Discrete Bare Bones Particle Swarm Optimization Algorithm for Solving Multidimensional Knapsack Problem

Shengjian Liu, Yan Yang* and Lin Luo

Department of Game Design, South China Institute of Software Engineering
Guangzhou University, Guangzhou, GuangDong, 510990, China
yangyan_08@yeah.net

Abstract. Aiming at the characteristics of multi-dimensional knapsack problem, the widely used bare bones particle swarm optimization algorithm is discretized, and the traditional objective function based on large penalty parameters is improved. The infeasible solution is transformed into a feasible solution by using the repair mechanism. The typical examples of 10 multi-dimensional knapsack problems are simulated and compared with four intelligent optimization algorithms. The simulation results show that the discrete bare bones particle swarm optimization algorithm has good convergence efficiency, high precision and good robustness when solving multi-dimensional knapsack problem.

1. Introduction

Since the introduction of the knapsack problem in 1897 by Mathews, its application field has been expanded and its formation has been generalized. The knapsack problem is limited by many kinds of resources, such as resource cost in project management, task time and other constraints. This kind of knapsack problem constrained by multiple resource is called Multidimensional Knapsack Problem (MKP) [1]. The MKP is also often referred to as multidimensional 0-1 knapsack problems or multiconstraint 0-1 knapsack problems.

A description of the MKP includes m objects and n knapsacks with specific capacities Cj (j = 1,...,n). There are binary variables xi (i = 1,...,m) that are set to 1 if the i-th object is selected to be put in the knapsacks, and 0, otherwise. Every object has a satisfaction value pi (i = 1,...,n) and a specific weight wij for each knapsack. Therefore, the optimization problem is defined as follows:

\[
\begin{align*}
\max f(x) &= \sum_{i=1}^{m} p_i x_i, i \in \{1,2,\ldots,m\}, x_i \in \{0,1\} \\
\text{s.t.} & \quad \sum_{i=1}^{m} w_{ij} x_i \leq C_j, j \in \{1,2,\ldots,n\}
\end{align*}
\]

The constrained optimization problem of equation (1) is transformed into unconstrained optimization problem as follows:

\[
\max f(x) = \sum_{i=1}^{m} p_i x_i - \sum_{j=1}^{n} \max(0, \sum_{i=1}^{m} w_{ij} x_i - C_j)
\]

Many problem-independent and domain-specific heuristics have been developed for optimization problems and, in particular, to the MKP that is NP-complete. The quest for a computational algorithm that are effective both in terms of processing time and quality of the solutions is the motivation of this work. Therefore, we propose a methodology for solving MKP using a relatively recent evolutionary
computation technique, the Particle Swarm Optimization (PSO), proposed by Kennedy and Eberhart [7]. PSO, in fact, has been applied successfully to several problems like pattern recognition, classification, scheduling, mobile robotics, image processing, and others [5]. In this paper, we propose an improved PSO to solve MKP.

2. Bare Bones Particle Swarm (BBPSO)
PSO is a heuristic method for optimization, inspired in the behavior of social agents found in nature. This behavior can be observed in bird flocking, bee swarming, and fish schooling, for instance. The PSO has just enough moving parts to make it hard to understand. The formula is very simple, it is even easy to describe the working of the algorithm verbally, yet it is very difficult to grasp in one’s mind how the particles oscillate around centers that are constantly changing; how they influence one another; how the various parameters affect the trajectory of the particle; how the topology of the swarm affects its performance; and so on.

