Indonesian Journal of Electrical Engineering and Computer Science
Vol. 1, No. 3, March 2016, pp. 411 ∼ 418
DOI: 10.11591/ijeecs.v1.i3.pp411-418

A New Hybrid Particle Swarm Optimization and Greedy for 0-1 Knapsack Problem

Phuong Hoai Nguyen[a,b], Dong Wang[a] , and Tung Khac Truong[*]
[a]College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China.n
[b]Department of Computer Science, University of Labour and Social Affairs, Vietnam.
[*]Faculty of Information Technology and IUHYRA, Industrial University of Ho Chi Minh city, Vietnam., e-mail: tungtk@iuh.edu.vn

Abstract

This paper proposes a new binary particle swarm optimization with a greedy strategy to solve 0-1 knapsack problem. Two constraint handling techniques are consider to cooperation with binary particle swarm optimization that are penalty function and greedy. The sigmoid transfer function is used to convert real code to binary code. The experimental results have proven the superior performance of the proposed algorithm.

Keywords: *Combinatorial, Greedy, 0-1 knapsack problem, PSO.*

1. Introduction

The 0-1 knapsack problem(KP01) is known to be a combinatorial optimization problem. The knapsack problem has a variety of practical applications such as cutting stock problems, portfolio optimization, scheduling problems [1] and cryptography [2, 3, 4]. The knapsack appears as a sub-problem in many complex mathematical models of real-world problems. In a given set of *n* items, each of them has an integer weight w_i and an integer profit p_i. The problem is to select a subset from the set of *n* items such that the overall profit is maximized without exceeding a given weight capacity *C*. It is a NP-Hard problem and hence it does not have a polynomial time algorithm unless $P = NP$ [5]. The problem may be mathematically modelled as follows:

$$Maximize \quad \sum_{i=1}^{n} x_i p_i; \tag{1}$$

$$Subject\ to \quad \sum_{i=1}^{n} x_i w_i \leq C,\ x_i \in \{0,1\},$$

$\forall i \in \{1, 2, \ldots, n\}$; where x_i takes values either 1 or 0 which represents the selection or rejection of the i^{th} item.

In recent years, many heuristic algorithms have been employed to solve KP01 problems: an ant colony optimization algorithm for KP01 proposed in [6] proposed ; a modified the parameters of the ant colony optimization model to adapt itself to KP01 problems proposed in [7]; a binary particle swarm optimization based on multi-mutation strategy to solve the knapsack problem proposed in [8] ; a quantum-inspired evolutionary algorithm for KP01 proposed in [9]; a schema-guiding evolutionary algorithm to solve KP01 problems proposed [10]; a global harmony search algorithm to solve KP01 proposed in [11]; a genetic algorithm (GA) for KP01 proposed in [12]; an improved GA with a dual population for KP01 proposed in [13]; a schema-modified operator to adjust the distribution of the population can be found in [10], an artificial chemical reaction optimization for KP01 proposed in [14].

Although many KP01 problems have been solved successfully by these methods, the research on them is still important, because some new and more difficult KP01 problems hidden

Received Jan 19, 2016; Revised Feb 11, 2016; Accepted Feb 27, 2016

in the real world have not yet been solved. Many algorithms provide possible solutions for some KP01 problems, but they may lose their efficiency on solving these problems due to their own disadvantages. For example, some methods proposed recently can only solve KP01 problems with very low dimension, but they may be unavailable to solve KP01 problems with high dimension.

Given the above consideration, we designed an particle swarm optimization with greedy strategy to solve KP01. The particle swarm optimization has a good searching ability that shows excellent operation in two important features of optimization metaheuristics: intensification and diversification[8, 15]. Beside, the greedy strategy in this research is used in one phase of repair function, but in another phase a randomly method is used which is proposed in [9]. The repair function mentioned in the paper adopts two advantages, the first is to make the algorithm have fast convergence by using a greedy strategy. The experimental results demonstrate the proposed algorithm is superior.

