Soft Comput (2018) 22:8167–8175
https://doi.org/10.1007/s00500-017-2760-y

METHODOLOGIES AND APPLICATION

Solving travelling salesman problem using black hole algorithm

Abdolreza Hatamlou[1]

Published online: 7 August 2017

Abstract Over the last few decades, many nature-inspired algorithms have been proposed for solving complex and difficult problems. Each algorithm has its own merits and drawbacks. One of the most recent nature-inspired algorithms, which has been applied successfully in many applications, is black hole (BH) algorithm. BH algorithm is a population-based meta-heuristic algorithm that is inspired by the black hole phenomenon. It starts with a random population of solutions to the given optimization problem. The most excellent solution at each iteration which has the best fitness is chosen to be the black hole and the other form the stars. The black hole pulls the stars towards it and causes them to search the problem space for finding optimal solution. In this paper, the application of the BH algorithm on solving travelling salesman problem (TSP) is investigated. The aim of TSP is to find a tour in a set of cities in such a way, each city is visited exactly once and return to the starting city where the length of the tour is minimized. In order to evaluate the efficiency of the BH algorithm, it has been tested on several benchmark data sets and compared to other well-known algorithms. The experimental results show that the BH algorithm can find high-quality solutions compared to genetic algorithm, ant colony optimization and particle swarms optimization algorithms. Moreover, the BH algorithm is faster than other test algorithms.

Keywords Black hole algorithm · Travelling salesman problem · Meta-heuristic algorithms

Communicated by V. Loia.

✉ Abdolreza Hatamlou
hatamlou@iaukhoy.ac.ir

[1] Department of Computer Science, Khoy Branch, Islamic Azad University, Khoy, Iran

1 Introduction

Solving complex problems using meta-heuristic algorithms is an active and interesting research area. A meta-heuristic algorithm is an iterative technique that simulates a natural process of physical phenomenon, chemical or biological system with self-organization and evolving capabilities. Nature-inspired meta-heuristic algorithms are becoming powerful and famous in solving complex optimization problems (Yang 2010). A wide range of nature-inspired algorithms have been proposed in recent years. For instance, the genetic algorithms (GA) (Leardi et al. 2009), particle swarm optimization (PSO) (Kennedy and Eberhart 1995; Hatamlou 2017), ant colony optimization (ACO) (Dorigo and Blum 2005), artificial bee colony (ABC) (Karaboga and Basturk 2007), gravitational search algorithm (GSA) (Rashedi et al. 2009), Cuckoo optimization algorithm (Rajabioun 2011), black hole algorithm (BH) (Hatamlou 2013), Grey Wolf Optimizer (GWO) (Mirjalili et al. 2014), and heart algorithm (Hatamlou 2014; Hatamlou and Ghaniyarlou 2016) are popular examples of nature-inspired algorithms.

These algorithms usually start with an initial population of candidate solutions for the given problem and then evolve to find the optimal solution. Intensification and diversification are two main features of the meta-heuristic algorithms. The diversification ensures that the algorithm explores the search space more efficiently by searching new and different areas. On the other hand, at the intensification phase the algorithm searches around the current best solutions and focuses on promising areas. Nature-inspired meta-heuristic algorithms have now been used in many applications and different fields such as computer science, data mining, industry, agriculture, computer vision, forecasting, medicine and biology, scheduling, and economy (Hatamlou et al. 2011a, b, c,

2012; Hatamlou and Hatamlou 2013a, b; Semwal et al. 2017, 2015a, b).

One of the most recent and popular nature-inspired algorithms is black hole algorithm (BH) (Hatamlou 2013). BH algorithm simulates the black hole phenomenon in the space to solve the optimization problems by searching the problem space in an efficient way. BH is very simple and promising, and it requires less number of parameters and computation time. Recently, it has demonstrated to be an efficient and effective approach for different optimization problems (Bouchekara 2014a, b; Heidari and Abbaspour 2014; Lenin et al. 2014).

This paper investigates the application of the BH algorithm on solving travelling salesman problem (TSP). Travelling salesman problem is a well-known and benchmark problem for studying and evaluating the performance of optimization algorithms. The aim of the travelling salesman problem is finding a tour of a finite number of cities, visiting each city exactly once and returning to the starting city where the length of the tour is minimized (Hoffman et al. 2013).

