained from this study could also be extended to other areas such as financial portfolio optimization under fee discounts and cloud resource procurement.

4. The proposed improved memetic algorithm

4.1. Algorithmic framework

The memetic algorithm uses genetic operators for global search and local development through local search, which can ensure the convergence of the algorithm and obtain high quality solutions at the same time. In this paper, we use the idea of a greedy algorithm and the relaxation of constraints for solution initialization. In addition, the crossover strategy has been improved. Because MA is prone to premature convergence, we overcome this shortcoming by dynamically adapting the diversity of the population under the basic framework of the MA. Also, we designed the local search operators for the characteristics of the ESD {0-1} KP-R problem. The basic framework of the proposed algorithm is as follows (Algorithm 1):

Algorithm 1 Basic algorithmic framework.

```
Input: ESD {0–1} KP-R Instance
Output: Optimal Solution $X^*$
 1: $POP_{Parent} \leftarrow \{X_{Greedy}, X_2, \cdots, X_n\}$
 2: for $i = 1, 2, \cdots, n$ do
 3:     $X_i \leftarrow VNS(X_i)$
 4: end for
 5: while stop condition not met do
 6:     $POP_{Offspring} = \varnothing$
 7:     for $i = 1, 2, \cdots, n$ do
 8:         two parents $X_i, X_j$ are randomly selected from $POP$
 9:         $off_1, off_2 \leftarrow Fuzzy\_Crossover(fits, POP_{Parent})$
10:         $off_1 \leftarrow VNS(off_1)$
11:         $off_2 \leftarrow VNS(off_2)$
12:         $POP_{Offspring} \leftarrow POP_{Offspring} \cup \{off_1, off_2\}$
13:     end for
14:     $POP_{Parent} \leftarrow PDAS(POP_{Parent}, POP_{Offspring})$ //Population Diversity Adjustment Strategy
15: end while
16: $X^* \leftarrow argmax\{f(X) | X \in POP\}$
17: return $X^*$
```

4.2. Initializing the population

The initial individuals in the population are constructed using a greedy strategy: For the first individual in the population, the discounted profit density of each item is first calculated. All items are then sorted in descending order based on this profit density. Items are sequentially selected and placed into the knapsack according to this sorted order until the capacity limit is reached. The remaining individuals are initialized by randomly assigning items to the knapsack.

Definition 4.1. *(Profit density of the items' combination in a set) For the set E_i, $i = 1, 2, \cdots, n$, noting that $M_{i,1}, M_{i,2}, \cdots, M_{i,\Delta j}$ $(1 \le j \le r_i)$ denotes any combination of j items contained in the i-th set, Δj denotes the number of all combinations containing j items, so we have that $M_{i,q} \in \{M_{i,1}, M_{i,2}, \cdots, M_{i,\Delta j}\}$ holds, where $q \in \{1, 2, \cdots, \Delta j\}$; $d_{i,j}$ denotes the discount rate corresponding to the combination, and we define the profit density of the combination as:*

$$e^i_{j,q} = \frac{\sum_{k=1}^{j} p_{i,m_{q_k}}}{\sum_{k=1}^{j} \omega_{i,m_{q_k}} \cdot d_{i,j}}, i = 1, 2, \cdots, n, j = 1, 2, \cdots, r_i. \tag{4}$$

However, since the number of items to be selected for each set cannot be determined during initialization, $d_{i,j}$ cannot be derived directly, so we first use the unit profit of each item (Eq (5)) in the set to make the selection. Each item is sorted according to its unit profit, and items with higher unit profit are prioritized in the backpack. Since the total weight of the items that end up in the backpack is less than the capacity of the backpack without multiplying by the discount factor, we use Eqs (6) and (7) to relax the constraint of capacity.

$$e^i_{j,q} = \frac{\sum_{k=1}^{j} p_{i,m_{q_k}}}{\sum_{k=1}^{j} \omega_{i,m_{q_k}}}, i = 1, 2, \cdots, n, j = 1, 2, \cdots, r_i, \tag{5}$$

$$\sum_{i=1}^{n} \left(\sum_{j=1}^{r_i} w_{i,j} x_{i,j} \right) \leq \xi \cdot C, \tag{6}$$

$$\xi = \frac{2}{d_{max} + d_{min}}, \tag{7}$$

where $m_{q_k}(k = 1, 2, \cdots, j)$ denotes the k-th item of the q-th combination containing j items, $p_{i,m_{q_k}}$ denotes the profit of the k-th item of the q-th combination in the i-th set, $w_{i,m_{q_k}}$ denotes the weight of the k-th item of the q-th combination in the i-th set, ξ is the coefficient of relaxation, and C denotes the capacity of the backpack.

When producing the initial population, if the individuals in the entire population are produced by the greedy strategy described above, then all individuals in the population are exactly the same. To avoid this situation, we only produce one individual using this strategy, and the other individuals in the population are randomly obtained. Random individuals are likely to cause the population to exceed the capacity of the backpack and, for this reason, the repair rule is designed as follows.

4.3. Representation of solutions and the repair rule

For the set E_i, $i = 1, 2, \cdots, n$, the combination of items is represented by the binary vector $X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,r_i}) \in \{0, 1\}^{r_i}$ and the solution is denoted as $Y = (X_1, X_2, \cdots, X_n)$. Due to the stochastic nature of the swarm intelligence algorithm solution, for the ESD {0-1} KP-R problem, the binary encoding approach may produce infeasible solutions. This is because some backpacks do not have selected items or the total weight of the selected items exceeds the backpack's capacity. The main methods to deal with infeasible solutions are the penalty function method, repair method, pure method, and separation method [4, 31]. Michalewicz [32] pointed out that the repair method is more effective compared to the penalty function method, we adopt the repair method to deal with infeasible solutions. Currently, the repair method for infeasible solutions of the D {0-1} KP problem is mainly the greedy repair method based on the profit density. Considering the local search of the algorithmic framework as well as the stochastic nature of the solution, so we adopt the random replacement method for repair, i.e., first randomly selecting one set, and then randomly choosing a smaller total number of combinations to replace the current combination in this set until the total weight of the selected items is not more than the backpack's capacity C. The repair rule for infeasible solutions is shown in Algorithm 2.

Algorithm 2 Repair of infeasible solutions.

```
Input: The Infeasible Solution $Y = (X_1, X_2, \cdots, X_n)$
Output: The Fixed Feasible Solution $Y$
1: while the total weight of the selected items exceeds the load capacity of the backpack do
2:     randomly select a set $E_i$
3:     compute all non-empty subsets of the set $E_i$
4:     calculate the weight of each non-empty subset
5:     randomly select a combination $X_i^{new}$ from non-empty subsets with a smaller weight
6:     $X_i \leftarrow X_i^{new}$
7: end while
8: return $Y$
```

4.4. Crossover

We will now introduce the affiliation function in the crossover operation. The degree of the affiliation function is an important concept in fuzzy set theory, which is used to describe the degree to which an element belongs to a fuzzy set. This degree is a value between 0 and 1, where 0 means that the element does not belong to the set, 1 means that it belongs completely to the set, and a value between 0 and 1 indicates the degree of belonging. The set $A = (sc_1, \mu_A(sc_i)), (sc_2, \mu_A(sc_2)), \cdots, (sc_{pop_{size}}, \mu_A(sc_{pop_{size}}))$ in Eqs (8) and (9) represents the fuzzy set of elite individuals and the set $B = \{(sc_1, \mu_B(sc_i)), (sc_2, \mu_B(sc_2)), \cdots, (sc_{pop_{size}}, \mu_B(sc_{pop_{size}}))\}$ represents the fuzzy set of mediocre individuals. Eqs (8) and (9) are the affiliation functions of A and B, respectively. The computation of the affiliation degree is as follows (Algorithm 3), and the visualization of μ is shown in Figure 1.

$$\mu_A(sc_i) = \begin{cases} 0, 0 \le sc_i \le \dfrac{pop_{size}}{2} \\ \left[1 + \left(\dfrac{sc_i - \frac{pop_{size}}{2}}{5}\right)^{-2}\right]^{-1}, \dfrac{pop_{size}}{2} \le sc_i \le pop_{size} \end{cases} \quad i = 1, 2, \cdots, pop_{size}, \tag{8}$$

$$\mu_B(sc_i) = \begin{cases} 1, 0 \le sc_i \le \dfrac{pop_{size}}{4} \\ \left[1 + \left(\dfrac{sc_i - \frac{pop_{size}}{4}}{5}\right)^{2}\right]^{-1}, \dfrac{pop_{size}}{4} \le sc_i \le pop_{size} \end{cases} \quad i = 1, 2, \cdots, pop_{size}, \tag{9}$$

where pop_{size} denotes the size of the population and sc_i denotes the descending order of the fitness of individual i in the population.

