Journal of Computer and Knowledge Engineering, Vol.4, No.2. 2021. (11-16)
DOI: 10.22067/cke.2021.63250.0

Application of Black Hole Algorithm for Solving Knapsack Problems*

Short Paper

Abdolreza Hatamlou[1]

Abstract: This study investigates the application of the Black Hole algorithm (BH) for solving 0–1 knapsack problems. Knapsack problem is a classic and famous problem for testing and analyzing the behavior of optimization and meta-heuristic algorithms. There is no single algorithm which is suitable for all types of the knapsack problem. So it is an open research area to solve knapsack problem using novel optimization algorithms efficiently. BH algorithm is one of the most recent nature-inspired algorithms that is inspired by the black hole phenomenon. Like other population-based algorithms, the black hole algorithm starts with an initial population of candidate solutions to an optimization problem and an objective function that is calculated for them. At each iteration of the Black hole algorithm, the best candidate is selected to be the black hole, and others called stars. If a star gets too close to the black hole, it will be swallowed by the black hole and is gone forever. Computational experiments with a set of large-scale instances show that the BH algorithm can be an efficient alternative for solving 0–1 knapsack problems. The results show that the algorithm can find high quality solutions in less time compared to similar meta-heuristic approaches. Based on the obtained results it is clear that BH algorithm is a stable algorithm as the standard deviation of finding solutions in different runs is smaller than other test algorithms.

Keywords: Knapsack Problems, Black Hole Algorithm, Optimization.

1. Introduction

The knapsack problem is one of the classical NP-hard problems and it has been thoroughly studied in the last few decades. It has many applications in different fields, such as project selection and investment decision-making, marketing, chemistry, information technology, portfolio optimization, optimal search strategies, production planning, logistics, and statistical sampling. The knapsack problem is a set of items that each has a specific weight and value. The goal is to select a number of items, so that the weight of selected objects is smaller or equal to the specified capacity for knapsack and the maximum value. In the general case, the most precious object is chosen to put in our knapsack. The problem often arises in resource allocation where there are financial constraints, and is studied in fields such as combinatory, computer science, complexity theory, cryptography and applied mathematics. A variety of knapsack problems occur, depending on the distribution of items and knapsacks [1-7].

The 0-1 knapsack problem is one of the most common types, in which there are N item. And the ith object weights w_i and values v_i and the knapsack has the capacity W. Each item may be chosen at most once. x_i is equal to 1, when an item is selected and otherwise it is equal to 0. Mathematically, the problem can be formulated as follows:

$$\text{Max} \ \sum_{i=1}^{n} v_i x_i \quad , \quad \text{s.t.} \ \sum_{i=1}^{n} w_i x_i \leq W \, , x_i \in \{0,1\} \, (i = 1, \dots, N) \tag{1}$$

Moreover, there are other kinds of knapsack problems that occur by changing the number of some problem parameter such as the number of items, number of objectives, or even the number of knapsacks: Multidimensional Knapsack Problems, Multiple Knapsack Problems, The Multiple-Choice Knapsack Problem, The Quadratic Knapsack Problem, Bounded Knapsack Problem, Unbounded Knapsack Problem, Nonlinear Knapsack Problems and etc. In this study, the model of 0-1 knapsack problem is used.

Recently meta-heuristic approaches have been studied and applied in different areas and applications. It has been shown that they have good performance in solving complicated problems with large search space [8-16]. Most of them in recent years have been used to solve knapsack problems: [17] proposes a novel global harmony search algorithm (NGHS) to solve 0–1 knapsack problems. [18] investigates solving the knapsack problem with imprecise weight coefficients using genetic algorithms. [19] proposes a novel ACO algorithm for the multidimensional knapsack problems (MKP). [20] presents an artificial bee colony (ABC) algorithm for the 0-1 Multidimensional Knapsack Problem (MKP_01) and [21] proposes a new hybrid approach combining artificial bee colony algorithm with a greedy heuristic and a local search for the quadratic knapsack problem. Particle swarm optimization algorithm was proposed to solve the Multidimensional Knapsack Problem (MKP) by [22] and in order to deal with binary optimization problems, discrete binary cuckoo search (BCS) algorithm was used by [23]. Many of these techniques could solve knapsack problems successfully.