The rest of the paper is organized as follows: In Sect. 2, the travelling salesman problem is discussed. A brief explanation of the black holes algorithm is given in Sect. 3. In Sect. 4, solving the travelling salesman problem using black hole algorithm is described, and finally, Sect. 5 contains a summary and conclusion of this work.

2 Travelling salesman problem

Travelling salesman problem (TSP) is one of the famous, well-known and widely studied problems in the field computer science and optimization problems, and many researchers have been attracted to solve it efficiently by applying novel and different algorithms (Adham and Bentley 2014; Ahmed 2014; Baltz et al. 2014; Battarra et al. 2014a; Changdar et al. 2014; Escario et al. 2015; Fei et al. 2014; Hoos and Stützle 2014; Hougardy and Wilde 2014; Jones and Adamatzky 2014; Karabulut and Tasgetiren 2014; Mohemmed et al. 2008; Ouaarab et al. 2014a, b; Sahana and Jain 2014; Uchida et al. 2014; Vallade and Nakashima 2014; Verma et al. 2014; Wang et al. 2015; Wang 2014; Zong and Wang 2014; Elloumi et al. 2014). It is a simple and straight problem; however, it remains one of the most challenging problems in operational research. In the travelling salesman problem (TSP), the aim is to find a tour in a set of cities in such a way, each city is visited exactly once and return to the starting city where the length of the tour is minimized. The travelling salesman problem has been used in different applications and areas including hardware designing, communications, and computer networks. Moreover, many industrial problems such as cellular manufacturing, machine scheduling, and frequency assignment problems can be formulated as a TSP.

The travelling salesman problem is a NP-hard problem, meaning that there is no exact algorithm to find its solutions in polynomial time. The minimal expected time to find its optimal solutions is exponential. So, usually meta-heuristic algorithms are applied by the researchers to find a "good" solution in this problem. Till now, different algorithms were used to solve TSP with more or less success. There are different ways to categorize these algorithms, each with its own advantages. The main characteristic is the ability of finding optimal solution: exact algorithms or meta-heuristic approaches (Baltz et al. 2014; Battarra et al. 2014a). The well-known and popular exact algorithms for solving TSP are: implicit enumeration, explicit enumeration, branch and bound algorithm, cutting plane algorithm, and dynamic programming. These algorithms work well for problems, which do not have more than 40–80 cities. Currently, the only algorithm guaranteed to find optimal solution of the travelling salesman problem of any size is considering each possible tour and searching for the tour with smallest cost (explicit enumeration). In this algorithm, each possible solution is a permutation of $1, 2, 3, \ldots, n$, where n is the number of cities. So the number of solutions becomes n! When n gets large, it will be impossible to find the cost of all these solutions in a polynomial time. Clearly, this algorithm, which will end up giving the optimal solution, is not suitable for large problems because of the time consumption required to calculate all the possible solutions. Even for the small problems, it can be seen that the required time is extremely high if we want to find every possible solution. For this reason, it is necessary to solve the larger TSP problems with the help of meta-heuristic approaches. Meta-heuristic algorithms are different from exact approaches in that they give no guarantee to find the optimal solutions of the optimization problems. Instead, they can find suboptimal solutions. But in most cases the quality of the obtained suboptimal solutions is good and acceptable, and we can find them in reasonable time.

Travelling salesman problem can be represented using a weighted graph $G = (N, E)$, where N is the set of n cities and E is the set of edges (paths) fully connecting all cities. Each edge $(i, j) \in E$ is assigned a cost d_{ij}, which is the distance between cities i and j. Usually, the Euclidean distance is applied to calculate the d_{ij} using the following equation:

$$d_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{1}$$

One simple and direct method for solving TSP is to select the route which has minimum total cost for all possible permutations of N cities. The number of permutations can be very large for problems, which have more than even 40 cities. For symmetrical TSP, each solution is represented in $2n$ different

ways. Since there are $n!$ possible ways to permute n numbers, the size of the search space is equal to the following equation:

$$|S| = n!/(2n) = (n-1)!/2. \quad (2)$$

In this paper, we apply a novel meta-heuristic algorithm called black hole algorithm (BH) to solve the travelling salesman problem (TSP), which is explained in detail in the next section.