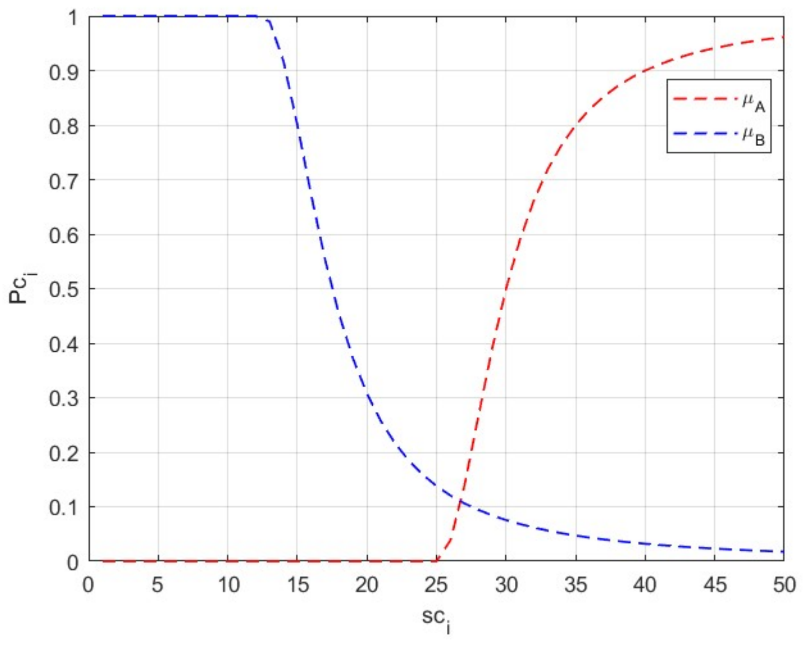


Figure 1. Membership functions $\mu_A(sc_i)$ and $\mu_B(sc_i)$. The functions map an individual's fitness rank to a membership degree in fuzzy sets A and B, which determine its selection probability in the improved memetic algorithm (IMA) algorithm.

It can be clearly seen that when the fitness of individual i is better, sc_i is smaller. When $\frac{pop_{size}}{2} \leq sc_i \leq pop_{size}$, the smaller sc_i is, the larger the value of $\mu_A(sc_i)$ is. When $\frac{pop_{size}}{4} \leq sc_i \leq pop_{size}$, the smaller sc_i is, the smaller the value of $\mu_B(sc_i)$ is.

In order to make full use of the gene fragments of the elite solutions, we designed Eq (10) to perform a comprehensive evaluation of each individual. A random number in $[-1, 1]$ is taken. When this random number is greater than Pc_i, we let this individual perform a crossover with the best fitness individual in the population. Conversely, we let it crossover with a random individual in the population (Algorithm 4). In this strategy, the initial solution obtained from the greedy initialization in Section 4.2 can also be fully exploited.

$$Pc_i = \mu_A(sc_i) - \mu_B(sc_i). \tag{10}$$

Obviously, this operation makes it easier for the better individuals to crossover with the best individual, speeding up the convergence efficiency. However, there is no loss of randomness in it, and it does not cause the population to converge prematurely.

The core of this adaptive strategy lies in dynamically assigning exploration and exploitation roles based on an individual's rank (sc_i) within the population. It implements a protection mechanism for the top 25% elite individuals ($sc_i \leq pop_{size}/4$), shielding them from exploratory disruptions and enabling deep exploitation of high-quality solution regions. The bottom 50% poorer-performing individuals ($sci \geq pop_{size}/2$) are assigned strong exploration tendencies, continuously injecting new genetic material into the population to fundamentally prevent premature convergence. Intermediate individuals achieve a dynamic balance between exploration and exploitation through smooth transition mechanisms. This design proactively allocates diversity production tasks, ensuring both the quantity of exploration and the quality of solutions, ultimately forming a healthy and sustainable ecological environment for population diversity.

Algorithm 3 Fuzzy.

```
Input: $sc_*$
Output: $Pc_*$
1: if $sc_* >= 0$ and $sc_*$¡= $pop_{size}/2$ then
2:     $\mu_A[sc_*] \leftarrow 0$
3: else
4:     Equation 8 was utilized to calculate the $\mu_A[sc_*]$
5: end if
6: if $sc_* >= 0$ and $sc_*$¡= $pop_{size}/4$ then
7:     $\mu_B[sc_*] \leftarrow 1$
8: else
9:     Equation 9 was utilized to calculate the $\mu_B[sc_*]$
10: end if
11: $Pc_* = \mu_A[sc_*] - \mu_B[sc_*]$
12: return $Pc_*$
```

Algorithm 4 Fuzzy-crossover.

```
Input: The Fitnesses of the Population $fits$, $POP$
Output: The offspring individuals obtained after crossover $off_1$, $off_2$
1: Descending ordering of $fits$
2: A randomly selected individual $p_1$ from $POP$
3: Get the index $sc_i$ of $p_1$ in $fits$
4: if $Fuzzy(sc_i) >=$ Random number between -1 and 1 then
5:     $off_1, off_2 = crossover(POP[inx], p_1)$
6: else
7:     $off_1, off_2 = crossover(p_1, p_2)$
8: end if
9: return $off_1$, $off_2$
```

4.5. Population diversity adjustment strategy

The main drawback of the MA is premature convergence, so in this study, we propose an adjustment strategy for population diversity to overcome this drawback. This strategy is able to improve the diversity of the population while ensuring that the solution has a good quality. The main factor contributing to the premature convergence of the algorithm is the decrease in the diversity of the population, for which it is necessary to adjust the diversity of the population during the iterations of the algorithm. The detailed pseudocode of this strategy is provided in Algorithm 6.

4.5.1. Delineation of the population

First, the diversity, or similarity, of a solution needs to be measured. We use the Hamming distance to measure the distance between two solutions, i.e., $g(u, v) = \sum |u_i - v_i|$. The smaller the distance, the more similar the two solutions are. To measure the similarity of a solution across the population, we define the solution-to-population distance as the average of the solution-to-population distances for each solution in the population (Eq (11)). It is easy to see that for a solution u, when the value of $g(u, POP)$ is smaller, then the diversity of this solution is worse.

$$D(u, POP) = \frac{1}{n} \sum_{i=1}^{n} g(u, v_i), POP = \{v_1, v_2, \cdots, v_n\}. \tag{11}$$

Next, we believe that all solutions in a population can be roughly categorized into four groups: (1) those with superior diversity and poor fitness; (2) those with superior fitness and poor diversity; (3) those with both superior fitness and diversity; and (4) those with both poor fitness and diversity. We use the two function values of $f(u)$ and $D(u, POP)$ to group the four sub-populations of the population, U_1, U_2, U_3, U_4. Corresponding to the four cases described above, for each sub-population, we select individuals according to different rules. This established strategy, introduced by Pisinger [33], effectively balances the exploration of diverse solutions with the exploitation of high-quality ones.

4.5.2. Information vector

When updating the population, we first merge the offspring population with the parent population and then divide the population according to Section 4.5.1, selecting individuals from U_1, U_2, U_3, U_4 according to different rules and putting them into the new population until its size is the same as the original population, selecting only one individual at a time. According to the information vector, the individuals from U_1, U_2 are selected, the individuals with better diversity are selected from U_3, and the individuals with better adaptation are selected from U_4. The information vector (IV) is constructed based on individual fitness, diversity, and remaining capacity of the backpack (Eqs (12)–(14)). The pseudo-code for this strategy is shown in Algorithm 5.

$$IV(u) = (1 - \alpha) \cdot Df(u) + \alpha \cdot (R_{max} - R(u)). \tag{12}$$

From Eq (12) we can see that the IV contains two components. The first component is $(1-\alpha)Df(u)$, where Df is detailed in Eq (13) and denotes the fitness-diversity score of individuals in the population. $f(u)$ and $D(u, POP)$ denote the fitness and diversity, respectively, of an individual u. $f(u)$ is represented by the objective function in Eq (3), and $D(u, U_i), i = \{1, 2\}$, is represented by Eq (11). In Eq (12) we normalize these two functions. It is clear that the value of Df is larger when both individual fitness and diversity are better. α is a constant of [0, 1].

$$Df(u) = \beta_t \cdot \frac{f(u) - f_{min}}{f_{max} - f_{min}} + (1 - \beta_t) \cdot \frac{D(u, U_i) - D_{min}}{D_{max} - D_{min}}, i = \{1, 2\}. \tag{13}$$

The second component on the right-hand side of Eq (12) is $\alpha \cdot (R_{max} - R(u))$, which normalizes the remaining capacity of the backpack corresponding to each individual (see Eq (14)).

$$R(u) = \frac{V(u) - V_{min}}{V_{max} - V_{min}}. \tag{14}$$

Algorithm 5 IV.

```
Input: $U^*$, the Current Number of Iterations $iter$ // $U^* = U_1, U_2$
Output: $IV$
1:  $w\_U_1 = \varnothing, fits\_U_1 = \varnothing$
2:  for $i$ in each $U^*$ do
3:      $w\_U_1[i], fits\_U_1[i] \leftarrow f(U_i[i])$
4:  end for
5:  $D(u, U^*)$ is obtained according to Eq (11)
6:  $V = \varnothing, R = \varnothing, IV = \varnothing, Df = \varnothing$
7:  for $j$ in each $U^*$ do
8:      $V(u)$ is obtained according to Eq (15)
9:  end for
10: for $k$ in each $U^*$ do
11:     $R(u)$ is obtained according to Eq (14)
12: end for
13: for $l$ in each $U^*$ do
14:     $Df(u)$ is obtained according to Eq (13)
15:     $IV(u)$ is obtained according to Eq (12)
16: end for
17: return $IV$
```

Equation 15 demonstrates the remaining capacity of the backpack, where C denotes the capacity of the backpack, n denotes the number of sets, r_i denotes the number of items in the i-th set, $x_{i,j}$ denotes the decision variable, and $d_{i,*}$ denotes the discount factor for the i-th set.

$$V(u) = C - \sum_{i=1}^{n} \left(\sum_{j=1}^{r_i} w_{i,j} x_{i,j} \cdot d_{i,*} \right). \tag{15}$$

In Eq (12) we arrange $IV(u)$ in descending order, and each time we select an individual with a smaller ordinal number from U_i to put into the new population. We can see that this individual is more likely to be selected if the score of the fitness-diversity $Df(u)$ is larger, and in order to keep the remaining capacity of the backpack in line with this, $R_{max} - R(u)$ is used in Eq (12).