The rest of paper is organized as follows: Section 2 presents an explanation of the black hole algorithm. Section 3 discusses the simulation results using thirteen datasets. Section 4 ends the paper with conclusions.

* Manuscript received, March, 31, 2020; accepted, July, 4, 2021.
[1] Associate Professor, Department of Computer Science, Khoy Branch, Islamic Azad University, Khoy, Iran.
Email: hatamlou@iaukhoy.ac.ir.

1. Black hole algorithm

The BH algorithm is a population-based method that has some common features with other population-based methods [24]. As with other population-based algorithms, a population of candidate solutions to a given problem is generated and distributed randomly in the search space. The population-based algorithms evolve the created population towards the optimal solution via certain mechanisms. For example, in GAs, the evolving is done by mutation and crossover operations. In PSO, this is done by moving the candidate solutions around in the search space using the best found locations, which are updated as better locations are found by the candidates. In the proposed BH algorithm the evolving of the population is done by moving all the candidates towards the best candidate in each iteration, namely, the black hole, and replacing those candidates that enter within the range of the black hole by newly generated candidates in the search space. In the BH algorithm the best candidate among all the candidates at each iteration is selected as a black hole and all other candidates form the normal stars. The creation of the black hole is not random. It is one of the real candidates of the population. Then, all the candidates are moved towards the black hole based on their current location and a random number. The details of the BH algorithms are as follows:

Like other population-based algorithms, in the black hole algorithm (BH) a randomly generated population of candidate solutions – the stars – are placed in the search space of some problem or function. After initialization, the fitness values of the population are evaluated and the best candidate in the population that has the best fitness value is selected to be the black hole and the rest form the normal stars. The black hole has the ability to absorb the stars that surround it.

After initializing the black hole and stars, the black hole starts absorbing the stars around it and all the stars start moving towards the black hole. The absorption of stars by the black hole is formulated as follows:

$$x_i(t+1) = x_i(t) + rand \times \left(x_{BH} -, x_i(t)\right) \; i = 1,2, \dots, N \tag{2}$$

where $x_i(t)$ and $x_i(t+1)$ are the locations of the ith star at iterations t and $+1$, respectively. x_{BH} is the location of the black hole in the search space. $rand$ is a random number in the interval [0, 1]. N is the number of stars (candidate solutions).

While moving towards the black hole, a star may reach a location with lower cost than the black hole. In such a case, the black hole moves to the location of that star and vice versa. Then the BH algorithm will continue with the black hole in the new location and then stars start moving towards this new location.

In addition, there is the probability of crossing the event horizon during moving stars towards the black hole. Every star (candidate solution) that crosses the event horizon of the black hole will be sucked by the black hole. Every time a candidate (star) dies – it is sucked in by the black hole – another candidate solution (star) is born and distributed randomly in the search space and starts a new search. This is done to keep the number of candidate solutions constant. The next iteration takes place after all the stars have been moved. The radius of the event horizon in the black hole algorithm is calculated using the following equation:

$$R = \frac{f_{BH}}{\sum_{i=1}^{N} f_i} \tag{3}$$

where f_{BH} is the fitness value of the black hole and f_i is the fitness value of the ith star. N is the number of stars (candidate solutions). When the distance between a candidate solution and the black hole (best candidate) is less than R, that candidate is collapsed and a new candidate is created and distributed randomly in the search space.

