Advances in Computer Science Research, volume 96
Proceedings of the International Conference on Mathematics, Geometry, Statistics, and Computation (IC-MaGeStiC 2021)

Hybrid Cat-Particle Swarm Optimization Algorithm on Bounded Knapsack Problem with Multiple Constraints

Kiswara Agung Santoso*, Muhammad Bagus Kurniawan, Ahmad Kamsyakawuni, Abduh Riski

Department of Mathematics, FMIPA, University of Jember
**Corresponding author. Email: kiswaras@gmail.com*

ABSTRACT
Optimization problems have become interesting problems to discuss, including the knapsack problem. There are many types and variations of knapsack problems. In this paper, the authors introduce a new hybrid metaheuristic algorithm to solve the modified bounded knapsack problem with multiple constraints we call it modified bounded knapsack problem with multiple constraints (MBKP-MC). Authors combine two popular metaheuristic algorithms, Particle Swarm Optimization (PSO) and Cat Swarm Optimization (CSO). The algorithm is named Hybrid Cat-Particle Swarm Optimization (HCPSO). The results of the implementation of the algorithm are compared with PSO and CSO algorithms. Based on the experimental results, it is known that the HCPSO algorithm is suitable and can reach to good-quality solution within a reasonable computation time. In addition, the new proposed algorithm performs better than the PSO and CSO on all MBKP-MC data used.

Keywords: *Hybrid cat-particle swarm optimization, Metaheuristic, Modified bounded knapsack problem.*

1. INTRODUCTION

The knapsack problem is one of the optimization problems that are very interesting to be discussed. According to [1], the knapsack problem is defined as a problem that we have to determine some items to be put into storage media, where each item has weight and profit. The total weight must not exceed the capacity of the storage media used. The items must be selected such that the total profit can be maximized. Knapsack Problem (KP) has some types based on the problem, which are 0-1 knapsack, bounded knapsack, and unbounded knapsack [2]. Based on the objective, a number of constraints, and a number of storage media, the knapsack problem is divided into variations: single objective knapsack, multi-objective knapsack, multidimensional knapsack, multiple constraints knapsack, and quadratic knapsack [3]. There are some researches discuss the combination of the type and variation of the knapsack problem, such as multiple constraints 0-1 knapsack problem [4] [5], multidimensional 0-1 knapsack problem [6], multidimensional 0-1 knapsack problem with multiple constraints [7, 8], multidimensional bounded knapsack problem [9, 10], multiple constraints bounded knapsack problem [11]. In this study, we will discuss a new hybrid metaheuristic algorithm. In Bounded Knapsack Problem (BKP), there is a maximum number of each item. In Modified Bounded Knapsack Problem (MBKP), we add a minimum number of each item that should be put into the storage media. And then, in this algorithm that we will discuss, we use three main constraints, weight, volume, and price. The total weight and volume must not exceed the capacity of the storage media, and the total price should not be greater than the capital. The optimization problem, included the knapsack problem, can be solved by some methods or algorithms. According to [12], a metaheuristic algorithm is an algorithm designed to solve optimization problems through an approach process that is inspired by nature, such as biology, physic, or ethology.

There are some metaheuristic algorithms that often used by some researchers, such as Particle Swarm Optimization (PSO) [13], Cat Swarm Optimization (CSO) [14], Genetic Algorithm (GA) [15, 16], Simulated Annealing (SA) [17, 18], Artificial Immune System (AIS) [19, 20], Ant Colony Optimization (ACO) [21, 22] and Tabu Search (TS) [23 – 25]. In this paper, we will focus on PSO and CSO algorithms. The PSO algorithm is known as the metaheuristic algorithm introduced by Eberhart and Kennedy in 1995, which is inspired by the

behavior of bird or fish swarm [13]. The PSO algorithm was first used to solve function optimization. After that, there are many other optimization problems solved using PSO algorithm, such as dynamic systems [26], pattern-matching task [21], traveling salesman problem [23], hybrid flow-shop scheduling [9], nonlinear differential equations [12], vehicle routing problem [3], vehicle routing problem with time windows [23]. The CSO algorithm was first proposed by Chu, et al. in 2006 [14] that is based on cat behaviors. There are two modes in the CSO algorithm for exploration and exploitation, that is seeking and tracing. Some researchers applied the CSO algorithm in many fields, such as function optimization [14], clustering [3], reliability-oriented task allocation [12], unconstrained optimization [5], artificial neural networks [16], 0-1 knapsack [11], workflow scheduling [18], non-unicast set covering [19]. Besides those two algorithms, many researchers have combined some algorithms as a hybrid. According to [20], the hybrid algorithm is more efficient for solving large-scale problems. A hybrid algorithm is also able to balance the exploration and exploitation, and able to balance the execution time and the quality of the final result.