During population renewal, elite individuals from both the offspring and parent gradually spread throughout the new population. This often leads to premature convergence of the algorithm. To solve this problem, we would like to ensure the diversity of the population in the early part of the iteration. At the same time, the convergence can be accelerated in the later part of the iteration [34]. Therefore, a linearly increasing coefficient β_t (Eq (16)) defined in Eq (13) is introduced to regulate the convergence rate of the algorithm.

$$\beta_t = \frac{t}{t_{max}}, \tag{16}$$

where t denotes the current number of iterations and t_{max} denotes the maximum number of iterations.

In the early stage of the iteration, we try to ensure the diversity of the population as much as possible without losing the stochasticity, so we use the information vector to select the best individuals. At the later stage of the iteration, in order to ensure the diversity and accelerate the convergence of the population, we directly select the individual with the largest fitness.

Algorithm 6 Population diversity adjustment strategy.

```
Input: Parent Population $POP_1$, Offspring Population $POP_2$
Output: The Updated Population $POP$
 1: $U \leftarrow POP_1 \cup POP_2$
 2: $POP \leftarrow argmax\{f(x)|x \in U\}$
 3: $U \leftarrow U - POP$
 4: while the total $|POP| < pop\_size$ do
 5:     $U_1 = \varnothing, U_2 = \varnothing, U_3 = \varnothing, U_4 \leftarrow \varnothing$
 6:     for each $x \in U$ do
 7:         if $f(x) \geq mean\{f(x)|x \in U\}$ and $g(x) \geq mean\{g(x)|x \in U\}$ then
 8:             $U_1 \leftarrow U_1 \cup \{x\}$
 9:         else if $f(x) < mean\{f(x)|x \in U\}$ and $g(x) \geq mean\{g(x)|x \in U\}$ then
10:             $U_2 \leftarrow U_2 \cup \{x\}$
11:         else if $f(x) \geq mean\{f(x)|x \in U\}$ and $g(x) < mean\{g(x)|x \in U\}$ then
12:             $U_3 \leftarrow U_3 \cup \{x\}$
13:         else
14:             $U_4 \leftarrow U_4 \cup \{x\}$
15:         end if
16:         if $t <= t_{max}/2$ then
17:             if $U_1 \neq \varnothing$ then
18:                 $temp \leftarrow argmax\{IV(x)|x \in U_1\}$
19:                 $U_1 \leftarrow U_1 - \{temp\}$
20:             else if $U_2 \neq \varnothing$ then
21:                 $temp \leftarrow argmax\{IV(x)|x \in U_2\}$
22:                 $U_2 \leftarrow U_2 - \{temp\}$
23:             else if $U_3 \neq \varnothing$ then
24:                 $temp \leftarrow argmax\{g(x)|x \in U_3\}$
25:                 $U_3 \leftarrow U_3 - \{temp\}$
26:             else
27:                 $temp \leftarrow argmax\{f(x)|x \in U_4\}$
28:             end if
29:             //Div
30:         else
31:             if $U_1 \neq \varnothing$ then
32:                 $temp \leftarrow argmax\{f(x)|x \in U_1\}$
33:                 $U_1 \leftarrow U_1 - \{temp\}$
34:             else if $U_2 \neq \varnothing$ then
35:                 $temp \leftarrow argmax\{f(x)|x \in U_2\}$
36:                 $U_2 \leftarrow U_2 - \{temp\}$
37:             else if $U_3 \neq \varnothing$ then
38:                 $temp \leftarrow argmax\{g(x)|x \in U_3\}$
39:                 $U_3 \leftarrow U_3 - \{temp\}$
40:             else
41:                 $temp \leftarrow argmax\{f(x)|x \in U_4\}$
42:             end if
43:         end if
44:         $POP \leftarrow POP \cup \{temp\}$
45:     end for
46: end while
```

4.6. Local search

In this work, we use the variable neighborhood search (VNS) algorithm to perform local search and design the neighborhood based on the profit density of the items' combinations (Definition 4.1).

We design the following three neighborhood structures for the ESD {0-1} KP-R, and the variable neighborhood search process is carried out sequentially in the order of N1 neighborhood, N2 neighborhood, and N3 neighborhood. The search range of the three neighborhood structures increases sequentially, which can effectively improve the algorithm's ability to find the optimal while avoiding falling into the local optimum.

(1) N1: Replace the current combination of the least profit-dense set with the combination with the highest profit density. This neighborhood improves the quality of the solution by replacing the combination with the minimum profit density, improving the algorithm's ability to find an better solution.

(2) N2: A set is randomly selected and the current combination is replaced with a combination of similar profit density (the difference in profit density between the new combination and the original combination falls within a certain threshold, which is set to 10 in this study). Since the optimal solution is likely to be located in the vicinity of the local optimum, this neighborhood helps the algorithm to continue the search in the vicinity of the local optimum.

(3) N3: A set is chosen at random and a combination is chosen at random to replace the current combination. This neighborhood may not help much to improve the quality of the solution but helps the algorithm to jump out of the local optimum.

The time complexity analysis is as follows: For the N1 operator, the operation requires traversing all n sets to locate the set containing the current combination with the minimum profit density, which has a time complexity of $O(n)$. Then, within that specific set, it traverses all k combinations to find the one with the maximum profit density, which has a time complexity of $O(k)$. Therefore, the overall time complexity of the N1 operator is $O(n + k)$. For the N2 operator, randomly selecting a set can be done in $O(1)$ time, and obtaining the profit density of the current combination in that set also requires $O(1)$ time. It then traverses all k combinations in the set to find a new combination whose profit density difference falls within a predefined threshold. This traversal process has a time complexity of $O(k)$. For the N3 operator, both randomly selecting a set and randomly choosing a combination within that set can be completed in $O(1)$ time.

Algorithm 7 Local search.

```
Input: Solution $X^*$
Output: $X^{new}$
1: $X^{new} = X^*$
2: $fit = f(X^*)$
3: $X\_n1 = neighborhood1(X^{new})$
4: if $f(X\_n1) > fit$ then
5:     $X^{new} = X\_n1$
6:     $fit = f(X\_n1)$
7: end if
8: $X\_n2 = neighborhood2(X^{new})$
9: if $f(X\_n2) > fit$ then
10:     $X^{new} = X\_n2$
11:     $fit = f(X\_n1)$
12: end if
13: $X\_n3 = neighborhood3(X^{new})$
14: if $f(X\_n3) > fit$ then
15:     $X^{new} = X\_n3$
16: end if
17: return $X^{new}$
```

4.7. Complexity analysis of the IMA

In this section, we analyze the overall time complexity of the IMA. For convenience, we assume that the total number of items is D, the population size is N, the maximal stopping condition is $Iter$, the number of individuals that need to use the IV is m, the length of IV is n, the number of individuals that crossover with the best individual is a, and the length of Pc is b. According to Section 4, the IMA improves the crossover and local search strategies in the framework of the MA, and adds greedy initialization and diversity adjustment strategies. By the description of the above section, we can easily see that the time complexity of its main body is $O(D * N)$, the time complexity of greedy initialization is $O(D)$, the time complexity of the diversity adjustment strategy is $O(m * n)$, and the time complexity of the crossover strategy increase is $O(a * b)$. Since the greedy strategy in the IMA is called only at initialization, the other strategies are used only in special cases. The time complexity of the IMA is $O(Iter*D*N)+O(D)+O(m*n))+O(a*b)$. Clearly, when the number of iterations is sufficient, the time complexity of the algorithm is approximately equal to $O(Iter * D * N)$. The space complexity of the algorithm is primarily determined by population storage. The algorithm needs to maintain a population of size N, with each individual having an encoding length of n, resulting in a space complexity of $O(N * n)$. The additional space required by other components, such as greedy initialization, diversity adjustment, and local search, is significantly smaller than the population storage overhead and does not affect the overall complexity order.

5. Experimental results and analysis

5.1. Description of experimental environment and test instances

The experiments were programmed in Python 3.12.1 (Numpy 1.26.3, Matplotlib 3.8.2), and the programs were run on an Arch Linux (Linux 6.7.2) operating system with a CPU of 12th Gen Intel i9-12900H @ 4.900GHz and RAM of 16G DDR5.

Four classes of ESD {0-1} KP-R instances are designed by borrowing the idea of generating experimental arithmetic for the D {0-1} KP problem from the literature [3]: uncorrelated instances of the ESD {0-1} KP-R (UESDKPR), weakly correlated instances of the ESD {0-1} KP-R (WESDKPR), strongly correlated instances of the ESD {0-1} KP-R (SESDKPR), inverse correlated instances of the ESD {0-1} KP-R (IESDKPR).

The weight coefficients W_i and profit coefficients P_i of the four types of ESD {0-1} KP-R instances are set as shown in Table 1. Here, $i = 1, 2, \cdots, \sum_{j=1}^{n} r_j$, and each coefficient is taken as a random integer on the corresponding interval. In order to make the instance parameters satisfy Eq (2), the discount coefficients for the combination of items are set as $d_{i,*} \in [0.6, 1]$. The maximum load of the backpack in this paper [33,35] is set as $C = \lfloor \min\{\Xi\} + \theta(\max\{\Xi\} - \min\{\Xi\}) \rfloor$, where $\theta \in [0.45, 0.75]$. It is easy to see that the instance size is determined by the number of sets n and the maximum number of set items r.