3 Black hole algorithm

The black hole (BH) algorithm is one of the most recent nature-inspired algorithms (Hatamlou 2013). It is based on the black hole phenomenon in the space. A black hole in space is what forms when a star of massive size collapses. The gravitational power of the black hole is too high that even the light cannot escape from it. The gravity is so strong because matter has been squeezed into a tiny space. Anything that crosses the boundary of the black hole will be swallowed by it and vanish and nothing can get away from its enormous power.

In the BH algorithm, like as other population-based algorithms a population of candidate solutions to the optimization problem is generated randomly. These candidate solutions are distributed in the problem space. After initialization, the fitness values of the candidate solutions are evaluated, and the best candidate among the population is chosen to be the black hole and the rest form the normal stars.

Then, the BH algorithm evolves the candidate solutions towards the optimal solution via a simple mechanism. The black hole (best candidate) starts absorbing the stars (other candidates) around it, and all the stars start moving towards the black hole. The absorption of the candidates towards the black hole is formulated as follows:

$$x_i(t+1) = x_i(t) + \text{rand} * (x_{\text{BH}} - x_i(t)) \quad i = 1, 2, \ldots N \quad (3)$$

where $x_i(t)$ and $x_i(t+1)$ are the locations of the ith star at iterations t and $t+1$, 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 total number of stars (candidate solutions).

While the stars move towards the black hole, it is possible that a star reaches to a better location in the search space which has better fitness than the black hole position. In such a case, the black hole is replaced by that star, and then, the BH algorithm continues with the new black hole and the stars start moving towards this new black hole.

During moving stars towards the black hole, there is the possibility of crossing the border of black hole (event horizon). Every star (candidate solution) that enters the range of the black hole will be sucked by the black hole. In such a case, every time a candidate (star) dies, another candidate solution (star) is born and distributed randomly in the search space and starts a new search. After all the stars have been moved and reached to new locations using Eq. (3), the next iteration will be take place.

The radius of the event horizon in the black hole algorithm is calculated using the following equation:

$$R = \frac{f_{\text{BH}}}{\sum_{i=1}^{N} f_i} \quad (4)$$

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). The flow chart of the black hole algorithm is shown in Fig. 1.

Based on the above description, the main steps in the BH algorithm are 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 Eq. (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

4 Experimental results

To evaluate the performance of the black hole (BH) algorithm for solving the travelling salesman problem (TSP), we have compared its results with three other approaches including genetic algorithm (GA), ant colony optimization (ACO), and particle swarms optimization (PSO) algorithms. Ten popular benchmark data sets from the literature have been used to assess the performance of these algorithms (Verma et al. 2014).

All algorithms have been implemented in MATLAB and run on an Intel(R) Core(TM) 2 Duo CPU 2 GHz PC with 2 GB memory. There are many parameters used for the GA and ACO and PSO algorithms. The size of population will

