Hybrid Bat Algorithm for the 0-1 Knapsack Problem (Postprint)

Authors: Wan Xiaoqiong, Zhang Huizhen

Date: 2018-05-24T00:00:00+00:00

Abstract

To address the shortcomings of the basic bat algorithm, such as its susceptibility to local optima entrapment and slow convergence speed, optimization research is conducted. Based on the specific characteristics of the 0-1 knapsack problem and building upon the original concepts and framework of the basic bat algorithm, the crossover mechanism from genetic algorithms and the inversion operator are introduced to establish novel position transition methods and local search rules. A greedy strategy is incorporated for solution feasibility enhancement and full utilization, enhancing local search capability and accelerating algorithm convergence speed, thereby constructing a novel hybrid bat algorithm. The hybrid bat algorithm is applied to two sets of 0-1 knapsack problem instances, and the simulation experimental results outperform the adaptive cellular particle swarm optimization algorithm, the basic bat algorithm, and the greedy binary bat algorithm. The results verify the feasibility and effectiveness of the hybrid algorithm in solving the 0-1 knapsack problem.

Full Text

Hybrid Bat Algorithm for Solving 0-1 Knapsack Problem

Wan Xiaoqiong, Zhang Huizhen† (School of Management, University of Shanghai for Science & Technology, Shanghai 200093, China)

Abstract

To address the limitations of the basic bat algorithm—namely, its tendency to fall into local optima and slow convergence speed—this paper proposes a novel hybrid bat algorithm tailored to the specific characteristics of the 0-1 knapsack problem. Building upon the fundamental concepts and framework of the basic bat algorithm, we introduce the crossover mechanism from genetic algorithms and an inverse operator to establish a new position transition method and local

search rule. Additionally, a greedy strategy is incorporated to ensure solution feasibility and full utilization of resources, thereby enhancing local search capability and accelerating convergence speed. The proposed hybrid bat algorithm was applied to two sets of 0-1 knapsack problem instances. Simulation results demonstrate that it outperforms adaptive cellular particle swarm optimization, the basic bat algorithm, and the greedy binary bat algorithm, confirming the feasibility and effectiveness of the hybrid approach for solving 0-1 knapsack problems.

Key words: 0-1 knapsack problem; bat algorithm; genetic algorithm; inverse operator; greedy strategy

0 Introduction

The knapsack problem (KP) is a classic NP-Complete combinatorial optimization problem in operations research that seeks to maximize the total value of items loaded into a weight-constrained knapsack. Numerous real-world problems can be formulated as knapsack problems, including investment decision-making, task scheduling, cargo loading, and material cutting. Knapsack problems are generally classified into three categories: 0-1 knapsack problem (0-1 KP), unbounded knapsack problem, and bounded knapsack problem. Among these, the 0-1 knapsack problem is the most fundamental, containing the essential design states and equations that underpin other variants, which can often be transformed into 0-1 knapsack problems for solution.

Since its introduction, the 0-1 knapsack problem has been a focal point of research. Existing solution methods fall into two main categories: exact algorithms and heuristic algorithms. Exact algorithms such as branch and bound, dynamic programming, backtracking, and greedy methods can obtain optimal solutions but suffer from combinatorial explosion as the number of items increases, making them impractical for large-scale problems. The emergence of heuristic algorithms has opened new avenues for addressing these challenges. Various heuristic approaches have been applied to the 0-1 knapsack problem, including ant colony optimization, harmony search, genetic algorithms, particle swarm optimization, wolf pack algorithms, whale optimization, and dragonfly algorithms. While these methods have achieved certain successes, they also exhibit shortcomings such as premature convergence and slow convergence in later stages.

To overcome the limitations of single heuristic algorithms, hybrid approaches have become increasingly popular for solving combinatorial optimization problems. Examples include hybrid genetic algorithms, ant colony algorithms based on genetic operators, and genetic algorithms incorporating greedy strategies. The bat algorithm (BA), proposed by Cambridge scholar Yang Xinshe in 2010 based on the echolocation behavior of bats, offers advantages such as few parameters, simple structure, and strong robustness. However, like other heuristic algorithms, the basic bat algorithm is prone to falling into local optima and

suffers from slow convergence. To address these weaknesses, this paper introduces the crossover mechanism from genetic algorithms to enhance global search capability, incorporates an inverse operator and greedy strategy to strengthen local search, and designs a novel hybrid bat algorithm (HBA) specifically for the 0-1 knapsack problem. Comparative experiments with benchmark instances from the literature demonstrate that HBA offers significant advantages in both optimization capability and convergence speed.

