An International Journal of Optimization
and Control: Theories & Applications
Vol.6, No.2, pp.103-113 (2016) © IJOCTA
ISSN: 2146-0957 eISSN: 2146-5703
DOI: 10.11121/ijocta.01.2016.00268
http://www.ijocta.com

Using 2-Opt based evolution strategy for travelling salesman problem

Kenan Karagul [a] Erdal Aydemir [b] and Sezai Tokat [c]

[a] Department of Logistics, Honaz MYO, Pamukkale University, Denizli, Turkey
Email: kkaragul@pau.edu.tr

[b] Department of Industrial Engineering, Engineering Faculty, Suleyman Demirel University, Isparta, Turkey
Email: erdalaydemir@sdu.edu.tr

[c] Department of Computer Engineering, Engineering Faculty, Pamukkale University, Denizli, Turkey
Email: stokat@pau.edu.tr

(*Received August 12, 2015; in final form March 24, 2016*)

Abstract. Harmony search algorithm that matches the (μ+1) evolution strategy, is a heuristic method simulated by the process of music improvisation. In this paper, a harmony search algorithm is directly used for the travelling salesman problem. Instead of conventional selection operators such as roulette wheel, the transformation of real number values of harmony search algorithm to order index of vertex representation and improvement of solutions are obtained by using the 2-Opt local search algorithm. Then, the obtained algorithm is tested on two different parameter groups of TSPLIB. The proposed method is compared with classical 2-Opt which randomly started at each step and best known solutions of test instances from TSPLIB. It is seen that the proposed algorithm offers valuable solutions.

Keywords: Travelling salesman problems; TSP; harmony search; HS; (μ+1) evolution strategy; 2-Opt; TSPLIB.
AMS Classification: 68T20; 90-08

1. Introduction

The travelling salesman problem (TSP) is one of the most popular combinatorial optimization problems in complexity theory [1]. TSP for minimizing the tour length is quite difficult to solve and classified as NP-Hard, it will be time consuming to solve larger instances. However, TSP is used in many theoretical and practical applications such as manufacturing planning, logistics, and electronics manufacturing. Due to the nature of TSP, obtaining the optimal solution is not possible in polynomial time if solved via integer programming. Also, it is known that the solution time extends exponentially as the problem size grows. Therefore, as an alternative solution approach, the meta-heuristics are commonly used to determine near optimal solutions in acceptable solution times [2-8].

In the related literature, many known meta-heuristics were used to solve TSPs for minimizing the tour lengths. For instance, Freisleben and Merz [9] presented an algorithm by using genetic algorithm (GA) to find near-optimal solution for a set of symmetric and asymmetric TSP instances and obtained high quality solutions in a reasonable time. Chowdhury et al. [10] also used GA for solving a flow-shop scheduling problem to minimize makespan via finding optimal order of cities. The simulated annealing (SA) algorithm is also used for TSP by Wang and Tian [11] in which an improved SA is employed. Meta-heuristics approach is generally used to solve the problem in reasonable time if the problem size increases. For large TSPs, Fiechter [12] used a parallel tabu search algorithm. Similarly, different types of ant colony algorithm are used for the TSP [13-15]. Also, Wang et al. [16] developed swap operator and swap sequence in order to use particle swarm optimization (PSO) for TSP.

Recently, with the progresses in computational sciences, the new meta-heuristics methods have

Corresponding Author. Email: erdalaydemir@sdu.edu.tr

been developed and used for solving combinatorial problems. Some of them are cuckoo search algorithm [17-21], firefly algorithm [22-24] and harmony search (HS) algorithm [25-28]. In Table 1, the main literature is chronologically summarized.