BBPSO is proposed by Kennedy in 2003[4]. The algorithm cancels the velocity term, and the position of the particle is obtained by random sampling obeying the Gaussian distribution. BBPSO simplifies the parameter adjustment and has been successfully applied to kinds of fields such as constrained optimization problem [5], data mining [6], power system regulation [7], and so on. The update method of the particle position in BBPSO is as follows:

\[
\begin{align*}
    x_{id}^{t+1} &\sim N(u_{id}, \delta_{id}^2) \\
u_{id} &= \frac{px_{id} + gx_{id}}{2} \\
    \delta_{id} &= \left| px_{id} - gx_{id} \right|
\end{align*}
\]

Where \( t \) is the current number of iterations, \( N(u_{id}, \delta_{id}^2) \) is a Gaussian distribution with a mean of \( u_{id} \) and a standard deviation of \( \delta_{id} \), \( x_{id} \) is \( i \)'s current position in the \( d-th \) search space, \( px_{id} \) is the best position found so far by individual \( i \) in the \( d-th \) search space, and \( gx_{id} \) is the best position found so far by the particle swarm in \( d-th \) search space.

The exploratory BBPSO is proposed by Kennedy at the same time[4]. In each dimension search space, the best position of individual is retained with 50% probability, and sampled by Gaussian distribution, otherwise. In this paper, propose an improved BBPSO based on the exploratory BBPSO.

3. Discrete Bare bones particle swarm (DBBPSO)

3.1 Conversion of position
The swarm intelligent optimization algorithms such as PSO and bat algorithm mainly solve the optimization problem of continuous function. The 0-1 knapsack problem is a typical discrete optimization problem. In the solution space, each item can only be placed or not. So, the solution is to preserve the original contiguous search space and use a conversion function such as sigma function. The conversion function compresses the range of real numbers from negative infinity to positive infinity to the range of (0,1), so that the original algorithm formula can be used to deal with discretization problems.

In continuous search space, randomly initialize a candidate solution and the probability that the \( j-th \) component of a potential solution approximates to the optimal solution is very low. For 0-1 knapsack problem, the \( j-th \) component is that item \( j \) is selected into the bag or not, and this component is probably the optimal solution with 50% probability. So it is better to directly define the domain of potential solution as \{0,1\}.
3.2 Update of position

Location update of DBBPSO refers to the idea of references 8, and the position and speed updating formulas are as follows:

\[ v_{id}^{t+1} = \left[ \frac{p_x^i + g_x^i + r_3}{2} \right] - x_{id} \]

\[ x_{id}^{t+1} = \begin{cases} 1 - x_{id}, & r_1 < 0.5 \text{ and } r_2 < \frac{2}{\pi} \arctan\left( \frac{v_{id}^{t+1}}{2} \right) \\ x_{id}, & \text{otherwise} \end{cases} \]

where \( t \) is the current number of iterations, \( r_1 \) and \( r_2 \) are random-uniform numbers in the interval [0, 1). \( r_3 \) is a random-uniform number in the interval [-0.5, 0.5]. \( x_{id}, p_x^i \) and \( g_x^i \) are defined as chapter 2.

3.3 Restore of invalid position

The initial position is random values, so part of the solution may not satisfy the constraint conditions and become invalid solution. If you use the formula (2) to calculate the fitness, the result is bound to be less than zero, and it will cause a lot of invalid test. So we use the greedy algorithm to fix position[9], and it can greatly shorten the optimization time. The concrete implementation process is as follows.

Step 1 Generate sequence \( T = (T_1, T_2, \cdots, T_m) \) according to ascending order of the value \( \{\rho_i\} \), where \( \rho_i \) is unit value of the item and it’s calculated by formula (5).

\[ \rho_i = \min \left( \frac{c_j}{w_j} \right), \quad i = 1,2,\cdots,m; \quad j = 1,2,\cdots,n \]

Step 2 Test the item coded as 1 in the infeasible solution by sequence T, that is, the items with low unit value are taken out, and set the coding as 0. Go to step 3 when the constraint is satisfied, otherwise, test the next item.

Step 3 Reverse sequence T and obtain feasible solution by step 2. Then, test the item coded as 0 and put it into the knapsack. If the item is loaded and the constraint is satisfied, set the item’s code as 1, otherwise, test the next item. Finally, the feasible solution repaired will be obtained.