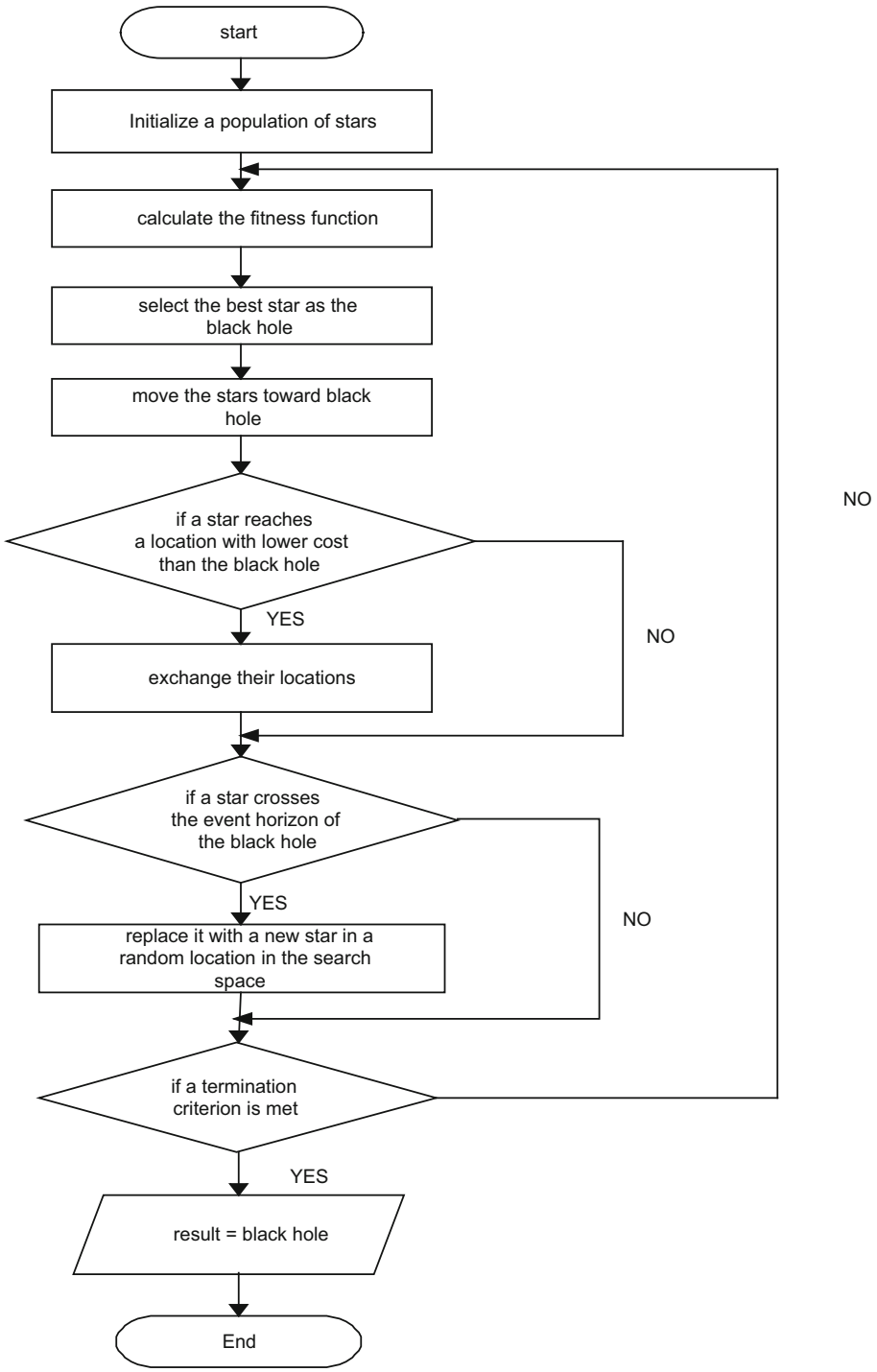


Fig. 1 Flow chart of black hole algorithm

be increased three times. In PSO algorithm social and cognitive probabilities, having $c1$ and $c2$, set as $c1 = c2 = 2$. While the inertia weight is taken as 0.9, the maximum of velocity v is taken as 100 and dimension of space as 10. In ACO algorithm, both α and β control the relative significance of pheromone trail and distance between cities in TSP, where $\alpha = 1.5$, $\beta = 2$. The rate of pheromone evaporation set as $\rho = 0.7$. In GA, crossover percentage and mutation percentage, having P_c and P_m, set as $P_c = 0.8$ and $P_m = 0.3$, and mutation rate is taken as Mu $= 0.02$. Each TSP run is conducted for 5 replications and 200 iterations.

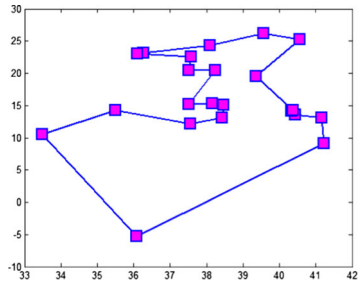
Table 1 Experimental results of algorithms on the test data sets by different populations