In our study, the harmony search algorithm was used as the core algorithm to solve TSPs. HS is first proposed by Geem et al. [5]. Weyland [29] proved that HS is theoretically a special case of an evolution strategy known as (μ +1) evolution strategy [30]. In Geem et al [5], the 20-cities TSP, constraint optimization problem, and water network pipeline design are solved. For 20-cities TSP, neighbouring city-going and city-inverting operators were designed. Their operators were used to find the closest city that will be visited next and to produce a new path on feasible nodes, respectively. Geem et al [5] did not give the details of the discrete structure. However, later Geem [31] detailed the HS for TSP that uses stochastic derivative for discrete variables

Table 1. Meta-heuristic studies on TSP

Meta-heuristics	***Literature directly on TSP***
Genetic Algorithm (GA)	Freisleben & Merz (1996), Chowdhury et al. (2013)
Simulated Annealing (SA)	Wang & Tian (2013),
Tabu Search (TS)	Fiechter (1994),
Ant Colony Optimization (ACO)	Stüzle & Hoss (1997), Randall & Montgomery (2003), Chu et al. (2004)
Particle Swarm Optimization (PSO)	Wang et al. (2003),
Cuckoo Search (CS)	Yang & Deb (2009), Ouyang et al. (2013), Ouaarab et al. (2013), Ouaarab et al. (2014)
Firefly Algorithm (FA)	Yang (2010), Jati & Suyanto (2011), Kumbharana & Pandey (2013a)
Harmony Search (HS)	Geem et al. (2001), Wang et al. (2010), Pan et al. (2011), Huang & Peng (2013), Yuan et al. (2013), Weyland (2015)
Hybrid Studies	Pang et al. (2004) (PSO&Fuzzy); Thamilselvan & Balasubramanie (2009) (GA&TS); Kaveh & Talatahari (2009) (PSO, ACO&HS); Yan et al. (2011); Chen & Chien (2011) (GA&ACO); Chen & Chien (2011) (GA,SA,PSO&ACO); Kumbharana & Pandey (2013b) (GA,SA&ACO); Yun et al. (2013) (HS&ACO)

In addition, we directly use the index values of the normally distributed harmony numbers in our method. In this process, besides the use of meta-heuristics algorithms, the hybrid approaches involving the hybridization of two or more heuristics were applied in order to eliminate the weakness of single meta-heuristics for solving the large scale TSP optimization problems [32-39]. On the other hand, HS is directly adapted for TSP.

In TSP, the main goal is to find the shortest closed tour that visits each city once and exactly once in a given list with the best route. There are tour construction methods such as the nearest neighbor, greedy, insertion heuristics, Christofides method. After the tour has been generated by any tour construction heuristics, it is improved with tour improvement heuristics such as 2-Opt, 3-Opt, k-Opt, Lin-Kernighan, Tabu-Search, Simulated Annealing, Genetic Algorithms etc. The 2-Opt approach is a well-known method used for this purpose. The 2-Opt is a simple local search algorithm and it was first proposed by Croes [2] for TSP. It swaps edges in a tour for shortening the total tour length.

The HS algorithm, a special case of evolution strategy which is called (μ+1) evolution strategy, is a meta-heuristic optimization method that inspired by the mimics of the improvisation ability of musicians. Using HS algorithm, the musical instruments are played with discrete notes under the musicians' experience and their improvisation ability randomly. The musical harmony, aesthetic standard, pitches of instruments and the improvisation process are design parameters of HS algorithm. HS works with the harmony size (HMS), the harmony considering rate and the pitch adjusting rate as optimization operators [5, 31].

A brief overview of TSP from the literature especially on meta-heuristics is surveyed in this section. The rest of the paper is organized as follows: the proposed method in which the HS algorithm with its continuous structure is directly used for the TSP is presented in Section 2. With the proposed algorithm, the transformation mechanism for HS to solve the TSP is obtained by using 2-Opt local search algorithm. Then, in Section 3, the obtained algorithm is tested on two different parameter groups of TSPLIB.