3.4 Implementation steps of DBBPSO

Step 1 Initialize parameter. Number of particles is \( N \), the maximum iteration number is \( T \), the dimension of item is \( m \), the number of constraint for knapsack is \( n \), the position of each particle is initialized randomly and the position is \( x_i, (1 \leq i \leq N) \).

Step 2 Calculate the fitness value according to formula (2). Set the individual historical optimal position and fitness to the current position and fitness of each particle. The best fit position is the optimal position of the group.

Step 3 Update position of each particle according to formula (4).

Step 4 Correct the infeasible solution according to formula (5).

Step 5 Calculate the fitness value according to formula (2).

Step 6 Update the historical optimal position of individual and particle swarm, and randomly reset the position for particle which has founded the optimal position.

Step 7 Determine whether the algorithm satisfies the end condition. If it is found that the optimal value or the maximum number of iterations, turn to step 8, otherwise, the number of iterations increases 1 and turn to step 3.

Step 8 Output the optimal position of the swarm, that is, the optimal solution of the problem. End of the algorithm.
4. Simulation experiment and algorithm validity test

In order to effectively verify the effect of the algorithm on the basis of fairness, the binary bat algorithm (BBA) [10] is selected and compared, and the potential solution (5) is corrected on the original algorithm to make it a feasible solution. The experimental hardware environment of these two algorithms is Intel Xeon E3-1230V2@3.30GHz and 16G memory, the operating system is Win10, and the experimental simulation software is python3.7 and numpy.

In addition, based on research literature in recent years, the binary opposite learning fireworks algorithm (BOLFWA) [11] and the binary particle swarm optimization with time-varying accelerated coefficients (BPSO-TVAC) [12] were compared. The binary reverse learning fireworks algorithm simulation experiment environment is: HPZ4 Intel (R) E5-1620v2 @3.70 GHz, 32 GB memory, Win8.1, 64bit, Matlab R2014a. The experimental environment for the binary time-varying accelerated particle swarm optimization algorithm is: Intel Core 2 processor with 2.83 GHz CPU and 2G RAM.

4.1 Choose test cases

In order to test the effectiveness of DBBPSO for multi-dimensional knapsack problem, this paper chooses the following five types of 10 examples [11] and each case independently run 20 times. The test cases show in table 1.

| Knapsack dimensional | Item number | Testing sample |
|----------------------|-------------|----------------|
| Low                  | Less        | weing1, weing2 |
| Low                  | Many        | weing7, weing8 |
| Middle               | Less        | pet2, pet3     |
| Middle               | Many        | pet5, pet7     |
| High                 | Many        | sent01, sent02 |

4.2 Set algorithm parameters

The common parameters are set in the same way. The population size \( N \) is twice the number of the items, \( N = 2m \), and the maximum number of iterations \( T \) is 10 times of the number of the items, \( T = 10m \). The non-universal parameters of each algorithm are shown in Table 2.

| algorithm                                      | parameter               |
|-----------------------------------------------|-------------------------|
| Discrete Bare Bones Particle Swarm Optimization | \( \lambda^0 = 0.25 \), \( r^0 = 0.5 \), \( \alpha = 0.9 \), \( \gamma = 0.9 \) |
| Binary Bat Algorithm (BBA)                     | \( r1 = 0.5 \)           |
| Binary Opposite Learning Fire Works Algorithm (BOLFWA) | \( \omega = 0.8^4 \), \( c1 = c2 = 1.414 \), \( v \in [-0.5, 0.5] \) |
| Binary Particles Swarm Optimization-Time Varying Accelerated Coefficients (BPSO-TVAC) | \( \omega = 0.8^4 \), \( c1 = c2 = 1.414 \), \( v \in [-0.5, 0.5] \) |

In order to effectively compare the optimal performance of each algorithm, evaluate the algorithms according to the following six indicators, such as the optimal value, the worst value, the average value(mean), and the standard deviation(STD) of the 20 times optimization results, the success rate(SR) and the average number of iterations for successful optimization(ANIS).