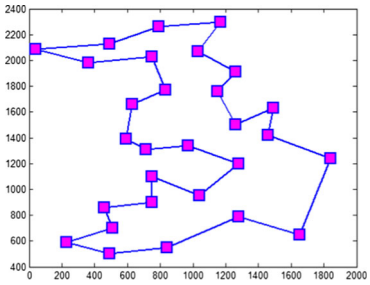
Problem	Population	L. ACO	L. PSO	L. GA	L. BH
ulysses22 (22 cities)	22	75.8944	75.3097	78.7865	75.1343
	70	75.3984	75.3097	76.7624	76.1028
	100	75.6536	76.5014	75.7744	75.3098
bays29 (29 cities)	29	9308.3133	9190.7908	9838.9833	9120.3388
	70	9252.5571	9076.9829	9695.15	9105.876
	100	9217.3296	10061.1974	9751.4255	9105.876
bayg29 (29 cities)	29	9496.1451	10215.1708	9779.1234	9329.251
	70	9395.3096	9723.6328	9726.0725	9329.251
	100	9395.3096	9458.8467	9579.1234	9360.9789
att48 (48 cities)	48	36140.8669	44728.553	35307.6782	34506.675
	70	35394.0453	35713.4313	35307.6782	34780.9828
	100	35750.6922	40684.3436	35312.5165	34817.3494
eil51 (51 cities)	51	458.1402	486.0377	549.62	438.9547
	70	454.8300	510.3702	554.2824	440.6119
	100	452.8102	534.3113	448.8397	442.3588
berlin52 (52 cities)	52	7859.3944	9109.0275	8225.7848	8088.9575
	70	7842.2707	10015.6048	8541.2008	8273.9146
	100	7708.7103	9239.3427	8779.7559	7804.6642
st70 (70 cities)	70	746.0173	1190.6963	1227.5028	736.0078
	100	735.2768	1185.5164	1112.3078	721.5652
	150	726.2620	1110.8186	1120.6461	754.914
eil76 (76 cities)	76	654.2776	945.6476	839.5299	600.8717
	100	650.1808	875.4937	619.2262	579.6002
	150	573.898	857.1774	557.3278	580.4846
gr96 (96 cities)	96	666.5597	1243.0271	742.9912	651.8578
	100	631.4399	1260.3899	737.9671	591.9497
	150	626.8571	1180.3758	694.5256	594.548
eil101 (101 cities)	101	989.8531	1383.9296	892.9387	717.9479
	100	991.1647	1408.4436	828.8806	718.2272
	150	978.2320	1300.2764	872.7488	721.1545

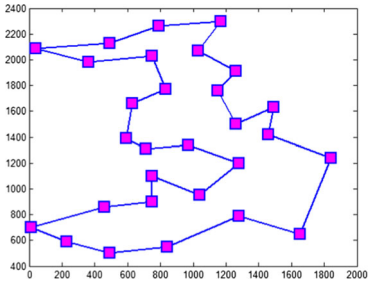
Table 1 summarizes the obtained results and comparison between BH, ACO and PSO and GA. In this table, L. ACO: the best length for ACO, L. PSO: the best length for PSO, L. GA: the best length for GA, L. BH: the best length for BH. From the given results in Table 1, everyone can see that the quality of the obtained solutions by the BH algorithm is better compared to PSO and ACO and GA in all data sets. This is true for all population sizes. Moreover, as the size of population increases, execution time of the BH algorithm decreases when compared to the time taken by PSO and ACO and GA. It means that the BH algorithm in finding optimal solutions is very fast compared to the PSO and ACO and GA.

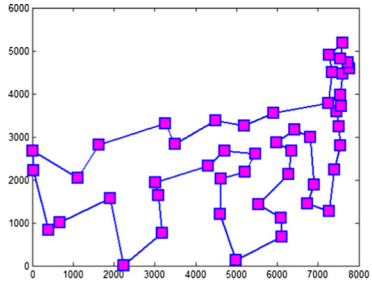
Table 2 shows the quality of the obtained solutions by four algorithms in terms of the best, average, and worst values after five independent runs for each of the ten data sets. Clearly, the small values show the high-quality solutions and vice versa. Moreover, for each algorithm the standard deviation of solutions (Std) on different runs is provided to show the reliability and stability of algorithms. A low standard deviation indicates that the respective algorithm is more reliable and stable to find optimal solution. The given results are conducted for 200 iterations, and the size of population is set to be 100.

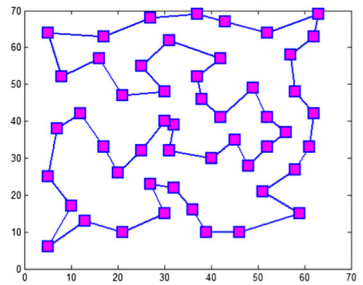
In Table 2, we can see that the black hole algorithm has achieved the best performance in terms of the best, average, and worst solutions on all the test data sets. The quality of the obtained solutions by the BH algorithms in all cases is better than the respective values of the GA and PSO and ACO algorithms. Besides, the standard deviation of obtained solutions by the BH algorithms is smaller than the other test algorithms, meaning that the BH algorithm is more reliable and stable in finding optimal solutions, while other algorithms may trap in local optima solutions.