2. Proposed algorithm: 2-Opt based harmony search algorithm

The proposed algorithm that is called 2-Opt Based Harmony Search Algorithm (2-Opt_cHS) combines the algorithms of 2-Opt and the HS. The first advantage of the proposed algorithm is to convert the real numbers into index values for solving combinatorial optimization problem such as TSP. Thus, the modified algorithm provides to solve discrete optimization problems.

In general, for TSP problems, roulette wheel selection is used in evolution strategies for the transformation between randomly generated real numbers of heuristic solutions and ordered numbers of combinatorial problem solutions. In this paper, HS algorithm with its continuous structure is directly used for the travelling salesman problem. The transformation of real numbers of continuous HS algorithm to integer numbers of discrete form is obtained by using 2-Opt local search algorithm which is used to define a function from continuous to discrete functions and vice versa. The pseudo code of the HS algorithm is given as Algorithm 1 in Table 2 [29] and the proposed algorithm is given as Algorithm 2 in Table 3.

Table 2. The pseudo code of the harmony search algorithm [29]

Algorithm 1: The Harmony Search Algorithm	
1:	Initialize the harmony memory with HMS randomly generated solutions
2:	**repeat**
3:	create a new solution in the following way
4:	**for** all decision variables **do**
5:	with probability HMCR use a value of one of the solutions in the harmony memory (selected uniform random numbers) and additionally change this value slightly with probability PAR
6:	otherwise (with probability 1-HMCR) use a random value for this decision variable
7:	**end for**
8:	**if** the new solution is better than the worst solution in the harmony memory **then**
9:	replace the worst solution by the new one
10:	**end if**
11:	**until** the maximum number of iterations has been reached
12:	**return** the best solution in the harmony memory

According to Algorithm 2, firstly, the objective function is generated with real number arrays for initial harmonics. And then, its limits and bandwidths, and the values are defined as parameters. The 2-Opt algorithm is used for designing discrete variables and is defined with the step size (v) and application parameter (opt). By using v, the 2-Opt application is used once in each v steps.

The use of v parameter is the design idea of this study as using 2-Opt at each step increases the simulation time and decreases the effect of using HS algorithm. When a solution is obtained after HS with 2-Opt procedure, it will be evaluated using the fitness function. Respecting to HMS and fitness of new solution, the new solution may be inserted in harmony memory or not. Eventually, the termination criteria can be defined as a problem dependent number of iterations or as reaching to a specific quality of solution. In this study, the algorithm will stop when maximum number of iterations is met, and otherwise the while loop case will be repeated for each iteration.

A sample TSP solution taken from TSPLIB, known as Burma14, is demonstrated in Table 2. As can be seen in Table 2, IH and NH are real Harmony numbers. By using their index values in HOI, the route information is obtained and then improved by using 2-Opt. The proposed algorithm with respect to the pseudo-code in

Algorithm 2 by taking maximum number of iteration as 5 is given. The 2-Opt is used instead of the initial solution and the 3rd iteration. At iteration 4, the optimal solution for Burma14 is obtained as 3323. In Figure 1, the route improvement at some steps are visualized for Burma14.

Table 3. The pseudo code of the proposed algorithm: 2-Opt based harmony search

Algorithm 2: The Proposed Algorithm: 2-Opt Based Harmony Search

1: Initialize the harmony memory with HMS randomly generated solutions
   Get Harmony Ordered Indexes
   Get 2-Opt Ordered Indexes
2: **repeat**
3: create a new solution in the following way
4: **for** all decision variables **do**
5: with probability HMCR use a value of one of the solutions
   in the harmony memory (selected **normal** random numbers)
   and additionally change this value slightly with probability PAR
6: otherwise (with probability 1-HMCR) use a random value for this decision variable
   if (at each step size *v*) for the 2-Opt is provided
   Get Harmony Ordered Indexes
   Get 2-Opt Ordered Indexes
   else
   Get Harmony Ordered Indexes
   end if
7: **end for**
8: **if** the new solution is better than the worst solution in the harmony memory **then**
9: replace the worst solution by the new one
10: **end if**
11: **until** the maximum number of iterations has been reached
12: **return** the best solution in the harmony memory