The SR is the ratio of the number of successful optimization to the number of experiments, and it shows whether the algorithm can obtain the global optimal solution stably. Successfully obtain the
optimal value is that the optimal value is close to the theoretical optimal value and the error of the optimal value is not more than 1%.

The ANIS is to sum up the number of iterations for each optimization, and then calculate the average value. If there is no successful optimization, the maximum number of iterations will be as number of optimization.

The comparison results are shown in table 3, where the results of the BOLFWA[11] and the BPSO-TVAC[12] come from the references.

Comparing the experimental average of the algorithm, the larger is the better. In the case of the same experimental average, the larger success rate is the better. If the first two indicators are same, the average number of iterations of the successful optimization is compared. The smaller is the better, and for the standard deviation, the smaller is the better. For each case, a comprehensive evaluation is performed, and the algorithm with the best effect is marked in bold type. Since there is a certain difference from the machine hardware in the reference, the index of optimization time is not considered.

Table 3. Experimental results of 4 algorithms

| Sample | Dimension (m-k) | Optimal solution | indicator | DBBPSO | BBA | BOLFWA | BPSO-TVAC |
|--------|----------------|------------------|-----------|--------|-----|--------|-----------|
| weing1 | 28-2           | 141278           | Optimal value 141278 | **141278** | 141278 | 141278 |
|        |                |                  | Worst value 141278  | **141278** | 140477 | 140543 |
|        |                |                  | mean 141278         | **141278** | 140888 | 141110 |
|        |                |                  | STD 0               | 0       | 327.84 | 282.83 |
|        |                |                  | SR 100%            | **100%** | 100%  | 100%   |
|        |                |                  | ANIS 32             | 26.25   | 76.13 | 58.11  |
| weing2 | 28-2           | 130883           | Optimal value 130883 | **130883** | 130883 | 130883 |
|        |                |                  | Worst value 130883  | **130883** | 130372 | 130158 |
|        |                |                  | mean 130883         | **130883** | 130582 | 130679 |
|        |                |                  | STD 0               | 73.32   | 525.89 | 400.45 |
|        |                |                  | SR 100%            | **100%** | 100%  | 100%   |
|        |                |                  | ANIS 19.05          | 123.45  | 120.95 | 89.25  |
| weing7 | 105-2          | 1095445          | Optimal value 1095445 | **1095445** | 1095445 | 1082147 |
|        |                |                  | Worst value 1095413.5 | **1095394.6** | 1091731 | 1011871 |
|        |                |                  | mean 1095382        | **1095382** | 1093053 | 1047735 |
|        |                |                  | STD 31.5            | 25.2    | 3363  | 13983  |
|        |                |                  | SR 100%            | **100%** | 100%  | 80%    |
|        |                |                  | ANIS 818.55         | 893.95  | 357.28 | 895.73 |
| weing8 | 105-2          | 624319           | Optimal value 624319 | 624319 | 624319 | 620060 |
|        |                |                  | Worst value 624157.35 | 623995.7 | 620060 | 419375 |
|        |                |                  | mean 621086         | 621086  | 622726 | 589631 |
|        |                |                  | STD 704.62          | 969.9   | 13381 | 86951  |
|        |                |                  | SR 100%            | **100%** | 100%  | 75%    |
|        |                |                  | ANIS 90.9           | 339.35  | 326.95 | 983.36 |
| pet2   | 10-10          | 87061            | Optimal value 87061 | 87061 | 87061 | 87061 |
|        |                |                  | Worst value 87061   | 87061   | 87061 | 87061 |
|        |                |                  | mean 87061          | 87061   | 87061 | 87061 |
|        |                |                  | STD 0               | 899.73  | 0     | 1455.69 |
|        |                |                  | SR 100%            | 45%     | 100%  | 90%    |
|        |                |                  | ANIS 6              | 8.11    | 13.10 | 48.40  |
The optimization results of typical test cases by the DBBPSO algorithm are shown in Table 3. The analysis results are as follows.