Table 1. ESD {0-1} KP-R instance weight factor and profit factor settings.

Instance	W_i	P_i
UESDKPR	$[2, 1000]$	$[2, 1000]$
WESDKPR	$[101, 1000]$	$[W_i - 100, W_i + 100]$
SESDKPR	$[2, 1000]$	$W_i + 100$
IESDKPR	$P_i + 100$	$[2, 1000]$

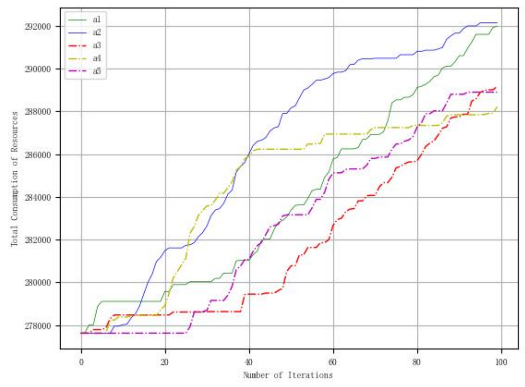
5.2. Parameter tuning

This section focuses on comparing the effects of the parameter involved in the information vector (i.e., α in Eq (12)). The population size of the algorithm is set to $pop_{size} = 50$, and the maximum number of iterations is set to $t_{max} = 100$. It is easy to see that the instance size is determined by the number of sets n and the maximum number of items in a set is r. Table 2 shows the optimal solutions obtained by the IMA on instance $20 * 6$ using different parameters, where the result denotes the optimal solution. Note that the parameter α determines the ratio between the fitness-diversity and the remaining backpack capacity. Thus, when α is larger, the proportion of the remaining backpack capacity in the information vector is more significant. On the contrary, if α is smaller, the proportion of fitness-diversity in the customer information vector becomes more significant. α needs a suitable value, otherwise the quality of the final solution will suffer. α is set to the following test values: $\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$.

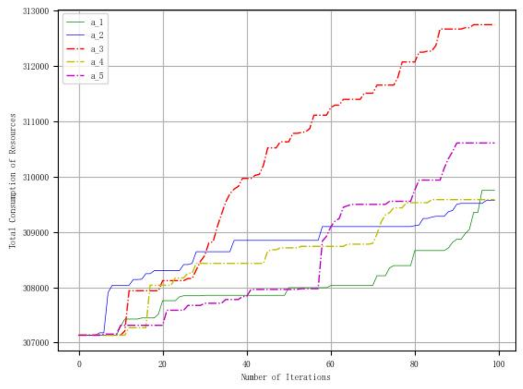
Table 2. Results with different parameters.

Experiment 1	α	Result	Experiment 2	α	Result
$\alpha 1$	0.1	291977	α_1	0.1	309753
$\alpha 2$	0.3	**292139**	α_2	0.2	309670
$\alpha 3$	0.5	289153	α_3	0.3	**312750**
$\alpha 4$	0.7	288208	α_4	0.4	309686
$\alpha 5$	0.9	288893	α_5	0.5	310610

First we take uniform values of α in $(0, 1)$ such that $\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$. After conducting one experiment (Experiment 1) it was found that the best results are obtained when $\alpha = 0.3$. The best value of α can be determined to be between 0.1 and 0.3. The second experiment was conducted to make the value of $\alpha = 0.3$ more convincing. To ensure the rigor of the experiment, we conducted a second experiment. The second experiment (Experiment 2) narrowed the spacing between each value taken so that $\alpha = 0.1, 0.2, 0.3, 0.4, 0.5$. The results of the two experiments are shown in Table 2 and Figure 2, which show that a taken as 0.3 gives the best solution to the problem. It is worth noting that to ensure that the results are not idiosyncratic, Experiment 1 and Experiment 2 are conducted as independent experiments of the same size ($n = 20$, $r = 6$), so the number of items in each set, profit coefficients, weight coefficients, and capacity of the backpack are randomly generated. The parameter tuning analysis, while revealing important patterns in algorithm behavior, would benefit from more rigorous statistical validation in future studies through confidence intervals and significance testing.



(1) $\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$.



(2) $\alpha = 0.1, 0.2, 0.3, 0.4, 0.5$.

Figure 2. Results with different parameters.

5.3. Sensitivity analysis of the components in the IMA

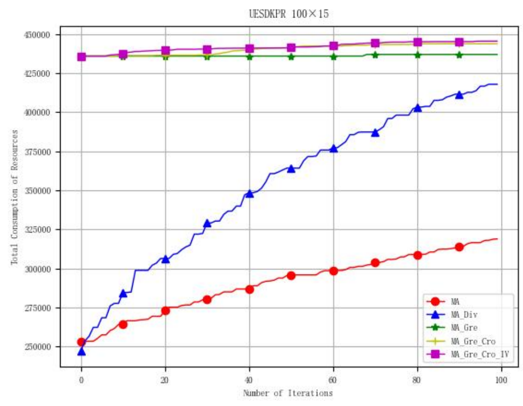
This section aims to clarify the impact of the components presented in the IMA. We attempt to do so by adding components one by one to the original MA. The strategy proposed in this paper focuses on speeding up convergence and maintaining population diversity.

In this work, we use a new initialization strategy, population updating strategies, and crossover strategy to accelerate the convergence of the IMA and to ensure the diversity of the population. The population size of the algorithm is set to $pop_{size} = 50$, and the maximum number of iterations is set to $t_{max} = 100$. It is easy to see that the instance size is determined by the number of sets n and the maximum number of items in each set is r. For this purpose, four algorithms are used as competitors of the IMA, i.e., "MA" does not contain new strategies, "MA_Div" means that the MA involves only diversity adjustment strategies for population division, "MA_Gre" means a greedy initialization strategy is added to "MA_Div", "MA_Gre_Cro" means a new crossover strategy is added to "MA_Gre", and "MA_Gre_Cro_IV " implies that an information vector has been added to the diversity adjustment strategy of "MA_Gre_Cro". In the table, "U","W","S","I" denote the four instances UESDKPR, WESDKPR, SESDKPR, IESDKPR, respectively. "res" denotes the results obtained by different algorithms with the same parameters. "CP" denotes the percentage difference between the solutions obtained by the IMA, MA, GA, and DPSO algorithms and the best solution obtained in this experiment under the same parameters. The smaller the value of the CP, the better the solution is; the larger the value of the CP, the larger the difference between the solution and the best solution is; and when the value of the CP is 0, this denotes that this solution is the best solution under the same parameters in the experiment.

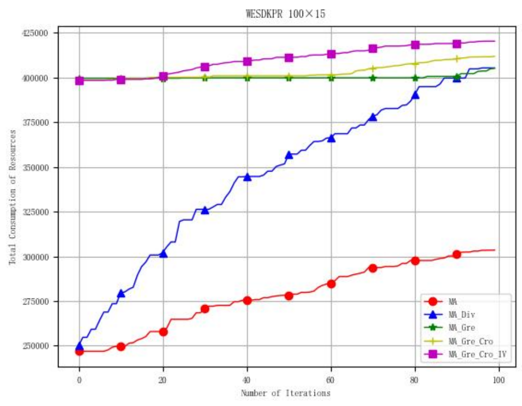
"Div" denotes the adjustment strategy of population diversity in Algorithm 5 without adding the information matrix. From Table 3 and Figure 3, it can be seen that the greedy initialization strategy and the adjustment strategy of population diversity designed in this paper play a very significant role in the solution effect of the algorithm. Each component added into the algorithm has its own optimization effect.

Table 3. Sensitivity analysis.

Instance	n	r	MA		MA_Div		MA_Gre		MA_Gre_Cro		MA_Gre_Cro_IV	
			res	CP/%	res	CP/%	res	CP/%	res	CP/%	res	CP/%
U	50	8	130233	19.06	128123	1.64	128413	1.41	128974	0.97	**130233**	0
W	50	8	114185	10.79	124192	1.86	124077	1.96	125862	0.51	**126512**	0
S	50	8	115231	14.41	130316	1.16	129390	1.89	130852	0.75	**131836**	0
I	50	8	118015	17.04	137025	0.80	136302	1.33	135421	1.99	**138125**	0
U	50	11	128947	23.37	153906	3.36	158143	0.59	159017	0.04	**159082**	0
W	50	11	145445	17.22	166261	2.54	167940	1.51	167805	1.60	**170491**	0
S	50	11	168406	0.56	168406	7.85	166973	1.42	166344	1.81	**169356**	0
I	50	11	137941	11.07	152004	0.80	**153221**	0	150286	1.95	149965	2.17
U	100	15	318968	39.66	417888	6.60	436980	1.94	443833	0.36	**445471**	0
W	100	15	303505	38.50	405376	3.69	405206	3.73	411904	2.05	**420355**	0



(1)



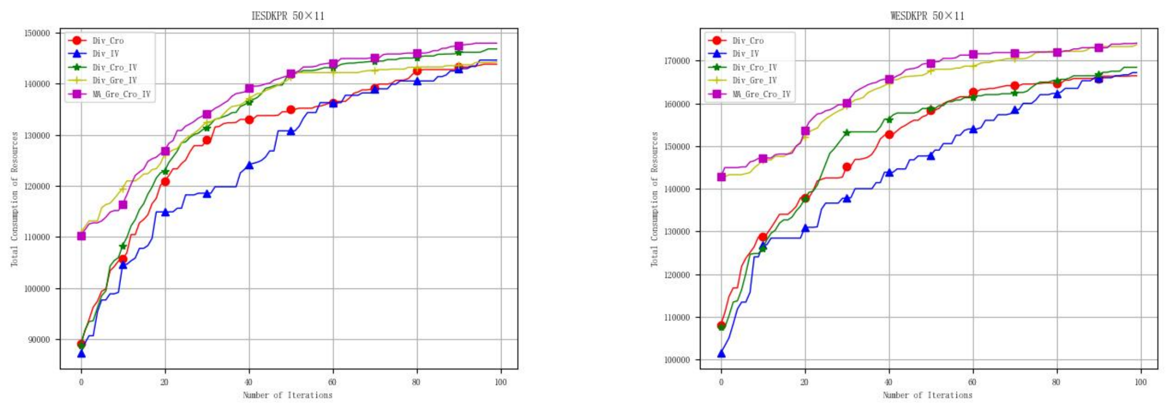
(2)

Figure 3. Sensitivity analysis of different policy components added in sequence.