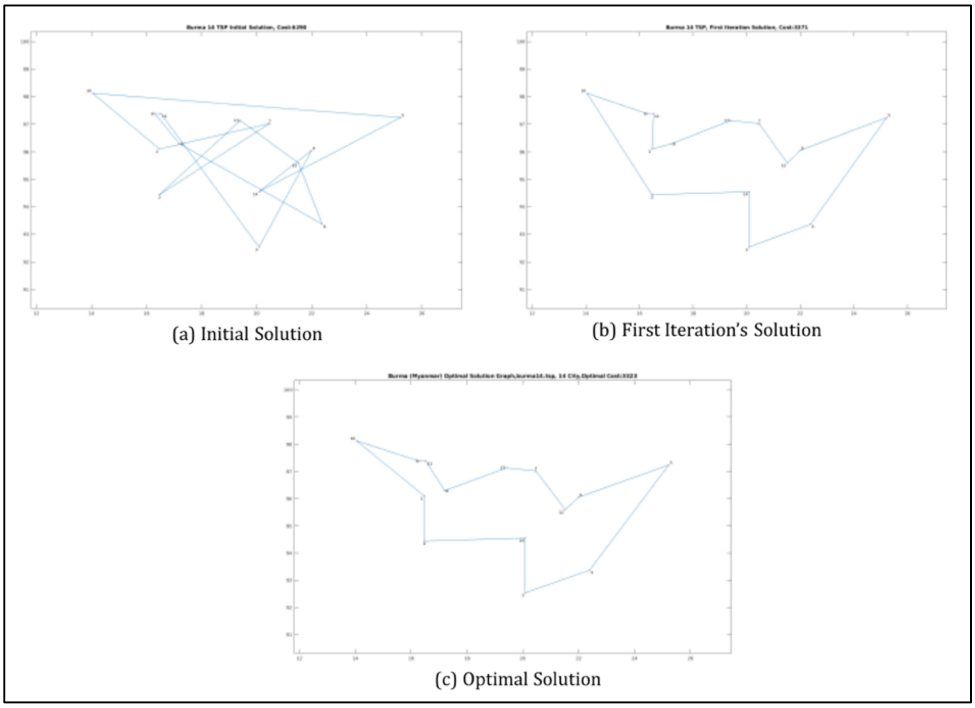

Figure 1. Burma14 solution graphs in the proposed algorithm iterations

Table 4. The proposed algorithms solution steps for Burma14

Level	Algorithms	2	3	4	5	6	7	8	9	10	11	12	13	14	Cost
Initial Solution	**IH**	2.3033	-0.1139	0.7853	-2.6272	-0.4115	4.6309	0.4681	0.2114	-2.6841	-0.111	1.2406	1.7914	-1.0448	
	HOI	10	5	14	6	3	11	9	8	4	12	13	2	7	
2opt		2-Opt Not Applied													6290
1	**NH**	-2.6455	-2.651	-1.0227	-0.3836	-0.1139	-0.158	0.2518	0.4681	0.7592	1.2074	2.1269	2.3215	4.5906	
	HOI	3	2	4	5	7	6	8	9	10	11	12	13	14	
2opt	**2-Opt OI**	8	13	7	12	6	5	4	3	14	2	10	9	11	3371
2	**NH**	0.2518	2.3688	-0.1647	2.0943	-0.1139	-0.3919	-1.0296	-2.6904	4.5745	4.2033	0.7515	0.5164	1.2113	
	HOI	9	8	7	4	6	2	13	12	14	5	3	11	10	
2opt	**2-Opt OI**	2	3	4	5	6	12	14	7	13	8	11	9	10	3448
3	**NH**	0.2146	-3.2888	-0.1647	2.0904	-3.436	0.7206	1.2203	-0.4167	0.5488	2.3025	4.244	-2.6809	4.5407	
	HOI	6	3	13	9	4	2	10	7	8	5	11	12	14	
2opt		2-Opt Not Applied													7927
4	**NH**	0.2617	2.3827	-0.1616	2.0943	-0.1314	0.7515	1.1767	-0.3962	0.5164	-1.0119	4.1601	-2.6559	4.5252	
	HOI	13	11	9	4	6	2	10	7	8	5	3	12	14	
2opt	**2-Opt OI**	2	14	3	4	5	6	12	7	13	8	11	9	10	3323
5	**NH**	0.2617	4.5213	2.3827	-0.1395	2.105	-0.0897	4.1389	0.7562	-2.638	1.1976	-1.0169	-0.3962	0.5506	
	HOI	10	12	13	5	7	2	14	9	11	6	4	8	3	
2opt	**2-Opt OI**	2	14	3	4	5	6	12	7	13	8	11	9	10	3323
	Best Solution	**2**	**14**	**3**	**4**	**5**	**6**	**12**	**7**	**13**	**8**	**11**	**9**	**10**	**3323**