Problem Formulation

The 0-1 knapsack problem is a classic combinatorial optimization problem that can be described as follows: given a knapsack with weight capacity C and n items each with weight w_i and value v_i, the objective is to select items to maximize total value while respecting the knapsack capacity constraint. The mathematical model is expressed as:

$$\max \quad F(x) = \sum_{i=1}^{n} v_i x_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \leq C \tag{2}$$

$$x_i = \begin{cases} 1, & \text{if item } i \text{ is loaded into the knapsack} \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in I \tag{3}$$

where $I = \{i \mid i = 1, 2, \ldots, n\}$ is the set of items, $v = \{v_i \mid i = 1, 2, \ldots, n\}$ is the set of item values, $w = \{w_i \mid i = 1, 2, \ldots, n\}$ is the set of item weights, and C is the knapsack capacity. The objective function (1) maximizes the total value of loaded items, constraint (2) ensures the total weight does not exceed capacity, and constraint (3) defines the binary decision variables.

2 Hybrid Bat Algorithm

The bat algorithm is an intelligent optimization method that simulates the echolocation behavior of bat swarms in nature. Individual bats serve as the basic units of the algorithm, emitting strong ultrasonic waves through their mouths and nostrils while receiving echoes through their large ear auricles. By analyzing the time delay between emission and reception, intensity differences between ears, and waveform variations, bats construct a three-dimensional spatial scene to perceive the distance, direction, and nature of prey or obstacles, enabling them to devise hunting and avoidance strategies. Algorithmically, this translates to a population of s bats randomly distributed in an n-dimensional problem space, where each bat represents a candidate solution and the objective function value corresponds to solution fitness. The optimization process

involves iterative adjustment of loudness and pulse emission rate as bats follow the current global best solution.

2.1 Hybrid Bat Algorithm for 0-1 Knapsack Problem

Originally developed for continuous function optimization, the bat algorithm has recently been adapted for discrete problems through novel improvements. To apply the bat algorithm to the 0-1 knapsack problem, we employ binary encoding for position initialization, introduce genetic algorithm crossover mechanisms for position updating, and utilize an inverse operator to construct new local search rules. This hybrid approach combines the strengths of multiple techniques to create an effective optimization framework.

2.1.1 Initialization of Bat Positions Based on the binary nature of 0-1 knapsack solutions, we initialize the bat population using binary encoding. Each bat's position is represented by a $1 \times n$ dimensional vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, where $x_{ij} \in \{0, 1\}$ for $1 \leq j \leq n$. When $x_{ij} = 1$, item j is loaded into the knapsack; otherwise, it is not. The mapping relationship is shown in equation (4):

$$x_{ij} = \left\{ \text{0-1 knapsack problem} \right. \tag{4}$$

2.1.2 Velocity and Position Update Rules **Velocity Update Rule.** Due to the discrete nature of the optimization problem, we eliminate the frequency parameter f from the basic bat algorithm and redefine the velocity of bat i as the Manhattan distance between its position x_i and the global best bat x^*. The velocity update formula for bat i at time t is:

$$v_i^t = \sum_{j=1}^{n} |x_{ij} - x_j^*| \tag{5}$$

Position Update Rule. We employ the crossover mechanism from genetic algorithms for position transitions, where each bat corresponds to a chromosome. The position update process consists of two main steps:

a) **Inheritance of identical genes.** The global best bat x^* serves as parent 1, and bat x_i serves as parent 2. By comparing corresponding gene positions, genes that are identical (both 1 or both 0) are directly inherited by the offspring. For positions where genes differ, the offspring gene is marked with a wildcard "* ", resulting in an intermediate offspring position denoted as x_{mid}.

b) **Wildcard resolution.** The number of wildcards in x_{mid} equals v_i^t, which represents the transition speed of bat i. A following probability p_{fol} is introduced to determine how these wildcard positions are resolved, as shown in equation (6):

$$x_{new} = x_{mid}(v_i^t, p_{fol}) \tag{6}$$

Specifically, when rand $> p_{fol}$, the offspring follows the global best bat and inherits the corresponding gene from parent 1; otherwise, it inherits from parent 2. This crossover mechanism allows offspring to inherit excellent genes from parents while maintaining population diversity through the probabilistic following behavior, thereby mitigating premature convergence. Figure 1 [Figure 1: see original paper] illustrates the crossover recombination process for $n = 6$.