Figure 3 indicates that the population diversity operation yields a significantly better optimization effect on the ESD {0-1} KP-R problem compared to the MA algorithm. To further evaluate the additional contributions of modules such as the crossover operation and information matrix, various combinations of these modules were tested based on population diversity, with the results presented in Figure 4 and Table 4. The abbreviations are defined as follows: "Div_Cro" refers to diversity adjustment strategies plus a crossover operation; "Div_IV" refers to diversity adjustment strategies plus an information vector; "Div_Cro_IV" refers to diversity adjustment strategies plus a crossover operation plus an information matrix; and "Div_Gre_IV" refers to diversity adjustment strategies plus greedy initialization plus an information vector.

Table 4. Sensitivity analysis.

Instance	n	r	Div_Cro		Div_IV		Div_Cro_IV		Div_Gre_IV		MA_Gre_Cro_IV	
			res	CP/%	res	CP/%	res	CP/%	res	CP/%	res	CP/%
U	50	8	122720	1.76	121668	2.64	124530	0.28	123736	0.01	124882	0
W	50	8	132486	2.62	131357	3.50	135955	0	132932	0.02	133581	1.78
S	50	8	148596	0.99	146342	2.54	148930	0.76	149847	0.14	150061	0
I	50	8	122935	0.99	121875	1.87	124158	0	121644	0.02	120201	3.29
U	50	11	151650	5.57	153752	4.12	155834	2.73	159076	0.01	160093	0
W	50	11	166488	4.59	167222	4.13	168468	3.36	173780	0.00	174125	0
S	50	11	170279	0.67	168049	2.01	169101	1.37	170210	0.01	171421	0
I	50	11	143832	2.86	144631	2.29	146823	0.77	144265	0.03	147949	0
U	100	15	376903	12.92	388412	9.57	365588	16.41	425588	0	420201	1.28
W	100	15	410224	1.16	414389	0.14	407677	1.79	402818	0.03	414987	0



(1) (2)

Figure 4. Sensitivity analysis on other components conducted under the premise of adding the population diversity operation.

5.4. Comparative analysis of different algorithm experiments

In this work, we mainly focus on the defect of premature convergence of the MA by designing greedy initialization, a new crossover strategy, and a population diversity adjustment strategy to improve the MA. Then, we design a local search algorithm for the characteristics of the ESD {0-1} KP-R problem. In order to verify the effectiveness of these four parts, this section will analyze the improved algorithm IMA in comparison with the MA, GA, and DPSO through experiments. The specific parameter settings are shown in Table 5. The crossover probability and mutation probability of the GA are set to 0.8 and 0.02, respectively, the population size of the algorithm is set to $pop_{size} = 50$, and the maximum number of iterations is set to $t_{max} = 100$. The maximum iteration number of the standard MA is 100. DPSO incorporates the core ideas of particle swarm optimization through probabilistic rules. Specifically, with an 0.8 probability, each individual undergoes crossover operations with its personal historical best and the global historical best, equivalent to setting $c_1 = c_2 = 0.8$, thereby simulating cognitive and social learning processes. Simultaneously, it introduces a 0.02 mutation probability to replace the inertia weight, equivalent to $w^{DPSO} = 0.98$, maintaining population diversity. This approach ingeniously adapts the particle flight mechanism from continuous space to combinatorial optimization problems.

The instance size is determined by the number of sets n and the maximum number of items in each set is r. Table 6 shows the optimal solutions obtained by solving four types of ESD {0-1} KP-R problems under different algorithmic conditions. In the table, CP denotes the percentage difference between the solutions obtained by the IMA, MA, GA, and DPSO algorithms and the best solution obtained in this experiment under the same parameters. The smaller the value of the CP is, the better the solution is; the larger the value of the CP is, the larger the difference between the solution and the optimal value is; and when the value of the CP is 0, it denotes that the corresponding solution is the optimal solution under the same parameters in this experiment.

Table 5. Parameter settings of all algorithms.

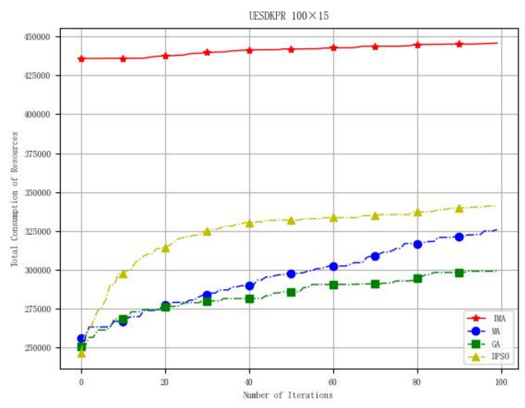
Algorithm	Parameter	Value
GA	Crossover Probability	0.8
	Mutation Probability	0.02
MA	Crossover Probability	0.8
	Mutation Probability	0.02
	Local Search Iterations	100
DPSO	Inertia Weight (w)	0.98
	Cognitive Coefficient (c_1)	0.8
	Social Coefficient (c_2)	0.8
Our Method	Crossover Probability	0.8
	Mutation Probability	0.02
	Local Search Iterations	100

From Table 6 and Figure 5, it can be seen that as the scale gradually increases, the optimization effect of the IMA algorithm in this paper is more and more obvious compared with other algorithms. When the scale is $n = 20$, $r = 6$, the maximum CP value is 16.33%; when the scale is $n = 20$, $r = 8$, the maximum CP value is 16.08%; when the scale is $n = 20$, $r = 10$, the maximum CP value is 23.37%; when the scale is $n = 50$, $r = 4$, the maximum CP value is 15.30%; when the scale is $n = 50$, $r = 8$, the maximum CP value is 28.42%; when the scale is $n = 50$, $r = 11$, the maximum CP value is 32.57%; when the scale is $n = 100$, $r = 10$, the maximum CP value is 40.34%; when the scale is $n = 100$, $r = 12$, the maximum CP value is 46.74%; and when the scale is $n = 100$, $r = 15$, the maximum CP value is 49.04%. It can be seen that as the size increases, the CP value is getting bigger and bigger. Especially in the scale of $n = 100$, $r = 15$, the optimal solution obtained by the IMA is at least 2.16% more than the other algorithms in the class of UESDKPR, at least 1.05% more in the class of WESDKPR, at least 2.90% more in the class of SESDKPR, and at least 1.21% more in the class of IESDKPR. After analysis, it is known that the IMA algorithm has a more obvious optimization effect in both small-scale and large-scale ESD {0–1} KP-R problems. After several experiments, we find that the IMA algorithm proposed in this paper outperforms the other three basic algorithms 100% in terms of solution results.

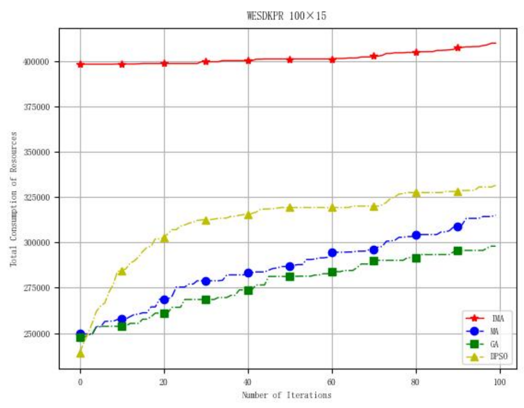
To quantify the performance advantage of the algorithm proposed in this study, we employed a paired sample t-test for statistical significance analysis. The test results demonstrate that our algorithm achieves exceptionally significant performance improvements compared to the baseline algorithms. Specifically, when compared to the MA algorithm, the results of our method are higher by an average of 7635.06 units, a difference that is highly statistically significant ($t(35) = 7.94, p < 0.001$). The advantage is even more pronounced against the GA algorithm, with an average improvement of 12438.89 units ($t(35) = 10.35, p < 0.001$). Even when compared to the relatively strong-performing DPSO algorithm, our method maintains a significant edge, yielding results that are higher by an average of 3459.89 units ($t(35) = 4.05, p = 0.0003$). In summary, the statistical tests provide compelling evidence that the algorithm proposed in this study consistently and significantly outperforms all baseline algorithms (MA, GA, and DPSO), and its superiority is not coincidental but is supported by a solid statistical foundation.

Table 6. Comparison of experimental results for different algorithms.