IH: Initial Harmonies, NH: New Harmonies, HOI: Harmony Ordered Indexes, 2-Opt OI: 2-Opt Ordered Indexes

3. Computational results

In this section, some benchmark problem sets from TSPLIB95 [40] such as *eil51, berlin52, st70, pr76, eil76, kroA100, kroB100, eil101, bier127, chr130, ch150, kroA150, kroB200 and lin318* are considered. As the simulation platform, i7 CPU and 4 GB RAM hardware and MATLAB® 8.2 software package are used. Also, some functions of the Matlog: Logistics Engineering Matlab Toolbox [41] are used. For the simulations, two different parameter sets are chosen that are given as two cases in Table 5.

Table 5. The parameter settings

Parameters	*Case 1*	*Case 2*
HMS	20	1
r_{accept}	0.6	0.95
r_{pa}	0.7	0.7
v	4	3

For Case 1, the parameter values are taken as the most frequently used ones in the literature. For Case 2, on the other hand, the parameters are obtained by trial and error and especially, in order to shorten the simulation time, and as a result HMS is chosen as 1. By trial and error, the acceptable value of v is taken as 3. For Case 1, the iteration is limited as 3600s or best known solution (BKS) whereas for Case 2, 500s or BKS is used. For both cases, each test instance is executed 100 times and the simulation results are analysed in Table 6 and Table 7 for both cases of Table 5 using the mentioned problem sets of TSPLIB.

In Table 6 and Table 7, #Opt/Run is the number of BKS values obtained in total of 100 runs, BKS is the best known solution, $BSol_j$ and $WSol_j$ are the obtained best and worst solutions of 100 runs of the jth instance, respectively. ASolj is the average of Soli (i=1…100) for jth instance and can be given as

$$ASol_j = \frac{\sum_{i=1}^{100} Sol_i}{100} \qquad i=1..100, \quad j=1..14 \tag{1}$$

where Sol_i (i=1…100) is each solution of 100 runs. ADevj and BDevj are the percentage deviations of the $ASol_j$ and $BSol_j$ from BKSj, respectively, and can be given as

$$ADev_j = \frac{\left|BKS_j - ASol_j\right|}{BKS_j} \times 100, \qquad j = 1..14 \tag{2}$$

$$BDev_j = \frac{\left|BKS_j - BSol_j\right|}{BKS_j} \times 100, \qquad j = 1..14 \tag{3}$$

It can be seen from Table 6 and Table 7 that the $ADev_j$ solutions of Case 2 are all better than Case 1 whereas $BDev_j$ (j=14) of Lin318 is only slightly worse for Case 2. When #Opt/Run values are taken into consideration, one can see that the parameter set of Case 2 is better in finding the BKS values in total simulations.