The rest of this paper is organized in sections: section 2. present previous algorithm for KP01, section 3. briefly gives the original framework of particle swarm optimization. Section 4. present the binary particle swarm optimization. Constraint handling techniques are described in section 5.. We survey the behavior of particle swarm optimization and compare the simulated results of the PSO in section 6.. We conclude this paper and suggest potential future work in section 7..

2. Related Works

2.1. Artificial chemical reaction optimization algorithm (ACROA)

The ACROA is a heuristic method proposed by Alatas in [16]. It inspired from the chemical reaction process. In the chemical reaction process, the system tend toward the highest entropy and the lowest enthalpy. The chemical reactions possess efficient objects, states, process, and events that can be designed as a computational method. Enthalpy or potential energy for minimization problem and entropy for maximization problem can be utilized as objective functions for the interested problem [16, 17].

3. Particle swarm optimization

The PSO conducts searches using a population of particles, a population of particles is randomly generated initially. The standard particle swarm optimizer maintains a swarm of particle that represent the potential solutions to problem on hand. Suppose that the search space is *D*-dimensional, and the position of *i*th particle of the swarm can be represented by a *D*-dimensional vector, x_i=$(x_{i1}, ..., x_{id}, ..., x_{iD})$. The velocity of the particle x_i can be represented by another *D*-dimensional vector v_i=$(v_{i1}, ..., v_{id}, ..., v_{iD})$. The best position previously visited by the *i*th particle is denoted as p_i=$(p_{i1}, ..., p_{id}, ..., p_{iD})$. In essence, the trajectory of each particle is updated according to its own flying experience as well as to that of the best particle in the swarm. The basic PSO algorithm can be described as:

$$v_{i,d}^{k+1} = w.v_{i,d}^{k} + c_1.r_1^k.(p_{i,d}^k - x_{i,d}^k) + c_2.r_2^k.(p_{g,d}^k - x_{i,d}^k) \quad (2)$$

$$x_{i,d}^{k+1} = x_{i,d}^{k} + v_{i,d}^{k+1} \quad (3)$$

where $v_{i,d}^k$ is *d*th dimension velocity of particle *i* in cycle *k*; $x_{i,d}^k$ is the *d*th dimension position of particle *i* in cycle *k*; $p_{i,d}^k$ is the *d*th dimension position of personal best (pbest) of particle *i* in cycle *k*; $p_{g,d}^k$ is the *d*th dimension position of global best particle (gbest) in cycle *k*; *w* is the inertia weight; c_1 is the cognitive weight and c_2 is a social weight; r_1 and r_2 are two random values uniformly distributed in the range of [0,1][8].

The pseudocode of the PSO is given in the algorithm 1.

Algorithm 1: PSO algorithm

```
Input: Initial parameters
Output: optimal solution
for each particle do
    Initialize particle
While maximum iterations or minimum error criteria is not attained for each particle do
    Calculate fitness value
    if the fitness value is better than its peronal best then
        set current value as the new pBest
Choose the particle with the best fitness value of all as gBest
for each particle do
    Calculate particle velocity according equation (2)
    Update particle position according equation (3)
return
```

Figure 1. Sigmoid function used in BPSO

4. Binary particle swarm optimization

The binary particle swarm optimization algorithm was introduced by Bansal and Deep to allow the PSO algorithm to operate in binary problem spaces [18]. It uses the concept of velocity as a probability that a bit (position) takes on one or zero. In the BPSO, Eq. (2) for updating the velocity remains unchanged, but Eq. (3) for updating the position is re-defined by the rule

$$x_{i,d}^{k+1} = \begin{cases} 0 & \text{if } rand() \geq S(v_{i,d}^{k+1}) \\ 1 & \text{if } rand() < S(v_{i,d}^{k+1}) \end{cases}$$

where S(.) is the sigmoid function for transforming the velocity to the probability as the following expression:

$$S(v_{i,d}^{k+1}) = \frac{1}{1 + e^{(v_{i,d}^{k+1})}} \tag{4}$$

Fig. 1 shows the sigmoid function using in BPSO.

5. Handling Constraints

The present by binary string sometimes make the solution violate the constraint. There are two common techniques that are penalty and repair function are used to handle it. In the first method, a penalty coefficient ratio with violated value is used to add to the fitness value. Through the iterations, the solutions with larger fitness have more change to reproduce, and otherwise [11]. Although, this method can help the algorithm can find the sufficient solution, but it do not helpful improve the quality of the solution. Following, two techniques are presented in details.