In this paper, we will combine PSO and CSO algorithms as Hybrid Cat-Particle Swarm Optimization (HCPSO) algorithm. The algorithm will be tested for solving MBKP-MC. The performance of the algorithms is also compared with the results of the PSO and CSO algorithms.

2. MODIFIED BOUNDED KNAPSACK

The Bounded Knapsack Problem (BKP) [3] is defined as follows. A set of items j = {1, 2, …, n} where each item has a weight w_j and a profit p_j. The problem that must be solved is how to choose items with some constrain so that the total profit can be maximized. The models which describe this problem can be written as formula below (BKP)

$$\boldsymbol{Maximize} \sum_{j=1}^{n} \boldsymbol{p_j x_j} \quad (1)$$

$$with\ constrain \sum_{j=1}^{n} w_j x_j \le C \quad (2)$$

$$0 \le x_j \le b_j, x_j \text{ integer}, j = 1, 2, \ldots, n \quad (3)$$

In this paper, we use Modified Bounded Knapsack Problem with Multiple Constraints (MBKP-MC). Besides the maximum number b_j of the identical copies of item j, we consider the minimum number a_j of each item type that should be selected. We also use three main constraints, which are weight (w_j), volume (v_j), and buy price/cost (c_j). The total weight and volume must not exceed the capacity of the storage media, and the total price should not be greater than the capital. Therefore, the problem is formalized as follows.

$$\boldsymbol{Maximize} \sum_{j=1}^{n} \boldsymbol{p_j x_j} \quad (4)$$

$$\sum_{j=1}^{n} w_j x_j \le C \quad (5)$$

$$\sum_{j=1}^{n} v_j x_j \le S \quad (6)$$

$$\sum_{j=1}^{n} \mathrm{w}_j x_j \le M \quad (7)$$

$$0 \le x_j \le b_j, x_j \text{ integer}, j = 1, 2, \ldots, n \quad (8)$$

where C, S and M are the weight capacity, space/volume capacity and capital respectively.

3. THE PROPOSED ALGORITHM

3.1 The PSO Algorithm

The Particle Swarm Optimization (PSO) algorithm is one of the metaheuristic algorithms first introduced by Russell Eberhart and James Kennedy in 1995 [13]. The PSO algorithm is based on a swarm of birds or fish in nature. The PSO algorithm has been applied in most optimization areas, intelligent computation, and design/scheduling [21]. In the PSO algorithm, there are N particles and each particle represents a candidate of solution. The PSO algorithm generates the initial position, and velocity for every particle randomly in search space. Every iteration, each particle updates their velocity and position based on their own best position P_i and global best position P_g. According to the Invalid source specified, the particle's velocity is updated using Equation (9), and then the position is updated using Equation (10).

$$V_i(t+1) = \omega V_i(t) + c_1 r_1 \left(P_g(t) - X_i(t)\right) + c_2 r_2 \left(P_i(t) - X_i(t)\right) \quad (9)$$

where c_1, c_2 are coefficient of acceleration in the interval [0, 2]; r_1, r_2 are random variables with uniform distribution in the interval [0, 1]; ω is inertia weight used to balance the global and local search; and t, $t + 1$ are current and next iteration respectively.

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (10)$$

The inertia weight ω is linearly decreased

$$\omega = \omega_{max} - t * \frac{\omega_{max} - \omega_{min}}{NumIter} \quad (11)$$

where ω_{max}, ω_{min}, respectively, are maximum and minimum weight, and $NumIter$ is a number of iterations.

3.2 The CSO Algorithm

Hunting ability to survive, but a house cat has a strong instinct interest in moving things. Although cats spend a lot of time sleeping, they have high alertness. They keep paying attention around to know if there is food or danger. Based on those two behaviors, the CSO algorithm is divided into two modes, seeking mode and tracing mode. The algorithm uses a mixture ratio (MR) to split the cats into seeking mode or tracing mode.

The seeking mode is used to model the situation while cats are sleeping. In seeking mode, there are four factors defined, seeking memory pool (SMP), seeking ranger of selected dimension (SRD), counts of dimension to change (CDC), and self-position considering (SPC).

SMP represents the size of the searching memory of each cat. Every cat will choose a new position in memory using Roulette Wheel. SRD is used to control changes in values of the selected dimension, which do not exceed the range. CDC is used to determine the number of dimensions to change. SPC is a Boolean variable. If the cat is the fittest individual, then SPC is true (1), otherwise, SPC is false (0). If SPC is true (1), the cat will create j copies new candidate position, where $j = SMP$, but if SPC is false (0), then the cat creates $j = SMP$ one copies and one candidate is the current position.