Table 6. Summary of computational results for the proposed method for Case 1

#	Name	BKS	BSol	WSol	ASol	ADev	BDev	#Opt/Run
1	***eil51***	426.00	426.00	429	426.95	0.22	0.00	33/100
2	***berlin52***	7542.00	7542.00	7542	7542	0.00	0.00	100/100
3	***st70***	675.00	675.00	679	675.79	0.12	0.00	43/100
4	***pr76***	108159.00	108159.00	109161	108538.43	0.35	0.00	3/100
5	***eil76***	538.00	539.00	552	546.9	1.65	0.19	0/100
6	***kroA100***	21282.00	21282.00	21495	21345.38	0.30	0.00	8/100
7	***kroB100***	22141.00	22179.00	22522	22368,68	1.03	0.17	0/100
8	***eil101***	629.00	635.00	654	646.71	2.82	0.95	0/100
9	***bier127***	118282.00	118936.00	121730	120206.68	1.63	0.55	0/100
10	***ch130***	6110.00	6139.00	6309	6244.38	2.20	0.47	0/100
11	***ch150***	6528.00	6598.00	6796	6700.02	2.64	1.07	0/100
12	***kroA150***	26524.00	26846.00	27538	27188.09	2.50	1.21	0/100
13	***kroA200***	29368.00	29693.00	30594	30146.91	2.65	1.11	0/100
14	***lin318***	42029.00	43113.00	44418	43881.24	4.41	2.58	0/100

Table 7. Summary of computational results for the proposed method for Case 2

#	Name	BKS	BSol	WSol	ASol	ADev	BDev	#Opt/Run
1	***eil51***	426.00	426.00	428.00	426.07	0.02	0.00	94/100
2	***berlin52***	7542.00	7542.00	7542.00	7542.00	0.00	0.00	100/100
3	***st70***	675.00	675.00	675.00	675.00	0.00	0.00	100/100
4	***pr76***	108159.00	108159.00	108701.00	108324.39	0.15	0.00	5/100
5	***eil76***	538.00	538.00	546.00	542.46	0.83	0.00	1/100
6	***kroA100***	21282.00	21282.00	21319.00	21293.08	0.05	0.00	34/100
7	***kroB100***	22141.00	22141.00	22356.00	22259.81	0.54	0.00	1/100
8	***eil101***	629.00	634.00	647.00	641.74	2.03	0.79	0/100
9	***bier127***	118282.00	118724.00	120241.00	119527.81	1.05	0.37	0/100
10	***ch130***	6110.00	6133.00	6245.00	6192.18	1.35	0.38	0/100
11	***ch150***	6528.00	6556.00	6719.00	6644.63	1.79	0.43	0/100
12	***kroA150***	26524.00	26690.00	27161.00	26981.45	1.72	0.63	0/100
13	***kroA200***	29368.00	29622.00	30144.00	29896.52	1.80	0.86	0/100
14	***lin318***	42029.00	43153.00	44281.00	43764.46	4.13	2.67	0/100

In addition, the solution times of each case are given in Table 8. Instead of berlin52 (i=2) Case 2 has better simulation times in average. This is an expected situation as HMS value is 1 for Case 2. Fig.2 is given in order to show the results $ADev_j$ and $BDev_j$ respectively. When the deviations for both cases are investigated from Fig.2, it can be seen that $ADev_j$ has more deviation than $BDev_j$.