2.1.3 Local Search We implement two local search strategies: an inverse operator for searching around the global best bat, and a greedy strategy for handling infeasible and underutilized solutions.

1) Inverse Operator. Unlike other intelligent optimization algorithms, the bat algorithm can dynamically balance global and local search by comparing the pulse emission rate r_i with a random number. When rand $> r_i$, a global best bat is selected from the current best solution set for random walk local search. Due to the binary encoding scheme, traditional local search methods are ineffective. We introduce an inverse operator with the following rule: with inversion probability p_{not}, randomly select $n \times p_{not}$ genes from the global best bat and invert them (0 becomes 1, 1 becomes 0) to generate a new local solution x_{new}. This random inversion enables the algorithm to escape local optima as the population becomes increasingly similar during iterations. The specific optimization method is:

$$x_{new,j_{not}} = \Big\{\text{Figure 1 Bat algorithm position update process} \tag{7}$$

where $j_{not} \in J_{not}$ represents the selected gene positions.

2) Greedy Strategy. As a constrained combinatorial optimization problem, the 0-1 knapsack problem may generate infeasible solutions (exceeding weight capacity) during random initialization, position updates, and inverse operator local search. Directly discarding these solutions would waste valuable search information and slow convergence. Therefore, we employ a greedy strategy to repair infeasible solutions: calculate the value-to-weight ratio $vol_i = v_i/w_i$ for loaded items and iteratively remove items with the lowest ratios until the total weight satisfies the capacity constraint. For feasible solutions that underutilize knapsack capacity, we similarly apply a greedy strategy: calculate vol_i for unloaded items and add those with the highest ratios until no further items can be added without violating the capacity constraint. This greedy strategy is applied after initialization, position updates, and inverse operator local search to ensure solution feasibility and full capacity utilization.

2.1.4 Update Rules for Loudness and Pulse Emission Rate As bats approach prey, their ultrasonic loudness gradually decreases while pulse emission

rate increases. The update formulas for loudness A_i^t and pulse emission rate r_i^t are:

$$A_i^t = \alpha A_i^{t-1} \tag{8}$$

$$r_i^t = r_i^0 \left(1 - e^{-\gamma t}\right) \tag{9}$$

where α is the loudness attenuation coefficient, γ is the pulse emission rate increase coefficient, and r_i^0 is the initial pulse frequency. For any $0 < \alpha < 1$ and $\gamma > 0$, we observe that $A_i^t \to 0$ and $r_i^t \to r_i^0$ as $t \to \infty$.

2.2 Algorithm Design for 0-1 Knapsack Problem

The complete hybrid bat algorithm for solving 0-1 knapsack problems is designed as follows:

a) **Initialize bat population.** Set population size s, position x_i, velocity v_i, following probability p_{fol}, inversion probability p_{not}, loudness A_i^0, initial pulse emission rate r_i^0, and maximum iterations N_{gen}. Initialize bat positions using equation (4).

b) **Update velocity and position.** Identify the global best bat x^*, update velocity v_i^t using equation (5), and update position x_{new} using the genetic crossover mechanism.

c) **Perform local search.** If rand $> r_i$, conduct local search on a selected global best bat using equation (7) to obtain x_{new}.

d) **Update satisfactory solutions.** If rand $< A_i$ and $F(x_{new}) < F(x^*)$, update the current satisfactory solution and its objective function value.

e) **Update loudness and pulse emission rate.** Decrease A_i using equation (8) and increase r_i using equation (9).

f) **Rank bats and identify global best.** Sort the population and find the current global best bat x^*.

g) **Check termination.** Increment iteration counter $N_{iter} = N_{iter} + 1$. If $N_{iter} > N_{gen}$, terminate and output results; otherwise, return to step b).