First, for test examples for 28 items with 2 resource constraints weing1, weing2, 4 intelligent evolution algorithms can be solved quickly, but for binary opposite learning fireworks algorithm (BOLFWA) and binary particle swarm optimization with time-varying accelerated coefficients (BPSO-TVAC), there is a standard deviation between the optimal solution and the theoretical optimal value. For weing1, the binary bat algorithm (BBA) has the best solution effect. The solving capability of the discrete bare bones particle swarm optimization algorithm (DBBPSO) is very well except the average iteration number is larger. For weing2, only the DBBPSO can find the theoretical optimal value.

Second, for the test examples of 105 items restricted by 2 kinds of resources, weing7, weing8, the optimization performance of 4 kinds of intelligent evolutionary algorithms gradually weakened, and the average iterations number of optimization is relatively large, which are more than 350 times. The BOLFWA is faster, but its average optimal value is poor than the BBA and DBBPSO.

Third, for 10 kinds of test cases with limited resources and few items, pet1, pet2, 4 kinds of intelligent evolution algorithms can be solved quickly. Except the discrete bare bones particle swarm algorithm, there are standard deviation for the optimal solutions and theories found by the other three algorithms. Comprehensive evaluation of the DBBPSO is the best.

| Test Case | Resource | Optimal Value | Worst Value | Mean | STD | SR | ANIS |
|-----------|----------|---------------|-------------|------|-----|----|------|
| pet3      | 15-10    | 4015          | 4015        | 4015 | 0   | 100% | 20.5 |
|           |          | 4015          | 4011.5      | 4005 | 4.77 | 100% | 75.2 |
|           |          | 4015          | 4005        | 4014 | 3.08 | 100% | 75.2 |
|           |          | 4015          | 4005        | 4014 | 21.28| 95%  | 43.45|
| pet5      | 28-10    | 12400         | 12400       | 12400| 0   | 100% | 20.5 |
|           |          | 12400         | 12330       | 12372| 0   | 100% | 75.2 |
|           |          | 12400         | 12330       | 12372| 25.46| 100% | 75.2 |
|           |          | 12400         | 12330       | 12372| 164.42| 90%  | 75.2 |
| pet7      | 50-5     | 16537         | 16537       | 16537| 0   | 100% | 20.5 |
|           |          | 16537         | 16517.1     | 16373| 0   | 100% | 75.2 |
|           |          | 16537         | 16517.1     | 16373| 79.29| 100% | 75.2 |
|           |          | 16537         | 16517.1     | 16373| 164.42| 90%  | 75.2 |
| Sent01    | 60-30    | 7772          | 7772        | 7772 | 0   | 100% | 20.5 |
|           |          | 7772          | 7772        | 7772 | 0   | 100% | 75.2 |
|           |          | 7772          | 7772        | 7772 | 69.09| 100% | 75.2 |
|           |          | 7772          | 7772        | 7772 | 681.2| 100% | 75.2 |
|           |          | 7772          | 7772        | 7772 | 354.20| 100% | 75.2 |
|           |          | 7772          | 7772        | 7772 | 467.40| 100% | 75.2 |
| Sent02    | 60-30    | 8722          | 8722        | 8722 | 0   | 100% | 20.5 |
|           |          | 8722          | 8722        | 8722 | 0   | 100% | 75.2 |
|           |          | 8722          | 8722        | 8722 | 68.01| 100% | 75.2 |
|           |          | 8722          | 8722        | 8722 | 151.67| 100% | 75.2 |
|           |          | 8722          | 8722        | 8722 | 569.46| 100% | 75.2 |

The optimization results of typical test cases by the DBBPSO algorithm are shown in Table 3. The analysis results are as follows.
Fourth, for 10 resource constraints, 28 items of test examples pet5, DBBPSO and BBA can find the theoretical optimal value, the later optimization time is shorter. For the 50 resource limit and the test example pet7 of 5 items, there are cases where the theoretical optimal value can not be found in each algorithm. The DBBPSO has a relatively good performance, but the optimization time is longer. The BOLFWA has a shorter optimization time, but the accuracy is not high.