5.1. Penalty function

The KP01 is maximization problem. The value of the position is equal to $\sum_{i=1}^{n} x_i p_i$ when the solution is not violated. Otherwise, a $penaltyFactor$ is used to decrease the fitness of the violate position. In this research, we use $penaltyFactor = 100$. The fitness function is described in algorithm 2.

Algorithm 2: Fitness function

Input: Solution x
Output: $Fitness$
$Fitness = \sum_{i=1}^{n} x_i p_i - penaltyFactor * max(0, \sum_{i=1}^{n} x_i w_i - C)$
return *Fitness*

5.1.1. Greedy

The repair operator is based on repeated random selection until the knapsack constraints are met, although this may consume a lot of CPU time in some cases. Conversely, the traditional greedy strategy has some other drawbacks in the knapsack problem and is analyzed in [12]. In this paper, a new repair operator is used and it depends on both the greedy strategy and random selection [19]. The advantage of this repair procedure is the balance between CPU time cost and not getting stuck in local optima. The items are sorted according to the profit-to-weight ratio p_i/w_i (i = 1, 2,. . . , n) so that they are not increasing. It means that:

$$\frac{p_i}{w_i} \geq \frac{p_j}{w_j},\ for\, i < j.$$

This repair operator consists of two phases. The first phase (called ADD) examines each variable in decreasing order of p_j/w_j and changes the variable from zero to one as long as feasibility is not violated. The second phase (called DROP) examines each variable in increasing order of p_j/w_j and changes the variable from one to zero if feasibility is violated. The aim of the DROP phase is to obtain a feasible solution from an infeasible solution, whilst the ADD phase seeks to improve the fitness of a feasible solution. The pseudo-code for the repair operator is given in Algorithm 3.

Algorithm 3: Repair operator

Input: Solution x
Output: Solution x
% ADD phase
$gap \leftarrow C - \sum_{i=1}^{n} x_i w_i$
$i \leftarrow 1$
while $(gap > 0)\ and\ (i \leq n)$ **do**
  if $(gap \geq w_i)$ **then**
    $x_i \leftarrow 1$
    $gap \leftarrow gap - w_i$
    $i \leftarrow i + 1$
% DROP phase
$over \leftarrow \sum_{i=1}^{n} x_i w_i - C$
$i \leftarrow n$
while $(over > 0)\ and\ (i \geq 1)$ **do**
  if $(over \leq w_i)$ **then**
    $x_i \leftarrow 0$
    $over \leftarrow over - w_i$
    $i \leftarrow i - 1$
return x

Table 1. The dimension and parameters of five test problems.

Instance	Dimension	Parameters (q, C, p)
f_1	4	q = (6, 5, 9, 7), C = 20, p = (9, 11, 13,15)
f_2	10	q = (30, 25, 20,18, 17, 11, 5, 2, 1, 1), C = 60, p= (20, 18, 17, 15, 15, 10, 5, 3, 1, 1)
f_3	7	q = (31, 10, 20, 19, 4, 3, 6), C = 50, p = (70, 20, 39, 37, 7, 5, 10)
f_4	5	q = (15, 20, 17, 8, 31), C = 80, p = (33, 24, 36, 37, 12)
f_5	20	q = (84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92), C = 879, p = (91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44)

Table 2. The detailed information of the optimal solutions.

Instance	Opt.solution x^*	Opt.value f(x^*)	Value of constraint g(x^*)
f_1	(1,1,0,1)	35	-2
f_2	(0,0,1,0,1,1,1,1,0,0)	50	0
f_3	(1,0,0,1,0,0,0)	107	0
f_4	(1,1,1,1,0)	130	-20
f_5	(1,1,1,1,1,1,1,1,1,0,1,1,1,1,0,1,0,1,1,1)	1025	-8

6. Simulation Results

In this section, we use BPSO and BPSOG for binary particle swarm optimization with penalty constraint and greedy constraint techniques, respectively. The ACROA use the greedy constraint.