(Note: The greedy strategy for solution feasibility and capacity utilization is applied after initialization, position updates, and inverse operator local search.)

3 Simulation Experiments and Analysis

To validate the effectiveness of the proposed hybrid bat algorithm, we conducted experiments using two sets of benchmark instances and compared performance against state-of-the-art algorithms.

Experimental Setup

The simulation environment consisted of an Intel(R) CoreTM i5-3320 CPU @ 2.60GHz with 2.0 GB RAM, 64-bit Windows 8 operating system, and MATLAB R2015a. For fair comparison, we matched experimental parameters with those used in comparative studies: population size $s = 50$, maximum iterations $N_{gen} = 500$, and $T = 30$ independent runs per instance for the first test set; $s = 50$, $N_{gen} = 300$ (or 500 for convergence analysis), and $T = 50$ runs for the second test set. Algorithm-specific parameters were set as: initial loudness $A_i^0 = 0.25$, initial pulse emission rate $r_i^0 = 0.5$, $\alpha = \gamma = 0.9$, following probability $p_{fol} = 0.5$, and inversion probability $p_{not} = 0.2$ for the first test set; $A_i^0 = 0.75$, $r_i^0 = 0.005$, $\alpha = 0.95$, $\gamma = 0.7$ for the second test set.

Test Results

First Test Set: Nine 0-1 knapsack instances from reference [23] were used for comparison with adaptive cellular particle swarm optimization (ACPSO). Table 1 provides instance dimensions, weight sets w, value sets v, knapsack capacities C, and known optimal values. Table 2 presents the experimental results.

The results show that HBA achieved the optimal value in all 30 runs for instances KP1-KP8 (except KP6 where success rate was still high), while ACPSO only achieved 100% success on KP1, KP5, and KP8. For KP9, HBA found a better solution (4987) than the previously known optimum (4986), demonstrating superior global search capability. Moreover, HBA required significantly fewer iterations to converge, as evidenced by the minimum, average, and maximum iteration counts. The total computational time was also substantially lower than ACPSO. Overall, HBA exhibits stronger robustness and faster convergence than ACPSO.

Second Test Set: Three instances from reference [24] were used for comparison with the basic bat algorithm (BA) and greedy binary bat algorithm (GBBA). Table 3 details instance 3, while Tables 4 and 5 present objective value comparisons and convergence analysis, respectively.

HBA and GBBA both found optimal solutions for all three instances, while basic BA only succeeded on instance 1. HBA achieved 100% success rates on instances 1 and 2, and a higher success rate on instance 3 compared to BA and GBBA. The average solution quality was consistently superior across all instances. Figures 2 [Figure 2: see original paper] through 4 [Figure 4: see original paper] illustrate the convergence curves, showing that HBA reaches optimal values within fewer iterations while BA becomes trapped in local optima.

4 Conclusion

This paper addresses the specific characteristics of the 0-1 knapsack problem by integrating genetic algorithm operators, an inverse operator, and a greedy

strategy into the basic bat algorithm framework. The resulting hybrid bat algorithm leverages the complementary strengths of these techniques to overcome the limitations of premature convergence and slow convergence inherent in the basic bat algorithm. Comprehensive simulation studies demonstrate that the proposed HBA outperforms adaptive cellular particle swarm optimization, the basic bat algorithm, and the greedy binary bat algorithm in terms of solution quality, convergence speed, and robustness. The hybrid bat algorithm represents an effective and efficient method for solving 0-1 knapsack problems.

References

[1] Fayard D, Plateau G. Resolution of the 0-1 knapsack problem comparison of methods [J]. Mathematical Programming, 1975, 8 (1): 272-307.

[2] Sheng Hongbo, Sun Juan, Sun Xiaoling. A rigor method for solving 0-1 polynomial knapsack problem [J]. Journal of the Shanghai University: Natural Science, 2006, 12 (4): 389-393.

[3] Shen Jingcheng, Shigeoka K, Ino F, et al. An out-of-core branch and bound method for solving the 0-1 knapsack problem on a GPU [C]// International Conference on Algorithms & Architectures for Parallel Processing, 2017.

[4] Elkihel M, Baz D E. An efficient dynamic programming parallel algorithm for the 0-1 knapsack problem [C]// Parallel Computing-advances & Current Issues-the International Conference Parco, 2015: 298-305.