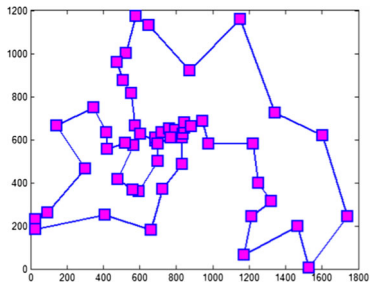


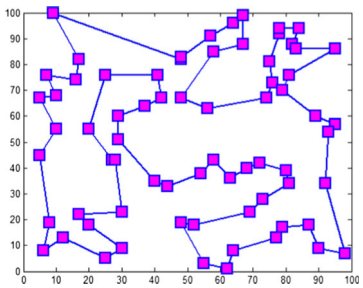


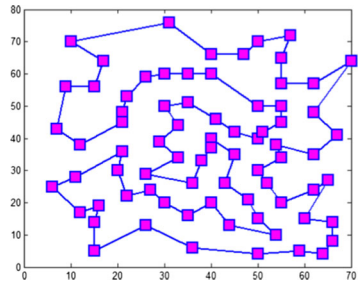


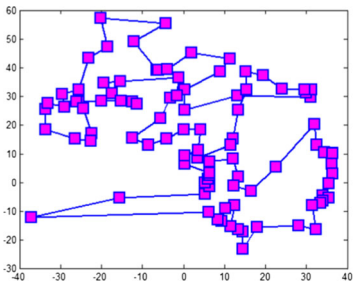












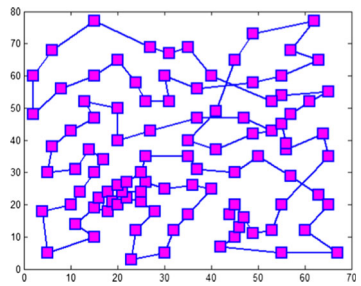


Fig. 2 TSP solutions found by the BH algorithm on the test data sets

Table 2 Comparison of the algorithms on the test data sets by population size of 100

Problem	Algorithm	Best	Worst	Average	Std	Time
ulysses22 (22 cities)	ACO	75.3984	75.8409	75.4869	0.19789	84.271226
	PSO	75.9104	77.1857	76.2186	0.55273	61.879917
	GA	75.7744	76.4434	75.9878	1.2307	63.392161
	BH	75.3097	75.9343	75.6844	0.34208	50.447445
bays29 (29 cities)	ACO	9239.1973	11014.4483	9823.2018	722.4152	88.256599
	PSO	9120.3388	9498.1711	9195.9053	168.9717	88.828688
	GA	9751.4255	10513.9142	10015.2309	319.8788	57.118643
	BH	9396.475	9507.1701	9463.2521	60.9588	52.104801
bayg29 (29 cities)	ACO	9447.4929	11033.5484	9882.2197	675.8331	99.957237
	PSO	9329.251	11332.7224	9947.0255	799.4073	75.296610
	GA	9579.1234	10411.1991	9771.9537	127.1131	56.161174
	BH	9375.4418	9375.4418	9375.4418	0	45.870954
att48 (48 cities)	ACO	35230.904	46204.242	39436.1813	4874.2954	133.457095
	PSO	36996.4434	61421.9919	47018.4132	9685.8943	84.738416
	GA	35312.5165	50671.4538	43620.6376	2004.0005	57.354531
	BH	34200.8643	35528.5196	34473.8434	589.8024	43.211738
eil51 (51 cities)	ACO	454.3895	469.0531	461.0175	6.2974	59.193278
	PSO	469.1551	737.5258	574.8022	107.2371	57.256463
	GA	448.8397	462.1142	453.4773	9.4157	59.639161
	BH	437.893	526.8977	458.9252	38.6365	44.390089
berlin52 (52 cities)	ACO	7757.0263	10541.1228	8522.9017	1152.2	65.070133
	PSO	9218.4682	14279.4331	11089.5286	2067.9323	68.648061
	GA	8779.7559	9565.3744	9288.4483	1301.2108	52.735340
	BH	8188.0714	9356.7483	8455.8304	508.9871	43.404461
st70 (70 cities)	ACO	711.6515	855.2032	757.754	59.6079	94.568215
	PSO	1030.8484	1756.1227	1321.8137	269.2793	55.284116
	GA	1112.3078	1242.2011	1158.8458	52.1734	55.095853
	BH	723.2691	1081.1087	797.5745	125.2272	45.330802
eil76 (76 cities)	ACO	574.2404	665.9995	594.1442	40.2152	61.741797
	PSO	804.2667	1195.9021	975.6397	152.4061	56.767078
	GA	619.2262	679.7864	652.0593	122.0972	46.691510
	BH	566.243	925.8417	659.1021	152.1754	46.540376
gr96 (96 cities)	ACO	555.7535	639.9167	580.5406	33.9301	84.389766
	PSO	1095.1109	1728.8241	1378.8696	247.5099	56.211707
	GA	737.9671	748.3543	742.4275	4.3282	63.244442
	BH	546.8397	1197.8761	807.2465	258.815	43.587907
eil101 (101 cities)	ACO	725.0996	868.2047	763.9207	59.9684	89.639741
	PSO	1158.704	1973.8192	1499.9911	319.7468	62.093024
	GA	828.8806	854.4381	838.8307	9.9642	55.188206
	BH	720.3838	1249.8684	897.3813	210.1446	45.833365