Based on the above description the pseudo code of the BH algorithm is summarized as follows:

- *Initialize a population of stars with random locations in the search space.*
- ***Loop***
- *For each star, evaluate the objective function.*
- *Select the best star that has the best fitness value as the black hole.*
- *Change the location of each star according to equation 3.*
- *If a star reaches a location with lower cost than the black hole, exchange their locations.*
- *If a star crosses the event horizon of the black hole, replace it with a new star in a random location in the search space.*
- *If a termination criterion (a maximum number of iterations or a sufficiently good fitness) is met, exit the loop.*
- ***End loop***

2. Experimental results

Seventeen samples with different numbers of items are used to evaluate and compare the performance of the proposed approach. The 10 data set extracted of [25] and F11, F12, and F13 are generated randomly. Moreover, the large scale problem F14 [26], F15 [27], F16 [27], and F17 [28] include 50, 50, 80, and 100 items, respectively. The corresponding maximum capacities of the knapsacks are 1000, 959, 1173, and 6718, respectively. All of computations were performed in MATLAB programming language environment. Table 1 summarizes the main features of F1-F13 problems such as profit v, weight w and capacity.

Black hole was compared with PSO, ACO and GA algorithms in order for better consideration. The population size for all algorithms was set at 60. The max generation of each run is 100. For GA, mutation rate was set to be 0.02. For PSO, the learning rate parameters were set to the values $c1$= $c2$=2 and the inertia weight w=1 [29]. The results obtained by four algorithms are presented in Table 2. Second column contains the best, worst, mean (average), standard deviation (std.dev) of solutions in 30 runs. Also, the run time for each algorithm in all datasets represented in this column. Columns third to sixth show the algorithms.

Table 1. The dimension and parameters of 13 test problems

instance	Dimension	parameters (v, w, W)
F1	4	v=(9,11,13,15) , w=(6,5,9,7) , W=20
F2	4	v=(6,10,12,13) , w=(2,4,6,7) , W=11
F3	5	v=(33,24,36,37,12) , w=(15,20,17,8,31) , W=80
F4	7	v=(70,20,39,37,7,5,10) , w=(31,10,20,19,4,3,6) , W=50
F5	10	v=(55,10,47,5,4,50,8,61,85,87) , w=(95,4,60,32,23,72,80,62,65,46) , W=269
F6	10	v=(20,18,17,15,15,10,5,3,1,1) , w=(30,25,20,18,17,11,5,2,1,1) , W=60
F7	15	v=(0.125126, 19.330424,58.500931, 35.029145, 82.284005,17.410810, 71.050142, 30.399487,9.140294, 14.731285, 98.852504,11.908322, 0.891140, 53.166295,60.176397) w=(56.358531, 80.874050, 47.987304,89.596240, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204,16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575) , W=375
F8	20	v=(44, 46, 90, 72, 91, 40, 75,35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29,75, 63) , w=(92, 4, 43, 83, 84, 68, 92, 82, 6, 44,32, 18, 56, 83, 25, 96, 70, 48, 14, 58) , W=878
F9	20	v=(91, 72, 90, 46, 55, 8, 35, 75,61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40,44) w=(84, 83, 43, 4, 44, 6, 82, 92, 25, 83,56, 18, 58, 14, 48, 70, 96, 32, 68, 92) W=879
F10	23	v=(981, 980, 979, 978, 977,976, 487, 974, 970, 485, 485, 970, 970,484, 484, 976, 974, 482, 962, 961, 959,958, 857) w=(983, 982, 981, 980, 979, 978, 488,976, 972, 486, 486, 972, 972, 485, 485,969, 966, 483, 964, 963, 961, 958, 959) , W=10000
F11	6	v=(8,9,11,4,3,12) , w=(39,20,20,25,38,32) , W=110
F12	12	v=(416,376,370,357,401,426,429,366,391,428,356,358) w=(31,21,26,26,26,33,37,27,27,36,34,31) , W=300
F13	30	v=(54,52,51,64,51,45,42,68,62,56,66,46,68,50,38,51,58,55,59,65,49,39,43,44, 49,55,59,38,51,58) w=(153,237,253,252,168,257,170,224,254,224,161,257,216,263,188,198, 194,220,163,230,225,174,232,229,265,227,271,273,254,174) , W=5300