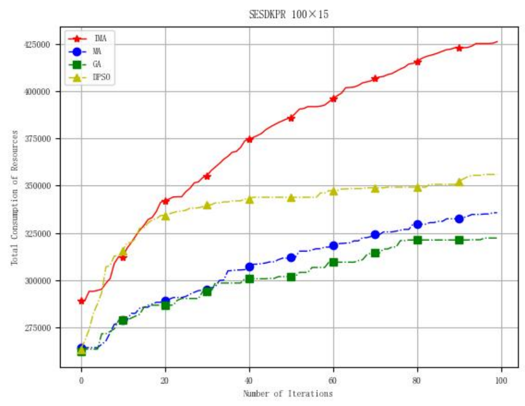
Instance	*n*	*r*	IMA		MA		GA		DPSO	
			res	CP/%	res	CP/%	res	CP/%	res	CP/%
U	20	6	**50444**	0	47262	6.73	45189	11.62	49143	2.64
W	20	6	**44391**	0	42509	4.42	40235	10.32	42160	5.29
S	20	6	**52261**	0	50787	2.90	44922	16.33	49839	4.85
I	20	6	**38362**	0	37858	1.33	34310	11.80	37634	1.93
U	20	8	**51822**	0	47751	8.52	46291	11.94	50591	2.43
W	20	8	**57449**	0	53956	6.47	52234	9.98	56852	1.05
S	20	8	**63604**	0	60085	5.85	58248	9.19	60737	4.72
I	20	8	**50843**	0	47833	6.29	43798	16.08	47506	7.02
U	20	10	**70004**	0	59172	18.30	57818	21.07	67611	3.53
W	20	10	**69769**	0	63193	10.40	64153	8.75	68312	2.13
S	20	10	**78281**	0	72504	7.96	63451	23.37	73191	6.95
I	20	10	**71054**	0	64822	9.61	61498	15.53	67240	5.67
U	50	4	**75786**	0	67713	11.92	68354	10.87	72486	4.55
W	50	4	**78752**	0	75723	4.00	71842	9.61	77906	1.08
S	50	4	**101902**	0	94013	8.39	88378	15.30	95494	6.71
I	50	4	**69128**	0	65483	5.56	60662	13.95	68297	1.21
U	50	8	**130493**	0	109217	19.48	103678	25.86	115225	13.25
W	50	8	**125822**	0	109553	14.85	102760	22.44	115195	9.22
S	50	8	**132209**	0	116277	13.70	102946	28.42	117064	12.93
I	50	8	**135331**	0	118565	14.14	109281	23.83	116402	16.26
U	50	11	**159059**	0	129428	22.89	122468	29.87	134735	18.05
W	50	11	**169111**	0	139841	20.93	127556	32.57	138971	21.68
S	50	11	**168100**	0	152191	10.45	152628	10.13	159688	5.26
I	50	11	**151985**	0	143131	6.18	127036	19.63	141050	7.75
U	100	10	**304478**	0	229264	32.80	216954	40.34	236857	28.54
W	100	10	**273889**	0	249108	9.94	247995	10.44	259316	5.61
S	100	10	**300845**	0	259376	15.98	229896	30.86	266115	13.05
I	100	10	**250326**	0	226786	10.37	212733	17.67	232100	7.85
U	100	12	**339256**	0	253006	34.09	231182	46.74	265771	27.64
W	100	12	**324449**	0	256854	26.31	243127	33.44	288264	12.55
S	100	12	**353174**	0	297682	18.64	270191	30.71	303096	16.52
I	100	12	**310181**	0	261622	18.56	234362	32.35	267017	16.16
U	100	15	**445732**	0	325898	36.77	299067	49.04	341127	30.66
W	100	15	**410055**	0	315132	30.12	297993	37.60	331596	23.66
S	100	15	**426139**	0	335708	26.93	322336	32.20	355944	19.72
I	100	15	**380723**	0	299332	27.19	276204	37.84	308310	23.48



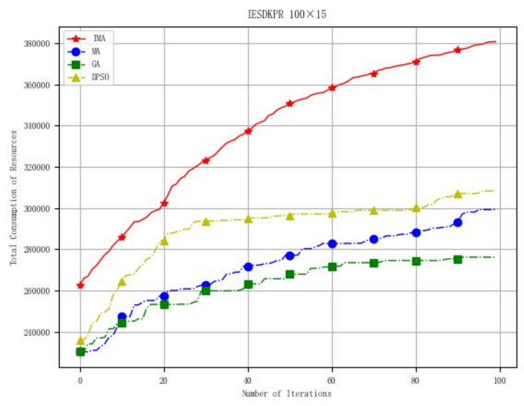
(1)



(2)



(3)

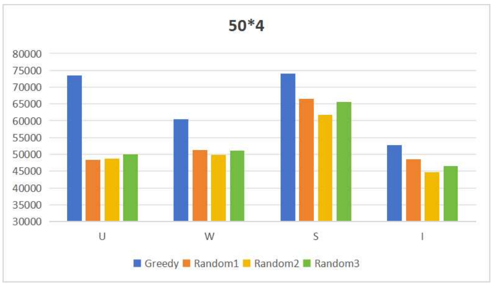


(4)

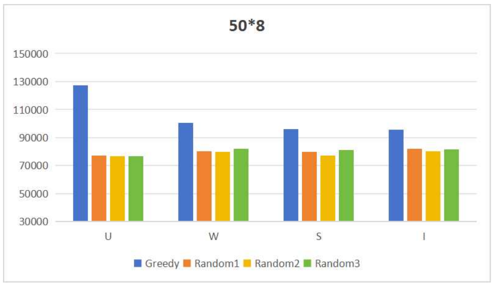
Figure 5. Comparison of optimization results from different algorithms.

5.5. Comparison of initial solutions

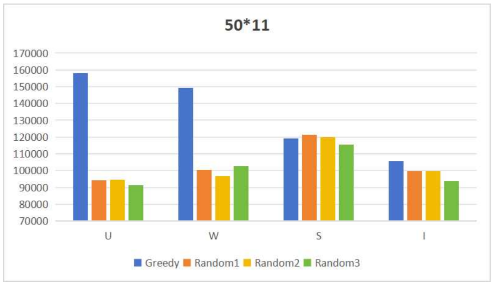
While comparing our IMA algorithm with other algorithms we compare the initial solutions of different algorithms. The effectiveness of the initialization strategy in this paper is further verified. Since the initial solutions of other algorithms are randomly generated and are likely to exceed the knapsack capacity, then the produced solutions seem to be good, but in fact they are infeasible solutions. It is necessary to adopt a repair strategy to obtain a feasible initial solution. However, due to the randomness in the repair process, the impact of this randomness on the results requires further in-depth discussion. Therefore in this paper we use the result of the first iteration to do the comparison of the initial solution. In the following, we use a total of 36 experiments with four different instances of UESDKPR, WESDKPR, SESDKPR, and IESDKPR, and 9 different sizes, to verify the effect of the greedy initialization in this paper.



(1)



(2)



(3)

Figure 6. Initial solutions with different parameters.

In the initialization experiments we can see that the initial solutions obtained after the greedy initialization designed in this paper outperform the vast majority of randomly generated initial solutions. From Figure 6 and Table 7, it can be seen that with the gradual increase of the scale, the initial solution gap of the IMA algorithm in this paper is getting bigger and bigger compared with other algorithms. In the presented results, except where $n = 50$, $r = 11$, the SESDKPR instance, when the scale is $n = 20$, $r = 6$, the maximum value of the CP is 45.21%; when the scale is $n = 20$, $r = 8$, the maximum value of the CP is 42.74%; and when the scale is $n = 20$, $r = 10$, the maximum value of CP is 62.61%. When the scale is $n = 50$, $r = 4$, the maximum value of the CP is 52.09%; when the scale is $n = 50$, $r = 8$, the maximum CP value is 65.94%; when the scale is $n = 50$, $r = 11$, the maximum CP value is 73.06%; when the scale is $n = 100$, $r = 10$, the maximum CP value is 76.94%; when the scale is $n = 100$, $r = 12$, the maximum CP value is 74.53%; and when the scale is $n = 100$, $r = 15$, the maximum CP value is 76.97%. It can be seen that as the scale increases, the CP value is getting larger. In particular, at the scale of $n = 100$, $r = 15$, the initial solution obtained by the IMA is at least 37.40% more than that obtained by other algorithms for UESDKPR, 12.87% more than that obtained by other algorithms for WESDKPR, 0.47% more than that obtained by other algorithms for SESDKPR, and 1.37% more than that obtained by other algorithms for IESDKPR. After analysis, it can be seen that the greedy initialization strategy in this paper has more obvious optimization effects in both small-scale and large-scale ESD {0–1} KP-R problems. After many experiments, we find that the IMA algorithm proposed in this paper outperforms the other three basic algorithms by 97.22% in terms of the initial solution.

Table 7. Comparison of initial solutions for different algorithms.