Table 8. Simulation times for test instances of Case 1 and Case 2

		Case 1 - Time (seconds)			***Case 2 - Time (seconds)***		
#	***Name***	***Best***	***Worst***	***Avg***	***Best***	***Worst***	***Avg***
1	***eil51***	12.14	3600.16	975.68	16.08	503.50	421.35
2	***berlin52***	2.60	344.55	72.35	5.70	451.47	101.48
3	***st70***	2.63	2453.73	430.54	13.16	523.59	383.36
4	***pr76***	145.21	3602.20	3479.21	33.24	523.67	503.19
5	***eil76***	1464.98	3601.34	3579.00	500.00	524.40	507.15
6	***kroA100***	61.60	3606.30	2960.44	38.56	549.28	496.07
7	***kroB100***	2490.69	3605.85	3590.16	500.01	563.28	524.03
8	***eil101***	3600.01	3603.44	3600.76	500.42	543.10	515.96
9	***bier127***	3600.00	3604.34	3601.52	500.14	567.12	534.31
10	***ch130***	3600.02	3604.38	3601.77	501.76	609.22	545.59
11	***ch150***	3600.05	3617.40	3604.29	501.55	681.52	622.67
12	***kroA150***	3600.06	3616.69	3603.84	502.44	708.62	571.90
13	***kroA200***	3600.20	3626.39	3611.84	504.01	1137.15	728.29
14	***lin318***	3600.24	3825.49	3671.98	2046.31	2466.96	2208.23

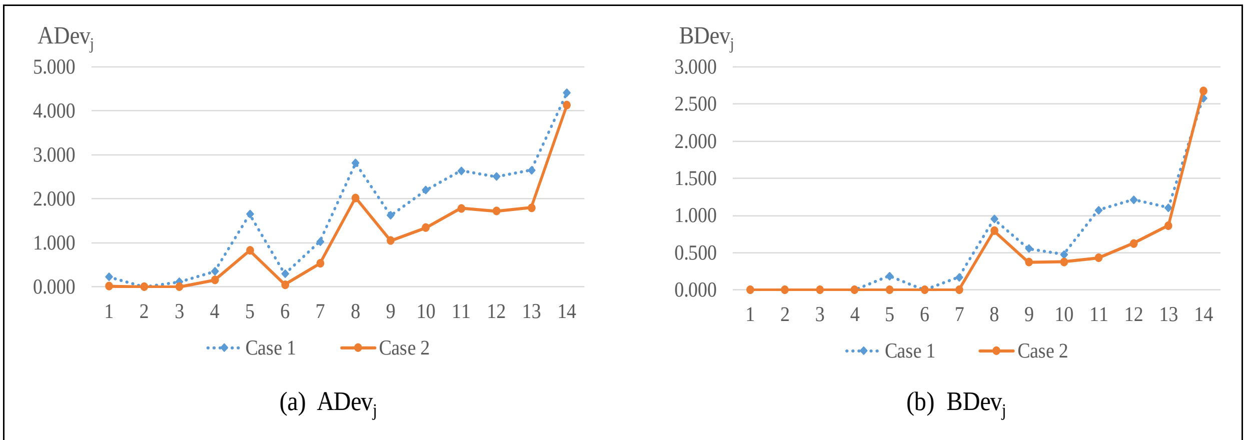

Figure 2. The results of $ADev_j$ and $BDev_j$ for Case 1 and Case 2

In order to show the main contribution of this study, the results of 2-Opt_cHS algorithm is compared with classical 2-Opt solutions. The comparison results are given in Table 9 where it is seen that an improvement is obtained in simulation performances by using the proposed method. The classical 2-Opt is simulated by randomly generating a new route information and then applying only the 2-Opt algorithm at each step.

The performances in Table 9 are also evaluated in Table 10 as an indicator of the solution quality using the $ADev_j$, $BDev_j$ and Opt/Run parameters between 2opt_cHS and classical 2opt algorithms.