The tracing mode is used in the model cases where cats trace the target or prey. Every cat in tracing mode updated their position based on the velocity. In the CSO algorithm, the velocity of cats is updated based on the best position. The procedure of the CSO algorithm is described into eight steps as follows.

1. Generate initial position and velocity of N cats in search space.
2. Evaluate the fitness value and save the best position as $xbest$.
3. Divide the cats into seeking and tracing mode.
4. If the cat is in seeking mode, create a new candidate position using Equation (12), and then choose one to replace the current position using Roulette Wheel.

$$x'_{j,d} = x_{j,d} \pm SRD * r * x_{j,d} \quad (12)$$

where r is a random number and $x_{j,d}$, $x'_{j,d}$ are current and new values of dimension d respectively.

5. If the cat is in tracing mode, update the velocity using Equation (13), and then update the position using Equation (14)

$$v_{k,d}(t+1) = v_{k,d}(t) + c_1 r_1 \left(x_{best,d}(t) - x_{k,d}(t)\right) \quad (13)$$

$$x_{k,d}(t+1) = x_{k,d}(t) + v_{k,d}(t+1) \quad (14)$$
$$d = 1, 2, \dots, D$$

where c_1 is acceleration coefficient; r_1 is a random number; $v_{k,d}(t)$, $v_{k,d}(t+1)$, respectively are current and new velocity; and $x_{k,d}(t)$, $x_{k,d}(t+1)$ are current and new position respectively.

6. Merge the cats from seeking and tracing mode.
7. Evaluate the fitness value and update the best position.
8. Check the termination criterion. If the criterion is reached, then the algorithm is stopped; else, back to step 3.

3.3 The HCPSO Algorithm

The Hybrid Cat-Particle Swarm Optimization (HCPSO) algorithm is the new hybrid algorithm we propose in this paper. We combine the CSO and PSO that are known as good metaheuristic algorithms. In the HCPSO algorithm, we use the full CSO scheme process with some additions and modifications. The algorithm saves the global best position and local best position like the PSO algorithm. And then, in the tracing mode, we apply the movement formula of PSO. Besides that, in the seeking mode, we use a modification of the value of the selected dimension based on the best position, and then we choose the best new candidate to replace the current position. This hybridization aims to get a better algorithm with a better convergence, and the execution time does not increase. All steps of the HCSPO algorithm to solve MBKP-MC are described as follows.

1. Generate initial position (X) of N individuals in search space ([0, 1]) and generate velocity (V).

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}, x_{kd} \in [0,1] \quad (15)$$

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1D} \\ v_{21} & v_{22} & \cdots & v_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ v_{N1} & v_{N2} & \cdots & v_{ND} \end{bmatrix} \quad (16)$$

where D is the number of items types.

2. Convert the position (X) into MBKP-MC solution term (Y) using Equation (18).

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1D} \\ y_{21} & y_{22} & \cdots & y_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{ND} \end{bmatrix} \quad (17)$$

$$y_{kd} = round(x_{kd} * (b_d - a_d)) \quad (18)$$

3. Check all the constraints. Make sure that all the solutions are an infeasible area which means all the solutions must meet the MBKP-MC constraints.
4. Evaluate the fitness value (total profit) of all solutions.
5. Save the best position of each individual as C_k and the best global solution as C_g.
6. Divide the individuals into seeking and tracing modes.
7. If individuals are in seeking mode. Create copies based on their own the best position C_k Equation (19) and modify the selected dimension based on the best global solution C_g Equation (20).

$$x'_{j,d} = C_{k,d},\ d = 1,2,\dots,D \quad (19)$$

$$x'_{j,d} = C_{g,d} \pm SRD * r * C_{g,d} \quad (20)$$

8. If individuals are in tracing mode. Update the velocity and position based on PSO movement as formulated in Equation (21) - Equation (22)

$$V_i(t+1) = \omega V_i(t) + c_1 r_1 \left(C_g(t) - X_i(t)\right) + c_2 r_2 \left(C_i(t) - X_i(t)\right) \quad (21)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (22)$$

9. Merge the cats from seeking and tracing mode and ensure all the positions do not exceed search space [0, 1]. If the solution exceeds the search space, then the solution must be transformed using Equation (23).

$$x_k = \begin{cases} \frac{x_k - \min(x_k)}{\max(x_k) - \min(x_k)}, & if\ \min(x_k) < 0 \\ \frac{x_k}{\max(x_k)}, & if\ \max(x_k) > 1 \end{cases} \quad (23)$$

10. Convert the new position (X) into the MBKP-MC solution term (Y). Check all the constraints and then evaluate the fitness value.
11. Update the best individual position C_k and the best global position C_g.
12. Check the termination criterion. If the criterion is reached, then the algorithm is stopped and the final solution is C_g in MBKP-MC solution term. But, if the criterion is not reached, go back to step 6.