Table 2. Experimental results

Dataset	Criteria	BH	PSO	ACO	GA
F1	Best	35	35	35	35
	Worst	35	35	35	35
	Mean	35	35	35	35
	STD	0	0	0	0
	Time	1.1009	1.3906	1.4089	1.8434
F2	Best	23	23	23	23
	Worst	23	23	23	23
	Mean	23	23	23	23
	STD	0	0	0	0
	Time	1.0446	1.3868	1.3886	1.7874
F3	Best	130	130	130	130
	Worst	130	130	130	130
	Mean	130	130	130	130
	STD	0	0	0	0
	Time	1.0275	1.3503	1.5505	2.5025
F4	Best	107	107	107	107
	Worst	107	102	107	105
	Mean	107	106.3000	107	106.1333
	STD	0	1.2077	0	1.0080
	Time	1.0222	1.3516	1.8457	2.5817
F5	Best	295	295	295	295
	Worst	295	287	295	294
	Mean	295	293.9000	295	294.9333
	STD	0	2.3245	0	0.2537
	Time	0.9857	1.3143	2.2787	1.7468
F6	Best	52	52	52	52
	Worst	52	50	52	52
	Mean	52	51.8667	52	52
	STD	0	0.4342	0	0

	Time	1.0097	1.3903	2.3495	2.8956
F7	Best	481.0694	481.0694	481.0694	481.0694
	Worst	481.0694	418.1158	481.0694	475.4784
	Mean	481.0694	470.1937	481.0694	480.8830
	STD	4.33361e-014	19.4325	4.3361e-014	1.0208
	Time	1.0385	1.3553	2.9845	2.4318
F8	Best	1024	1024	1024	1024
	Worst	1018	957	1018	984
	Mean	1023.4000	1008.9667	1023.4000	1017.5667
	STD	1.8308	18.8249	1.8308	9.8740
	Time	1.0626	1.3998	3.6573	1.8483
F9	Best	1025	1025	1025	1025
	Worst	1019	947	1017	967
	Mean	1024.40000	1010.0333	1023.9333	1019.8000
	STD	1.8308	17.1233	2.4486	11.6867
	Time	1.5968	1.8957	3.6441	1.9119
F10	Best	9767	9767	9767	9765
	Worst	9761	9741	9746	9735
	Mean	9.7651e+003	9.7587e+003	9.7547e+003	9.7565e+003
	STD	1.9286	6.4869	4.9560	5.9581
	Time	1.0289	1.3691	4.0206	1.8590
F11	Best	36	36	36	36
	Worst	36	36	36	36
	Mean	36	36	36	36
	STD	0	0	0	0
	Time	1.5245	2.0522	1.7160	3.5485
F12	Best	3961	3961	3961	3961
	Worst	3961	3927	3961	3952
	Mean	3961	3957.5333	3961	3960.3333
	STD	0	7.1474	0	1.6884
	Time	1.7820	1.8426	2.5173	3.0406
F13	Best	1358	1358	1246	1357
	Worst	1328	1273	1080	1277
	Mean	1336.8667	1322.3189	1140.8333	1327.5503
	STD	5.0154	17.2174	43.2021	15.2189
	Time	1.1688	1.4057	5.0208	4.7708
F14	Best	3103	3098	3103	3071
	Worst	3103	2912	3039	2886
	Mean	3103	3058.2349	3081.4730	3011.42
	STD	0	30.2586	25.7512	34.2675
	Time	2.1754	2.4237	7.4287	6.2041
F15	Best	4882	4860	4882	4843
	Worst	4882	4538	4773	4475
	Mean	4882	4728.7534	4820.8561	4693.3284
	STD	0	45.6315	69.7428	71.2532
	Time	2.4153	2.8127	8.0716	6.8527
F16	Best	5181	5143	5183	5138
	Worst	5178	4849	5052	4700
	Mean	5179.4752	5026.3612	5123.7187	4951.3524
	STD	1.2794	36.4218	45.2791	84.2687
	Time	2.6048	3.0014	8.7495	6.9913
F17	Best	26559	26507	26547	26325
	Worst	26547	24830	25919	25010
	Mean	26549.9176	26124.7319	26361.2854	25719.4234
	STD	3.0748	315.1241	120.7138	284.6253
	Time	2.6041	2.9243	8.7410	6.9017