In general, the simulation results show that the BH algorithm is a robust and precise approach for solving travelling salesman problem. This algorithm provides the optimum value and small standard deviation as opposed to other algorithms.

Figure 2 shows a set of obtained solutions by the black hole algorithm on the used data sets.

5 Conclusion

Modelling and simulating natural phenomena for solving complex problems have been an interesting research area for several decades. In this paper, the application of black hole algorithm (BH) for solving the travelling salesman problem (TSP) is investigated. In order to evaluate the efficiency of

the BH algorithm on solving TSP, it has been tested on several benchmark data sets and compared with two popular algorithms in the literature. The experimental results shows that the BH algorithm can find high-quality solutions compared to genetic algorithm (GA) and ant colony optimization (ACO) and particle swarms optimization (PSO) algorithms. Moreover, the BH algorithm is faster than GA and ACO and PSO algorithms in all test data sets.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Adham MT, Bentley PJ (2014) An artificial ecosystem algorithm applied to the travelling salesman problem. In: Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion. ACM

Ahmed ZH (2014) Improved genetic algorithms for the travelling salesman problem. Int J Process Manag Benchmark 4(1):109–124

Baltz A et al (2014) Exact and heuristic algorithms for the Travelling salesman problem with multiple time windows and hotel selection. J Oper Res Soc 66(4):615–626

Battarra M et al (2014a) Exact algorithms for the traveling salesman problem with draft limits. Eur J Oper Res 235(1):115–128

Bouchekara H (2014a) Optimal power flow using black-hole-based optimization approach. Appl Soft Comput 24:879–888

Bouchekara H (2014b) Optimal design of electromagnetic devices using a black-hole-based optimization technique. IEEE Trans Magn 49(12):5709–5714

Changdar C, Mahapatra GS, Pal R Kumar (2014) An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness. Swarm Evol Comput 15:27–37

Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344(2–3):243–278

Elloumi W et al (2014) A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP. Appl Soft Comput 25:234–241

Escario JB, Jimenez JF, Giron-Sierra JM (2015) Ant colony extended: experiments on the travelling salesman problem. Expert Syst Appl 42(1):390–410

Fei T et al (2014) The artificial fish swarm algorithm to solve traveling salesman problem. In: Proceedings of international conference on computer science and information technology. Springer, New Delhi. doi:10.1007/978-81-322-1759-6_78

Hatamlou A (2017) A hybrid bio-inspired algorithm and its application. Appl Intell. doi:10.1007/s10489-017-0951-y

Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci 222:175–184

Hatamlou A (2014) Heart: a novel optimization algorithm for cluster analysis. Progr Artif Intell 2(2–3):167–173

Hatamlou A, Ghaniyarlou E (2016) Solving knapsack problems using heart algorithm. Int J Artif Intell Soft Comput 5(4):285–293

Hatamlou A, Hatamlou M (2013) PSOHS: an efficient two-stage approach for data clustering. Memet Comput 5(2):155–161

Hatamlou A, Hatamlou M (2013) Hybridization of the gravitational search algorithm and big bang-big crunch algorithm for data clustering. Fund Inf 126(4):319–333

Hatamlou A, Abdullah S, Nezamabadi-pour H (2011) Application of gravitational search algorithm on data clustering, rough sets and knowledge technology. Springer, Berlin