[5] Xu Ying. Application of backtracking method in 0-1 knapsack problem [J]. Software Guide, 2008 (12): 54-55.

[6] Shi Lan, Zhang Yihong, Lv Jianhui. Optimization algorithm of 0-1 knapsack problem based on absolute greedy and expected efficiency [J]. Application Research of Computers, 2014, 31 (3): 684-687.

[7] Hu Xiaobing, Huang Xiyue. Solving 0-1 knapsack problem based on ant colony optimization algorithm [J]. Journal of Systems Engineering, 2005, 20 (5): 520-523.

[8] Qin Ling, Bai Yun, Zhang Chunfang, et al. Ant colony algorithm for 0-1 knapsack problem [J]. Computer Engineering, 2006, 32 (6): 212-214.

[9] Li Ruoping, Ouyang Haibin, Gao Liqun, et al. Learned harmony search algorithm and its application to 0-1 knapsack problems [J]. Control and Decision, 2013, 28 (2): 205-210.

[10] Wang Ling, Yang Ruixin, Xu Yin, et al. An improved adaptive binary harmony search algorithm [J]. Information Sciences, 2013, 232 (5): 58-87.

[11] Zou Dexuan, Gao Liqun, Li S, et al. Solving 0-1 knapsack problem by a novel global harmony search algorithm [J]. Applied Soft Computing Journal, 2011, 11 (2): 1556-1564.

[12] Wang Li, Shao Dinghong, Lu Jingui. The genetic algorithm of solving 0/1 knapsack problem [J]. Computer Simulation, 2006, 23 (3): 154-156.

[13] Caro F. A flipping local search genetic algorithm for the multidimensional 0-1 knapsack problem [C]// Proc of Spanish Association Conference on Current Topics in Artificial Intelligence, 2005: 21-30.

[14] Zhang Guoli, Wei Yi. An improved particle swarm optimization algorithm for solving 0-1 knapsack problem [C]// International Conference on Machine Learning & Cybernetics, 2008, 2: 915-918.

[15] Haddar B, Khemakhem M, Rhimi H. A quantum particle swarm optimization for the 0-1 generalized knapsack sharing problem [J]. Natural Computing, 2016, 15 (1): 153-164.

[16] Wu Husheng, Zhang Fengming, Zhan Renjun, et al. A binary wolf pack algorithm for solving 0-1 knapsack problem [J]. Systems Engineering and Electronics, 2014, 36 (8): 1660-1667.

[17] Abdel-Basset M, El-Shahat D, Sangaiah A K. A modified nature inspired meta-heuristic whale optimization algorithm for solving 0-1 knapsack problem [J]. International Journal of Machine Learning & Cybernetics, 2017 (1): 1-20.

[18] Abdel-Basset M, Luo Qifang, Miao Fahui, et al. Solving 0-1 knapsack problems by binary dragonfly algorithm [C]// Proc of International Conference on Intelligent Computing, 2017: 491-502.

[19] Cotta C, Troya J M. A hybrid genetic algorithm for the 0-1 multiple knapsack problem [C]// Artificial Neural Nets & Genetic Algorithms, 1998: 250-254.

[20] Hu Zhijun, Li Rong. Ant colony optimization algorithm for the 0-1 knapsack problem based on genetic operators [C]// Advanced Materials Research, 2011: 973-977.

[21] Zhao Jiangfei, Huang Tinglei, Pang Fei, et al. Genetic algorithm based on greedy strategy in the 0-1 knapsack problem [C]// Proc of International Conference on Genetic & Evolutionary Computing, 2010: 105-107.

[22] Yang Xinshe. A new metaheuristic bat-inspired algorithm [J]. Computer Knowledge & Technology, 2010, 284: 65-74.

[23] Li Zhiyong, Ma Liang, Zhang Huizhen. Adaptive cellular particle swarm algorithm for solving 0/1 knapsack problem [J]. Computer Engineering, 2014, 40 (10): 198-203.

[24] Wu Congcong, He Yichao, Chen Yiying, et al. Binary bat algorithm for solving 0-1 knapsack problem [J]. Computer Engineering and Application, 2015, 51 (19): 71-74.

Note: Figure translations are in progress. See original paper for figures.