As seen from the given results in Table 2, the BH algorithm outperforms other test algorithms in all datasets. It provides better solutions with small standard deviation. It means that the BH algorithm is stable and reliable compared to other algorithms. Moreover, BH algorithm is the fastest algorithm among all test algorithms.

Moreover, we have used the Friedman's test to show the differences between the algorithms. The results obtained by the Friedman's test in Table 3 indicate that the BH algorithm is ranked first and there are significant differences in the results of the algorithms. From the results of the Holm's method in Table 5, it could be concluded that the control algorithm (BH) performs better than other algorithms, with a significant level of 0.05.

Table 3. Average ranking of algorithms based on the mean values

Algorithm	BH	PSO	ACO	GA
Ranking	1.5588	3.1764	2.2058	3.0588

Table 4. Results of Friedman's and Iman-Davenport's tests based on the mean values

Method	Statistical value	p-value	Hypothesis
Friedman	17.77058	0.00049	Rejected
Iman–Davenport	8.55655	0.00011	Rejected

Table 5. Results of the Holm's method based on the mean values (BH is the control algorithm)

i	Algorithm	z	p-value	α/i	Hypothesis
3	PSO	3.65316	0.00025	0.01666	Rejected
2	GA	3.38747	0.00070	0.025	Rejected
1	ACO	1.46126	0.14394	0.05	Not Rejected

3. Conclusion

In this study, the Black Hole algorithm was applied to solve knapsack problems. The experimental results using several benchmark datasets demonstrated strong convergence and stability for 0–1 knapsack problems by the Black Hole algorithm. The low run time and simplicity are the main advantages of Black Hole algorithm. Moreover, black hole continues searching without entrapment in local optimum and can find the global optimum with a high degree of confidence. In general, according to comparison between the results of the test algorithms, Black Hole obtained better solutions with less execution time.

References

[1] Kellerer, H. and V.A. Strusevich, *Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications.* Algorithmica, vol.57, no.4: pp.769-795, 2010.

[2] A. Hatamlou, E. Ghaniyarlou, Solving knapsack problems using heart algorithm, *IJAISC*, vol. 5, no. 4, pp.285-293. 2016.

[3] Tavares, J., F.B. Pereira, and E. Costa, *Multidimensional knapsack problem: A fitness landscape analysis.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 38, no. 3: pp.604-616. 2008.

[4] Truong, T.K., K. Li, and Y. Xu, *Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem.* Applied Soft Computing, vol.13, no.4, pp.1774-1780. 2013.

[5] Aho, I., *Interactive Knapsacks: Theory and Applications*. University of Tampere. 2002:

[6] Martello, S. and P. Toth, *Knapsack problems: algorithms and computer implementations*. John Wiley \\& Sons, Inc. 296. 1990:

[7] Kellerer, H., et al., *Knapsack Problems.* Springer, 2003.

[8] A Hatamlou, Solving Travelling Salesman Problem Using Heart Algorithm, *International Journal of Applied Evolutionary Computation (IJAEC)*, vol. 8, no.4, 32-42. 2017.

[9] A Hatamlou, Numerical Optimization Using the Heart Algorithm, *International Journal of Applied Evolutionary Computation (IJAEC)*, vol. 9, no.2, 33-37. 2018.

[10] M Ruhnavaz, A Hatamlou, Modeling Ghotour-Chai River's Rainfall-Runoff process by Genetic Programming, Journal of Advances in Computer Research, vol.9, no. 1, 71-84, 2018.