4. COMPUTATIONAL RESULTS

Experimental Design

In order to study the performance of the proposed algorithm of this paper, we generate 20 MBKP-MC Instances. The test instances involve different size of problem. The numbers of items types (n) that we use are 30, 40, 50, 75 and 100. We apply simplex method to know the optimal value of the generated Instances. In particular, we use the following parameter setting: N = 100, $NumIter$ = 5000, MR = 0.5, SMP = 5, CDC = 0.2 (30, 40, 50 items), CDC = 0.1 (75 items), CDC = 0.08 (100 items), SRD = 0.1, $c1$ = 0.5, $c2$ = 1.5, ωmax = 0.9 and ωmin = 0.7.

Table 1. The average results obtained

Table 2.Problem Set	Opt	HCPSO		PSO		CSO	
		Z	Time	Z	Time	Z	Time
Mbkp-mc_30_1	1095500	1095500	49.861	1095500	16.7622	1082000	57.9483
Mbkp-mc_30_2	1174900	1174200	50.4247	1173500	16.91	1159500	56.1688
Mbkp-mc_30_3	997100	996300	51.0127	995700	16.8141	987900	58.3414
Mbkp-mc_30_4	1300800	1300800	47.6789	1300800	16.3793	1273100	56.1159

The average results on Table 1, that the HCPSO algorithm is the best algorithm, followed by PSO and then CSO. The PSO algorithm has the shortest execution time. Although, the average execution time of the HCPSO algorithm is less than CSO on all problem sets. These results mean that the HCPSO algorithm is suitable and can reach to good-quality solution within a reasonable computation time. The formula (24) is used to calculate the percentage of deviation (PD) and evaluate the improvement rates of the approach. The smaller PD indicates a better result because it approaches the optimal value.

$$PD = \frac{Opt - Z}{Opt} x100\% \quad (24)$$

where Opt is the optimal value and Z denotes the result of the algorithm.

The percentage of deviation of the algorithms for each problem is shown in Table 2.

Table 2. The percentage of deviation

Problem Set	PD		
	HCPSO	PSO	CSO
Mbkp-mc_30_1	0.058421	0.181652	1.853035
Mbkp-mc_30_2	0.175334	0.433228	2.279343
Mbkp-mc_30_3	0.143416	0.405175	2.321733
Mbkp-mc_30_4	0.063807	0.164514	2.805197
Average	**0.371218**	**0.984129**	**3.359516**

Based on the percentage of deviation as we can see in Table 2, the HCPSO algorithm has also the smallest average compared with PSO and CSO. We can also know that the larger problem set doesn't affect the percentage of deviation HCPSO algorithm. It is different with PSO and CSO; the percentage of deviation of both algorithms increases on larger problem sets. Thus, we can use the Hybrid Cat-Particle Swarm Optimization to solve large-sized modified bounded knapsack problems with multiple constraints.

5. CONCLUSIONS

In this paper, the Hybrid Cat-Particle Swarm Optimization (HCPSO) as a combination of Particle Swarm Optimization (PSO) and Cat Swarm Optimization (CSO) algorithms, has been proposed to solve modified bounded knapsack problem with multiple constraints (MBKP-MC). This hybridization aims to accelerate the convergence to the optimal value. The proposed algorithm has been applied to the MBKP-MC data and the results have been compared with the PSO and CSO algorithms. Based on the experimental results, it is known that the HCPSO algorithm provides better performance in the two mentioned algorithms on all problem sets. Thus, it makes the HCPSO algorithm convenient to use.

REFERENCES

[1] A. Gherboudj, A. Layeb and S. Chikhi, Solving 0-1 Knapsack Problem by A Discrete Binary Version of Cuckoo Search Algorithm, International Journal of Bio-Inspired Computation, 2012, vol. 4, no. 4, pp. 229-236.

[2] D. Pisinger, A Minimal Algorithm for the Multiple-Knapsack Problem, European Journal of Operational Research, 1995, vol. 83, no. 2, pp. 394-410.

[3] H. Kellerer, U. Pferschy and D. Pisinger, Knapsack Problem, Berlin: Springer, 2003.

[4] D. K. Agrafiotis and W. Cedeno, Feature Selection for Structure-Activity Correlation Using Binary Particle Swarms, Journal of Medicinal Chemistry, 2002, vol. 45, no. 5, pp. 1098-1107.