Instance	n	r	IMA		MA		GA		DPSO	
			res	CP/%	res	CP/%	res	CP/%	res	CP/%
U	20	6	**44597**	0	32457	37.40	31991	39.40	30712	45.21
W	20	6	**37441**	0	30359	23.32	30364	23.30	28107	33.20
S	20	6	**40548**	0	34218	18.49	32492	24.79	35191	15.22
I	20	6	**34733**	0	26249	32.32	24774	40.19	25646	35.43
U	20	8	**47718**	0	33664	42.74	32724	45.81	33734	41.45
W	20	8	**44571**	0	39630	12.46	36522	22.03	40244	10.75
S	20	8	**43670**	0	40178	8.69	39318	11.06	41696	4.73
I	20	8	**40978**	0	32072	27.76	30223	35.58	31300	30.92
U	20	10	**68435**	0	45803	51.59	41909	65.68	42700	62.61
W	20	10	**58506**	0	50273	16.37	51288	14.07	50164	16.62
S	20	10	**64278**	0	50961	26.13	49910	28.78	46829	37.26
I	20	10	**47591**	0	45447	4.71	46088	3.26	42477	12.03
U	50	4	**73471**	0	48307	52.09	48790	50.58	49913	47.19
W	50	4	**60409**	0	51289	17.78	49839	21.20	51017	18.40
S	50	4	**74012**	0	66545	11.22	61789	19.78	65520	12.96
I	50	4	**52822**	0	48533	8.83	44724	18.10	46587	13.38
U	50	8	**127288**	0	77057	65.18	76850	65.63	76706	65.94
W	50	8	**100434**	0	80406	24.90	79776	25.89	82088	22.34
S	50	8	**96269**	0	79965	20.3	76928	25.14	80912	18.97
I	50	8	**95573**	0	81766	16.88	80230	19.12	81459	17.32
U	50	11	**157994**	0	94076	67.94	94498	67.19	91294	73.06
W	50	11	**149382**	0	100528	48.59	96713	54.45	102480	45.76
S	50	11	119159	1.83	**121349**	0	120056	1.07	115687	4.89
I	50	11	**105515**	0	99742	5.78	99850	5.67	93772	12.52
U	100	10	**301568**	0	173335	73.97	170429	76.94	174648	72.67
W	100	10	**227842**	0	201854	12.87	190374	19.68	195345	16.63
S	100	10	**204164**	0	188564	8.27	187435	8.92	192374	6.12
I	100	10	**171022**	0	166453	2.74	166818	2.52	168561	1.46
U	100	12	**333859**	0	194223	71.89	191285	74.53	194726	71.45
W	100	12	**300835**	0	203106	48.11	202548	48.52	200972	49.69
S	100	12	**236778**	0	230296	2.81	235651	0.47	228597	3.57
I	100	12	**207964**	0	205135	1.37	192651	7.94	187771	10.75
U	100	15	**435937**	0	255789	70.42	250579	73.97	246326	76.97
W	100	15	**398463**	0	249551	59.67	247554	60.96	238876	66.80
S	100	15	**289318**	0	264411	9.41	262234	10.32	263606	9.75
I	100	15	**251077**	0	234618	7.01	241377	4.01	238098	5.45

The time complexity of the greedy initialization strategy is $O(n \log n)$, primarily arising from the sorting operation of items. For the repair strategy following random initialization, each iteration of the repair process involves randomly selecting only one item for removal. This makes the cost of a single repair iteration constant, $O(1)$, while the worst-case scenario requires $O(n)$ iterations, resulting in a total repair cost of $O(n)$. Since the strategy employed in this paper applies the greedy initialization to only one individual in the population (to avoid multiple identical individuals) and uses random generation plus repair for all other individuals, the time complexity of the proposed initialization strategy is $O(n * \log n) + O((N-1) * n)$, assuming a population size of N. In contrast, the initialization strategy of other algorithms has a time complexity of $O(N * n)$. From the perspective of asymptotic time complexity, when the population size N is fixed, the time complexity of both strategies simplifies to $O(n * \log n)$ (because the $O(N * n)$ term is dominated by $O(n * \log n)$). When the problem size n is fixed, the complexity of both strategies simplifies to $O(N)$. Therefore, the two strategies are approximately equivalent in terms of scalability.

5.6. Ablative analysis of components in a local search

The detailed implementation, search mechanisms, and a discussion on the computational cost differences of the various neighborhood operators utilized in the local search phase are thoroughly presented in Section 4.6.

Analyzing Table 8 and Figure 7, it can be seen that the three kinds of neighborhood operators N1, N2, N3 used in this paper are more optimal than the solutions obtained by the other six combinations under the same conditions. Therefore, the effectiveness of the neighborhood operation designed in this paper is verified by ablation experiments.

Table 8. Ablative analysis of neighborhood operators.

Instance	n	r	N1	N2	N3	Iterations	Result	CP/%
U	50	11	1	1	1	100	**158134**	0
U	50	11	1	1	0	100	157989	0.09
U	50	11	0	1	1	100	156456	1.07
U	50	11	1	0	1	100	157870	0.16
U	50	11	1	0	0	100	157653	0.30
U	50	11	0	1	0	100	156556	1.00
U	50	11	0	0	1	100	156995	0.72

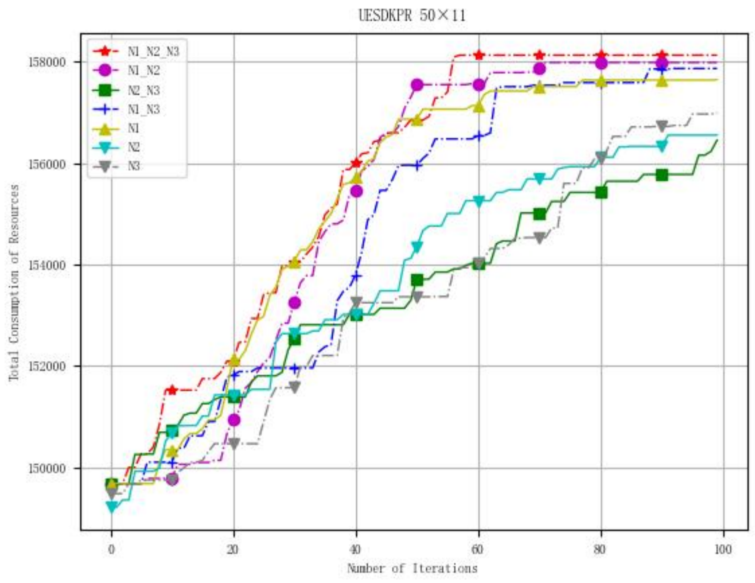


Figure 7. Ablation analysis and comparison of neighborhood strategies.

To validate the effectiveness of executing the neighborhood operations sequentially in the order of N1, N2, and N3, we rearranged the search sequences of the three neighborhood operators and conducted comparative experiments. The experimental results (see Figure 8 and Table 9) demonstrate that the algorithm achieves optimal performance when the execution order of the neighborhood operators is N1 → N2 → N3.

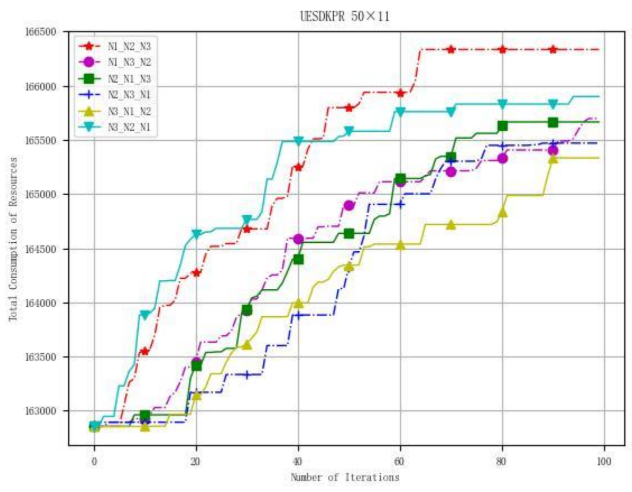


Figure 8. Comparison of optimization results for different operator orders.

Table 9. Optimization results for different operator orders.

Instance	n	r	Order	Iterations	Result	CP/%
U	50	11	N1-N2-N3	100	**158134**	0
U	50	11	N1-N3-N2	100	166335	0.39
U	50	11	N2-N1-N3	100	165697	0.40
U	50	11	N2-N3-N1	100	165666	0.52
U	50	11	N3-N1-N2	100	165471	0.61
U	50	11	N3-N2-N1	100	165901	0.26

6. Conclusions and future research

In this paper, we present the extended simplified discounted {0-1} knapsack problem (ESD {0-1} KP-R) model with stochasticity. Compared with the ordinary ESD {0-1} KP model, it increases the size and stochasticity of the problem, which is more suitable for the actual concept of "discounting" and has stronger generalization. Then, we use an improved memory algorithm to solve the ESD {0-1} KP-R model. First, a greedy strategy with slack constraints is designed to obtain the initial solution. Second, a crossover strategy with fuzzy sets is designed to generate a good population of offspring. Third, in order to overcome the disadvantage that memory algorithms tend to fall into local optimization, a population diversity adjustment strategy with information vectors is designed in this paper. This strategy combines the parent and child populations into a set of candidate solutions, and then divides all the solutions in the set into four sub-populations based on the fitness and diversity of the solutions, using different selection methods to adjust the diversity of the populations while ensuring the quality of the selected solutions. Next, three neighborhood structures are designed based on the profit density of term combinations. They are used to improve the search capability of the algorithm, to explore the neighborhood of local solutions, and to jump out of the local optimum, respectively. The local search is performed by the variable neighborhood search algorithm. Finally, the effectiveness of the proposed improvement strategy and the feasibility of the proposed algorithm in solving the ESD {0-1} KP-R are experimentally demonstrated.

In the future, other intelligent algorithms can be explored to solve the ESD {0-1} KP-R proposed in this paper to find a more suitable solution for this problem. As larger problem instances require longer computation times, we will continue in-depth research aimed at optimizing the efficiency of key components (such as initialization, crossover operators, and local search strategies) to significantly reduce the solving time for large-scale ESD {0-1} KP-R problems. First, exploring more refined metrics for measuring solution structural diversity is a critical direction. The current Hamming distance metric does not fully account for the structural information of item grouping within solutions. Future work could investigate weighted Hamming distance, edit distance, or graph-based similarity metrics to more accurately capture the structural properties of the solution space, potentially leading to the design of more effective diversity maintenance strategies. Additionally, the impact of different classification thresholds (e.g., median or standard deviation) used in conjunction with these new metrics warrants further investigation. This could potentially lead to the design of more effective diversity maintenance strategies. Second, future work could include a systematic sensitivity analysis of instance generation parameters (such as θ) to gain a deeper understanding of their impact on algorithm performance.