[11] P Mohammadi, A Hatamlou, M Masdari, A comparative study on remote tracking of Parkinsons disease progression using data mining methods, arXiv preprint arXiv:1312.2140, 2013.

[12] A. Hatamlou, A hybrid bio-inspired algorithm and its application, Applied Intelligence, vol. 47, no. 4, pp. 1059-1067, 2017.

[13] A. Hatamlou, Solving travelling salesman problem using black hole algorithm, Soft Computing, vol.22, no. 24, pp. 8167-8175, 2018.

[14] B. Javidy, A. Hatamlou, S Mirjalili, Ions motion algorithm for solving optimization problems, *Applied Soft Computing*, 32, 72-79. 2015,

[15] A. Bouyer, A. Hatamlou, An efficient hybrid clustering method based on improved cuckoo optimization and modified particle swarm optimization algorithms, *Applied Soft Computing*, vol.67, pp. 172-182, 2018.

[16] A. Hatamlou, Heart: a novel optimization algorithm for cluster analysis. Prog. Artif. Intell. vol. 2, no. 2-3, pp.167-173, 2014.

[17] Zou, Dexuan, et al. "Solving 0–1 knapsack problem by a novel global harmony search algorithm." *Applied Soft Computing* 11.2, pp.1556-1564. (2011):

[18] Lin, Feng-Tse. "Solving the knapsack problem with imprecise weight coefficients using genetic algorithms." *European Journal of Operational Research* 185.1 133-145, (2008):

[19] Ji, Junzhong, et al. "An ant colony optimization algorithm for solving the multidimensional knapsack problems." *Intelligent Agent Technology, 2007. IAT'07. IEEE/WIC/ACM International Conference on*. IEEE, 2007.

[20] Sundar, Shyam, Alok Singh, and André Rossi. "An artificial bee colony algorithm for the 0–1 multidimensional knapsack problem." *Contemporary*

Computing. Springer Berlin Heidelberg, pp. 141-151. 2010.

[21] Pulikanti, Srikanth, and Alok Singh. "An artificial bee colony algorithm for the quadratic knapsack problem." *Neural Information Processing*. Springer Berlin Heidelberg, 2009.

[22] Kong, Min, and Peng Tian. "Apply the particle swarm optimization to the multidimensional knapsack problem." *Artificial Intelligence and Soft Computing–ICAISC 2006*. Springer Berlin Heidelberg, pp.1140-1149. 2006.

[23] Gherboudj, Amira, Abdesslem Layeb, and Salim Chikhi. "Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm." *International Journal of Bio-Inspired Computation* 4.4, 229-236. (2012):

[24]. Hatamlou, Abdolreza. "Black hole: A new heuristic optimization approach for data clustering." *Information Sciences* 222, 175-184, (2013):

[25] Wang, L., et al., *An improved adaptive binary harmony search algorithm.* Information Sciences, 232: pp.58-87, 2013.

[26] K. Chen, L. Ma, Artificial glowworm swarm optimization algorithm for 0-1 knapsack problem, *Appl. Res. Comput.* vol.30, no. 4, pp. 996–998, (2013).

[27] J.Q. Liu, Y.C. He, Gu Qian Q. Solving knapsack problem based on discrete particle swarm optimization, *Comput. Eng. Design*, vol. 29, no. 13, pp.3189–3191, (2007).

[28] W.L. Xiang, M.Q. An, Y.Z. Li, et al., A novel discrete global-best harmony search algorithm for solving 0-1 knapsack problems, Discret. Dyn. Nat. Soc. (2014) 12, http://dx.doi.org/10.1155/2014/573731, Article ID 573731.

[29] Chen, H., Y. Zhu, and K. Hu, *Discrete and continuous optimization based on multi-swarm coevolution.* Natural Computing, vol. 9, no. 3. pp.659-682. 2010.