[5] M. Clerc, Discrete Particle Swarm Optimization, Illustrated by the Traveling Salesman Problem, in New Optimization Technique in Engineering, Berlin, Heidelberg, Springer, 2004, pp. 2019-239.

[6] C. T. Tseng and C. J. Liao, A Particle Swarm Optimization Algorithm for Hybrid Flow-Shop Scheduling with Multiprocessor Tasks, International Journal of Production Research, 2008, vol. 46, no. 17, pp. 4655-4670.

[7] M. Babaei, A General Approach to Approximate Solutions of Nonlinear Differential Equations Using Particle Swarm Optimization, Applied Soft Computing, 2013, vol. 13, no. 7, pp. 3354-3365.

[8] C. Pornsing, A Particle Swarm Optimization for Vehicle Routing Problem, Rhode Island University of Rhode Island, 2014.

[9] Z. M. Fatimah, Penerapan Algoritma Particle Swarm Optimization untuk Vehicle Routing Problem with Time Windows pada Kasus Pendistribusian Barang, Universitas Jember, Jember, 2016.

[10] B. Santosa and M. K. Ningrum, Cat Swarm Optimization for Clustering, International Conference of Soft Computing and Pattern Recognition, 2009, pp. 54-59.

[11] R. Shojaee, H. R. Faragardi, S. Alaee, and N. Yazdani, "A New Cat Swarm Optimization based Algorithm for Reliability-Oriented Task Allocation in Distributed Systems," 6th International Symposium on Telecommunications (IST), 2012, pp. 861-866.

[12] I. Boussaid, J. Lepagnot, and P. Siarry, A Survey on Optimization Metaheuristics, Information Sciences, 2013, vol. 237, pp. 82-117.

[13] R. Eberhart and J. Kennedy, Particle Swarm Optimization, Proceeding of the IEEE International Conference on Neural Networks, 195, vol. 4, pp. 1942-1948.

[14] S. C. Chu, P. W. Tsai, and J. S. Pan, Cat Swarm Optimization, Pacific Rim International Conference on Artificial Intelligence, 2006, pp. 854-858.

[15] P. C. Chu and J. E. Beasley, A Genetic Algorithm for the Multi Dimension Knapsack Problem, Journal of Heuristics, 1998, vol. 4, no. 1, pp. 63-86.

[16] L. V. Snyder and M. S. Daskin, A Random-Key Genetic Algorithm for the Generalized Traveling Salesman Problem, European Journal of Operational Research, 2006, vol. 174, pp. 38-53.

[17] D. R. Din, Heuristic and Simulated Annealing Algorithms for Wireless ATM Backbone Network Design Problem, Journal of Information Science & Engineering, 2008, vol. 24, no. 2.

[18] M. Andresen, H. Brasel, M. Morig, J. Tusch, F. Werner and P. Willenius, Simulated Annealing and Genetic Algorithms for Minimizing Mean Flow Time in an Open Shop, Mathematical and Computer Modelling, 2008, vol. 48, no. 7-8, pp. 1279-1293.

[19] O. Engin and A. Doyen, A New Approach to Solve Hybrid Flow Shop Scheduling Problems by Artificial Immune System, Future Generation Computer Systems, 2004, vol. 20, pp. 10831095.

[20] C. A. C. Coello and N. C. Cortes, Solving Multiobjective Optimization Problems Using an Artificial Immune System, Genetic Programming and Evolvable Machines, 2005, vol.6, pp.163-190.

[21] M. Kong, P. Tian, and Y. Kao, A New Ant Colony Optimization Algorithm for the Multi Dimension Knapsack Problem, Computers & Operations Research, 2008, vol. 35, no. 8, pp. 2672-2683.

[22] M. Mavrovouniotis, F. M. Muller and S. Yang, Ant Colony Optimization with Local Search for Dynamic Traveling Salesman Problems, IEEE Transactions on Cybernetics, 2016, vol. 47.

[23] V. Pureza and P. M. Franca, Vehicle Routing Problems via Tabu Search Metaheuristic, Centre De Recherche Sur Les Transports Publication, 1991.

[24] W. C. Chiang and R. A. Russell, A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows, INFORMS Journal on Computing, 1997, vol.9, no. 4, pp. 417-430.

[25] F. Glover and M. Laguna, Tabu Search, in Handbook of Combinatorial Optimization, Boston, MA, Springer, 1998, pp. 2093-2229.

[26] X. Hu and R. C. Eberhart, Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems, Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02 (Cat. No. 02TH8600), IEEE, 2002, vol. 2.