Author contributions

Zhouxi Qin: Conceptualization, software, data curation, writing—original draft, visualization. Dazhi Pan: Conceptualization, methodology, validation, formal analysis, writing—review and editing, supervision, project administration, funding acquisition. Ke Yang: Conceptualization, software, data curation, writing—original draft, visualization. Dapeng Gao: Funding acquisition, project administration, supervision, writing—review and editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 11871059), Natural Science Foundation of Sichuan Education Department (Grant No. 18ZA0469), Sichuan Science and Technology Program (Grant No. 2022NSFSC0558), and Research Project on Graduate Education Reform of China West Normal University (Grant Nos. 2022XM24, 2024XM05).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. H. Kellerer, U. Pferschy, D. Pisinger, *Knapsack problems*, Berlin, Heidelberg: Springer, 2004. https://doi.org/10.1007/978-3-540-24777-7
2. J. Guder, *Discounted knapsack problems for pairs of items*, Nuremberg: University of Erlangen-Nurnberg, 2005.
3. A. Rong, J. R. Figueira, K. Klamroth, Dynamic programming based algorithms for the discounted {0–1} knapsack problem, *Appl. Math. Comput.*, **218** (2012), 6921–6933. https://doi.org/10.1016/j.amc.2011.12.068
4. Y. C. He, X. Z. Wang, W. B. Li, X. L. Zhang, Y. Y. Chen, Research on genetic algorithms for the discounted {0–1} knapsack problem, *Chin. J. Comput.*, **39** (2016), 2614–2630. https://doi.org/10.11897/SP.J.1016.2016.02614
5. Z. Dai, B. Zhou, Y. Long, Z. Wang, Greedy repair strategy of particle swarm optimization for discounted {0–1} knapsack problem, *Appl. Res. Comput.*, **29** (2022), 2363–2368. https://doi.org/10.19734/j.issn.1001-3695.2021.12.0701
6. Y. Yang, D. Z. Pan, Y. Li, D. L. Tan, New simplified model of discounted {0–1} knapsack problem and solution by genetic algorithm, *J. Comput. Appl.*, **39** (2019), 656–662. https://doi.org/10.11772/j.issn.1001-9081.2018071580

7. Q. Zhang, D. Z. Pan, Modeling of extended SD {0–1} KP backpack problem and solution of genetic algorithm, *J. China West Norm. Univ. Nat. Sci.*, **42** (2020), 214–220. https://doi.org/10.16246/j.issn.1673-5072.2020.02.015

8. H. Lin, Y. Deng, Improved greedy algorithm to solve extended simplified discounted {0–1} knapsack problem, *J. Southwest China Norm. Univ. Nat. Sci. Ed.*, **47** (2022), 63–71. https://doi.org/10.13718/j.cnki.xsxb.2022.11.009

9. L. M. Schmitt, Theory of genetic algorithms, *Theor. Comput. Sci.*, **259** (2001), 1–61. https://doi.org/10.1016/S0304-3975(00)00406-0

10. G. Rudolph, Convergence analysis of canonical genetic algorithms, *IEEE Trans. Neural Netw.*, **5** (1994), 96–101. https://doi.org/10.1109/72.265964

11. P. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, *Caltech Concurrent Comput. Program C3P Rep.*, **826** (1989), 37.

12. H. Duan, X. Zhang, C. Xu, *Biomimetic intelligent computing*, Science Press, 2011.

13. B. Peng, Z. Lü, T. C. E. Cheng, A tabu search/path relinking algorithm to solve the job shop scheduling problem, *Comput. Oper. Res.*, **53** (2015), 154–164. https://doi.org/10.1016/j.cor.2014.08.006

14. C. Wilbaut, R. Todosijević, S. Hanafi, A. Fréville, Variable neighborhood search for the discounted {0–1} knapsack problem, *Appl. Soft Comput.*, **131** (2022), 109821. https://doi.org/10.1016/j.asoc.2022.109821

15. M. Črepinšek, S. H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: A survey, *ACM Comput. Surv.*, **45** (2013), 1–33. https://doi.org/10.1145/2480741.2480752

16. X. Tian, D. Ouyang, Y. Wang, H. Zhou, L. Jiang, L. Zhang, Combinatorial optimization and local search: A case study of the discount knapsack problem, *Comput. Electr. Eng.*, **105** (2023), 108551. https://doi.org/10.1016/J.COMPELECENG.2022.108551

17. H. Zhu, Y. He, X. Wang, E. C. Tsang, Discrete differential evolutions for the discounted {0–1} knapsack problem, *Int. J. Bio-Inspired Comput.*, **10** (2017), 219–238. https://doi.org/10.1504/IJBIC.2017.087924

18. Y. Feng, G. G. Wang, S. Deb, M. Lu, X. J. Zhao, Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization, *Neural Comput. Appl.*, **28** (2017), 1619–1634. https://doi.org/10.1007/s00521-015-2135-1

19. Y. Yang, D. Z. Pan, Y. C. He, Improved repair strategy genetic algorithm solve discount {0–1} knapsack problem, *Comput. Eng. Appl.*, **54** (2018), 37–42. https://doi.org/10.3778/j.issn.1002-8331.1711-0255

20. X. J. Liu, Y. C. He, F. J. Lu, C. C. Wu, X. F. Cai, Chaotic crow search algorithm based on differential evolution strategy for solving discount {0–1} knapsack problem, *J. Comput. Appl.*, **38** (2018), 137–145. https://doi.org/10.11772/j.issn.1001-9081.2017061445

21. Y. C. He, X. Z. Wang, S. G. Gao, Ring theory-based evolutionary algorithm and its application to D{0–1} KP, *Appl. Soft Comput. J.*, **77** (2019), 714–722. https://doi.org/10.1016/j.asoc.2019.01.049

22. X. P. Xu, L. Xu, F. Wang, L. Liu, Learning monkey algorithm based on lagrange interpolation to solve discounted {0–1} knapsack problem, *J. Comput. Appl.*, **40** (2020), 3113–3118. https://doi.org/10.11772/j.issn.1001-9081.2020040482

23. C. C. Wu, J. L. Zhao, Y. H. Feng, M. Lee, Solving discounted {0–1} knapsack problems by a discrete hybrid teaching-learning-based optimization algorithm, *Appl. Intell.*, **50** (2020), 1872–1888. https://doi.org/10.1007/s10489-020-01652-0
24. T. K. Truong, A new moth-flame optimization algorithm for discounted {0–1} knapsack problem, *Math. Probl. Eng.*, **2021** (2021), 5092480. https://doi.org/10.1155/2021/5092480
25. X. Hao, Y. C. He, X. B. Zhu, Q. L. Zhai, Discrete hybrid multi-verse optimization algorithm for solving discounted 0–1 knapsack problem, *Comput. Eng. Appl.*, **57** (2021), 103–113. https://doi.org/10.3778/j.issn.1002-8331.2008-0454
26. M. Zhang, W. H. Deng, J. Li, Y. W. Zhong, Modified ant colony optimization algorithm for discounted {0–1} knapsack problem, *Comput. Eng. Appl.*, **57** (2021), 85–95. https://doi.org/10.3778/j.issn.1002-8331.2006-0278
27. A. Sulaiman, M. Sadiq, Y. Mehmood, M. Akram, G. A. Ali, Fitness-based acceleration coefficients binary particle swarm optimization (FACBPSO) to solve the discounted knapsack problem, *Symmetry*, **14** (2022), 1208. https://doi.org/10.3390/SYM14061208
28. H. S. Zhang, Y. C. He, J. H. Wang, F. Sun, M. L. Li, Enhanced group theory-based optimization algorithm for solving discounted {0–1} knapsack problem, *J. Front. Comput. Sci. Technol.*, **18** (2024), 1526–1542. https://doi.org/10.3778/j.issn.1673-9418.2305055
29. J. T. Zhao, X. C. Luo, A population-based simulated annealing approach with adaptive mutation operator for solving the discounted {0–1} knapsack problem, *Appl. Soft Comput.*, **181** (2025), 113480. https://doi.org/10.1016/j.asoc.2025.113480
30. Z. Dai, Y. Zhang, DJAYA-RL: Discrete JAYA algorithm integrating reinforcement learning for the discounted {0–1} knapsack problem, *Swarm Evol. Comput.*, **95** (2025), 101927. https://doi.org/10.1016/j.swevo.2025.101927
31. C. A. C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art, *Comput. Methods Appl. Mech. Eng.*, **191** (2002), 1245–1287. https://doi.org/10.1016/S0045-7825(01)00323-1
32. Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, *Evol. Comput.*, **4** (1996), 1–32. https://doi.org/10.1162/evco.1996.4.1.1
33. D. Pisinger, Where are the hard knapsack problems?, *Comput. Oper. Res.*, **32** (2005), 2271–2284. https://doi.org/ 10.1016/j.cor.2004.03.002
34. Á. E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, *IEEE Trans. Evol. Comput.*, **3** (1999), 124–141. https://doi.org/10.1109/4235.771166
35. Y. He, X. Wang, W. Li, X. Zhang, Y. Chen, Research on solving discounted {0–1} knapsack problem based on genetic algorithm, *Chin. J. Comput.*, **39** (2016), 2614–2630. https://doi.org/10.11897/SP.J.1016.2016.02614