The performance of BPSOG algorithm is extensively investigated by a large number of experimental studies. Nine 0-1 knapsack instances are considered to testify the validity of the BPSOG.

All the algorithms are implemented in matlab 2014a. The test environment is set up on a laptop with core i5 M520 CPU at 2.4 GHz, 4G RAM, running on Windows 8.1.

6.1. The performance of three algorithms on solving 0-1 knapsack problems with small dimension sizes

In this section, five test functions collected from [11] are used. In Table 1, four test functions with dimension are 4, 10, 7, 5, and 20, respectively. Table 2 describes the optimal solutions of each function.

The experiment for these five test functions is run 25 independent times. To extend study the performance of BPSOG, four strong correlated instances with large dimension are also used.

6.2. The performance of three algorithms on solving 0-1 knapsack problems with large dimension sizes

To test the performance of BPSOG on KP01 with large dimension, it is compared with both BPSO and ACROA on the 0-1 knapsack problem. In these test cases, strongly correlated sets of data are considered. The weights w_i, respective prices p_i and the knapsack capacity *C* are calculated as follows:

$$w_i = \textit{rand}(1, 10); \tag{5}$$

Table 3. Experimental results: the number of items 50, 100, 500 and 1000, the maximum number of function evaluation 100000, the number of runs 25.

Instances	Algorithms	Best profit	Worst profit	Average profit	stDev
50	ACROA	1536	1507	1528.70	6.10
	BPSO	1533	1485	1501.20	17.64
	BPSOG	**1536**	**1536**	**1536.00**	**0.00**
100	ACROA	2927	2855	2893.76	18.73
	BPSO	2876	2792	2827.92	19.73
	BPSOG	**2978**	**2977**	**2977.96**	**0.20**
500	ACROA	14775	14428	14582.96	102.22
	BPSO	14152	13908	14017.44	56.61
	BPSOG	**15369**	**15234**	**15298.24**	**37.18**
1000	ACROA	28840	27961	28257.48	192.94
	BPSO	27332	27044	27173.92	88.74
	BPSOG	**30050**	**29712**	**29819.76**	**81.17**

Figure 2. The convergence curves of BPSO and BPSOG on the kp01.

$$p_i = w_i + 5, i = 1, 2, \ldots, n; \tag{6}$$

$$C = \frac{1}{2}\sum_{i=1}^{n} w_i; \tag{7}$$

where rand(1, 10) generates an integer in [1, 10] uniformly at random.

We do experiment on four test instances with 50, 100, 500 and 1000 items. Fig. 2 shows the convergence curves of the best profits of BPSO and BPSOG in the four instances. The BPSOG shows better diversification and intensification when it is fast convergence and finds out the better profit value compared with BPSO.

It indicates the global search ability and the convergence ability of BPSOG. BPSOG outperformed BPSO and ACROA in terms of convergence rate and profit amount.

As shown in Figs. 2, the BPSOG displays no premature convergence in average profits throughout the iterations. The BPSO show premature convergence compared with BPSOG in 500 items test instances. The BPSOG shows better diversification and intensification when it is fast convergence and finds out the better profit value compared with BPSO.

Table 3 shows the experimental results of the instances. We adopt the same termination criterion, and the function evaluation limit is set to 100000, for all the test. For all the instances, the BPSOG yields superior results compared with that of BPSO and ACROA. The series of experimental results demonstrate the superiority and effectiveness of BPSOG. In comparison with BPSO and ACROA; the experimental results show that BPSOG outperforms the other algorithms in both solution quality and computing time. The reason for this superior performance of BPSOG is that our proposed algorithm has a good search ability and a greedy repair operator.

The ACROA is coded as describing in [14]. ACROA there is only parameter $reactantNum$ and it is set to 10. There are many possible PSO parameter setting. In this study, the parameters for BPSO and BPSOG are setting as: inertia weight $w = 2$, local weight $c_1 = 2$, global weight $c_2 = 2$.