Hatamlou A, Abdullah S, Nezamabadi-pour H (2012) A combined approach for clustering based on K-means and gravitational search algorithms. Swarm Evolut Comput 6:47–52

Hatamlou A, Abdullah S, Hatamlou M (2011) Data clustering using big bang-big crunch algorithm. In: Communications in computer and information science, pp 383–388

Hatamlou A, Abdullah S, Othman Z (2011) Gravitational search algorithm with heuristic search for clustering problems. In: 3rd conference on data mining and optimization (DMO)

Heidari AA, Abbaspour RA (2014) Improved black hole algorithm for efficient low observable UCAV path planning in constrained aerospace. Adv Comput Sci Int J 3(3):87–92

Hoffman KL, Padberg M, Rinaldi G (2013) Traveling salesman problem. In: Encyclopedia of operations research and management science. Springer, Berlin. pp 1573–1578

Hoos HH, Stützle T (2014) On the empirical scaling of run-time for finding optimal solutions to the travelling salesman problem. Eur J Oper Res 238(1):87–94

Hougardy S, Wilde M (2014) On the nearest neighbor rule for the metric traveling salesman problem. Discrete Appl Math 195(2015):101–103

Jones J, Adamatzky A (2014) Computation of the travelling salesman problem by a shrinking blob. Nat Comput 13(1):1–16

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471

Karabulut K, Tasgetiren M Fatih (2014) A variable iterated greedy algorithm for the traveling salesman problem with time windows. Inf Sci 279:383–395

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: IEEE international conference on neural networks: proceedings

Leardi R et al (2009) Genetic algorithms. In: Comprehensive chemometrics. Elsevier, Oxford, pp 631–653

Lenin K, Reddy BR, Kalavathi MS (2014) Black hole algorithm for solving optimal reactive power dispatch problem. Int J Res Manag Sci Technol 2:2321–3264

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61

Mohemmed AW, Sahoo NC, Geok TK (2008) Solving shortest path problem using particle swarm optimization. Appl Soft Comput 8(4):1643–1653

Ouaarab A, Ahiod Bd, Yang X-S (2014) Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput Appl 24(7–8):1659–1669

Ouaarab A, Ahiod Bd, Yang XS (2014) Improved and discrete cuckoo search for solving the travelling salesman problem. In: Cuckoo search and firefly algorithm. Springer, Berlin, pp 63–84

Rajabioun R (2011) Cuckoo optimization algorithm. Appl Soft Comput 11(8):5508–5518

Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248

Sahana SK, Jain A (2014) High performance ant colony optimizer (HPACO) for travelling salesman problem (TSP). In: Tan Y, Shi Y, Coello CAC (eds) Advances in swarm intelligence. Springer, Berlin, pp 165–172

Semwal VB et al (2015) Biologically-inspired push recovery capable bipedal locomotion modeling through hybrid automata. Robot Auton Syst 70:181–190

Semwal VB, Chakraborty P, Nandi GC (2015) Less computationally intensive fuzzy logic (type-1)-based controller for humanoid push recovery. Robot Auton Syst 63:122–135

Semwal VB, Mondal K, Nandi GC (2017) Robust and accurate feature selection for humanoid push recovery and classification: deep learning approach. Neural Comput Appl 28(3):565–574

Uchida A, Ito Y, Nakano K (2014) Accelerating ant colony optimisation for the travelling salesman problem on the GPU. Int J Parallel Emerg Distrib Syst 29(4):401–420

Vallade B, Nakashima T (2014) Improving particle swarm optimization algorithm and its application to physical travelling salesman problems with a dynamic search space. In: Lee R (ed) Applied computing and information technology. Springer, Berlin, pp 105–119

Verma OP, Jain R, Chhabra V (2014) Solution of travelling salesman problem using bacterial foraging optimisation algorithm. Int J Swarm Intell 1(2):179–192

Wang Y (2014) The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem. Comput Ind Eng 70:124–133

Wang P, Sanin C, Szczerbicki E (2015) Evolutionary algorithm and decisional DNA for multiple travelling salesman problem. Neurocomputing 150:50–57

Yang XS (2010) Nature-inspired metaheuristic algorithms. Luniver Press, Bristol

Zong D, Wang K (2014) Hybrid nested partitions method for the traveling salesman problem. In: Wen Z, Li T (eds) Foundations of intelligent systems. Springer, Berlin, pp 55–67