Finally, for 30 resource limits, 60 items test examples sent01, sent02, BOLFWA and BPSO-TVAC not only have long optimization time, but also low precision. The DBBPSO and BBA can find the theoretical optimal value every time. The optimal average iteration number is less than 43 times, and the DBBPSO proposed in this paper shows good solution ability.

In summary, both the DBBPSO and BBA proposed in this paper have a good performance, and Figure 1 to 10 further show the performance of the two algorithms in the optimization process.
5. Conclusion
Inspired by continuous bare bones particle swarm algorithm and binary bat algorithm, the V-transform function is constructed to convert continuous data into discrete space, Gaussian sampling is simplified to uniform sampling, and the infeasible solution is feasible corrected using greedy algorithm. Discrete
A bare-bones particle swarm algorithm is created, which are compared with the classic binary bat algorithm, binary opposite learning fireworks algorithm and binary particle swarm optimization with time-varying accelerated coefficients algorithm. The experimental results show that the discrete bare bones particle swarm optimization algorithm can achieve good optimization results when solving different types of multi-dimensional knapsack problems, especially in the problem of high dimension backpack and large number of items. There is no set parameters and optimize the time end, and the quality of the search is high, and the algorithm is robust. In the future, the algorithm can be applied to the actual engineering project.

Acknowledgments
This research was supported by the Key platform and Major Program of Guangdong Province under Grant No.2016KTSCX189 and by South China Institute of Software Engineering of Guangzhou University under Grant No. ky201823.

References
[1] DEANE J, AGARWAL A. Neural, genetic, and neurogenetic Approaches for solving the 0-1 multidimensional knapsack Problem[J].International Journal of Management and Information Systems, 2013, 17(1):43-54.
[2] Wang L, Liu B. Particle swarm optimization and scheduling algorithms. Beijing: Tsinghua University Press, 2008 (in Chinese)
[3] Shi Y, Eberhart R. A modified particle swarm optimizer//Proceedings of the IEEE Conference on Evolutionary Computation. Anchorage, USA, 1998: 69-73.
[4] Kennedy J. Bare bones particle swarms//Proceedings of the IEEE Swarm Intelligence Symposium. Indianapolis, IN, USA, 2003: 80-87.
[5] Campos M., Krohling R.A. Bare bones particle swarm with scale mixture of Gaussians for dynamic constrained optimization//Proceedings of the IEEE Congress on Evolutionary Computation. Beijing, China, 2014: 202-209.
[6] Zhang Y., Gong D, Hu Y, Zhang W. Feature selection algorithm based on bare bones particle swarm optimization. Neurocomputing, 2015, 148: 150-157.
[7] Koshti A, Arya L D, Choube S C. Voltage stability constrained distributed generation planning using modified bare bones particle swarm optimization. Journal of the Institution of Engineers (India) Series B (Electrical, Electronics & Telecommunication and Computer Engineering), 2013, 94(2): 123-133.
[8] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization," Swarm and Evolutionary Computation, vol. 9, pp. 1-14, 2013.
[9] WU H S,ZHANG F M ,ZHAN R J,et al.Improved binary wolf pack algorithm for solving multidimensional knapsack problem [J].Systems Engineering and Electronics,2015,37(5):1084-1091(in Chinese)
[10] Mirjalili S, Mirjalili SM, Yang X-S (2013) Binary bat algorithm. Neural Comput Appl 25(3–4):663–681.
[11] Xue J J , Wang Y , Meng X F, et al. Binary Opposite backward learning Fireworks algorithm for Multidimensional knapsack problem[J].Systems Engineering and Electronics, 2017,39(2):451-457(in Chinese)
[12] CHIL M C , LIN C J , CHERM M S , et al. Particle swarm optimization with time - varying acceleration coefficients for the multidimensional knapsack problem [J] . Applied Mathematical Modelling, 2014, 38(4): 1338 - 1350.