7. Conclusion

In this paper, a new algorithm has been proposed based on the binary particle swarm optimization with a greedy to solve 0-1 knapsack problem efficiently. Two constraint techniques based on penalty factor and greedy strategy is proposed to improve the efficiency of the proposed algorithm. The simulation results on five state of the art benchmark instances and strong correlated data sets demonstrate that the proposed algorithm has superior performance compared with previous algorithms. The new approach provides better quality solutions when solve large scale instances.

Acknowledgement

This work was partly supported by National Natural Science Foundations of China (No. 61301148 and No. 61272061), the fundamental research funds for the central universities of China (No.531107040263, 345 531107040276), the Research Funds for the Doctoral Program of Higher Education of China (No. 20120161110002 and No. 20130161120019), Hunan Natural Science Foundation of China (No. 14JJ7023).

References

References

[1] S. Martello, Knapsack Problem: Algorithms and Computer Implementations, John Wiley and Sons, New York, 1990.
[2] B. Chor, R. Rivest, A knapsack-type public key cryptosystem based on arithmetic in finite fields, Information Theory, IEEE Transactions on 34 (5) (1988) 901 –909.
[3] R. Goodman, A. McAuley, A new trapdoor knapsack public key cryptosystem, in: T. Beth, N. Cot, I. Ingemarsson (Eds.), Advances in Cryptology, Vol. 209 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 1985, pp. 150–158.
[4] C.-S. Laih, J.-Y. Lee, L. Harn, Y.-K. Su, Linearly shift knapsack public-key cryptosystem, Selected Areas in Communications, IEEE Journal on 7 (4) (1989) 534 – 539.
[5] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, America, 1979.
[6] C.-Y. Lee, Z.-J. Lee, S.-F. Su, A new approach for solving 0/1 knapsack problem, in: Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on, Vol. 4, 2006, pp. 3138 –3143.
[7] H. Shi, Solution to 0/1 knapsack problem based on improved ant colony algorithm, in: Information Acquisition, 2006 IEEE International Conference on, 2006, pp. 1062 –1066.
[8] Z. Li, N. Li, A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem, in: Proceedings of the 21st annual international conference on Chinese control and decision conference, CCDC'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 3090–3095.
[9] K.-H. Han, J.-H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, Evolutionary Computation, IEEE Transactions on 6 (6) (2002) 580 – 593.
[10] Y. Liu, C. Liu, A schema-guiding evolutionary algorithm for 0-1 knapsack problem, in: Computer Science and Information Technology - Spring Conference, 2009. IACSITSC '09. International Association of, 2009, pp. 160 –164.
[11] D. Zou, L. Gao, S. Li, J. Wu, Solving 01 knapsack problem by a novel global harmony search algorithm, Applied Soft Computing 11 (2) (2011) 1556 – 1564, the Impact of Soft Computing for the Progress of Artificial Intelligence.
[12] J. Zhao, T. Huang, F. Pang, Y. Liu, Genetic algorithm based on greedy strategy in the 0-1 knapsack problem, in: Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on, 2009, pp. 105 –107.
[13] W. Shen, B. Xu, J. ping Huang, An improved genetic algorithm for 0-1 knapsack problems, in: Networking and Distributed Computing (ICNDC), 2011 Second International Conference on, 2011, pp. 32 –35.

[14] T. K. Truong, K. Li, Y. Xu, A. Ouyang, T. T. Nguyen, Solving 0–1 knapsack problem by artificial chemical reaction optimization algorithm with a greedy strategy, Journal of Intelligent and Fuzzy Systems 8 (5) (2015) 2179–2186.
[15] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm intelligence 1 (1) (2007) 33–57.
[16] B. Alatas, Acroa: Artificial chemical reaction optimization algorithm for global optimization, Expert Systems with Applications 38 (10) (2011) 13170 – 13180.
[17] B. Alatas, A novel chemistry based metaheuristic optimization method for mining of classification rules, Expert Systems with Applications 39 (12) (2012) 11080 – 11088.
[18] J. C. Bansal, K. Deep, A modified binary particle swarm optimization for knapsack problems, Applied Mathematics and Computation 218 (22) (2012) 11042 – 11061.
[19] T. K. Truong, K. Li, Y. Xu, Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem, Applied Soft Computing 13 (4) (2013) 1774–1780.