Table 9. Solutions for test instances of *2opt_cHS* and *classical 2opt*

			2opt_cHS			*Classical 2-opt*		
#	*Name*	*BKS*	*BSol*	*WSol*	*ASol*	*BSol*	*WSol*	*ASol*
1	***eil51***	426.00	426.00	428.00	426.07	431.00	479.00	449.51
2	***berlin52***	7542.00	7542.00	7542.00	7542.00	7542.00	8821.00	8196.08
3	***st70***	675.00	675.00	675.00	675.00	678.00	774.00	712.13
4	***pr76***	108159.00	108159.00	108701.00	108324.40	108943.00	119484.00	112415.10
5	***eil76***	538.00	538.00	546.00	542.46	548.00	605.00	573.18
6	***kroA100***	21282.00	21282.00	21319.00	21293.08	21367.00	24069.00	22312.73
7	***kroB100***	22141.00	22141.00	22356.00	22259.81	22389.00	25201.00	23407.38
8	***eil101***	629.00	634.00	647.00	641.74	659.00	697.00	676.96
9	***bier127***	118282.00	118724.00	120241.00	119527.80	119408.00	133490.00	126352.70
10	***ch130***	6110.00	6133.00	6245.00	6192.18	6200.00	6909.00	6497.10
11	***ch150***	6528.00	6556.00	6719.00	6644.63	6717.00	7342.00	7019.57
12	***kroA150***	26524.00	26690.00	27161.00	26981.45	27155.00	29305.00	28369.89
13	***kroA200***	29368.00	29622.00	30144.00	29896.52	29856.00	32406.00	31204.24
14	***lin318***	42029.00	43153.00	44281.00	43764.46	43814.00	46295.00	44991.41

According to Table 10, it can be seen from the BKS results of the proposed 2-Opt-cHS algorithm, ADevj and BDevj average deviations are -1.10 and -0.44, respectively. On the other hand, for the classical 2-Opt algorithm, ADevj and BDevj average deviations are -6.38 and -1.72, respectively. Therefore, it is seen that by using 2-Opt together with HS, the performance is improved with respect to classical 2-Opt algorithm. Also, the proposed 2-Opt_cHS algorithm can reach BKS values of seven test instances. However, for the classical 2-Opt algorithm, BKS value of one test instance (berlin52) is obtained. Thus, less deviation values and higher #Opt/Run values of 2-Opt_CHS show the advantage obtained using the proposed method. Thus, it is seen that using only classical 2-Opt at each step, optimal solutions cannot be obtained in general. This is an indicator that the proposed method has an improvement by applying the faster HS at each step and slower 2-Opt at some steps.

Table 11 is designed to compare our solutions with the results obtained from literature. It can be observed that all the average solution performances of meta-heuristics including the basic Discrete Cuckoo Search (DCS) have worse results than proposed 2-Opt_cHS. On the other hand, improved DCS of their study and proposed 2-Opt_cHS in our study, have similar performances and both are better than the other meta-heuristics mentioned in their study.

4. Conclusions and further research

The travelling salesman problems are mostly studied in the class of NP-Hard problems. In order to solve these problems many techniques and solution approaches are designed in the literature. In this paper, a harmony search algorithm is directly used as a solution method. The transformation of real numbers of continuous harmony search algorithm to integer numbers of discrete form is obtained by using index values and the 2-Opt local search algorithm. As computational test instances the problem sets of eil51, berlin52, st70, pr76, eil76, kroA100, kroB100, eil101, bier127, ch130, ch150, kroA150, kroB200 and lin318 from TSPLIB are selected and two different cases are designed for experimental study. The results have shown that acceptable solutions can be obtained with the given algorithm.

The results of the proposed method are compared with conventional 2-Opt algorithm and also with other meta-heuristics. Consequently, it is shown that by using the proposed 2-Opt_cHS algorithm useful results could be obtained. The proposed method can be used for all TSP variants such as production planning, electronic manufacturing, and logistics.

Table 10. Relatively comparison for the solutions of *2opt_cHS* and *classical 2opt*

		2opt_cHS			*2-opt*		
#	*Name*	*ADev*	*BDev*	*#Opt/Run*	*ADev*	*BDev*	*#Opt/Run*
1	***eil51***	-0.02	0.00	94/100	-5.52	-1.17	0/100
2	***berlin52***	0.00	0.00	100/100	-8.67	0.00	1/100
3	***st70***	0.00	0.00	100/100	-5.50	-0.44	0/100
4	***pr76***	-0.15	0.00	5/100	-3.94	-0.72	0/100
5	***eil76***	-0.83	0.00	1/100	-6.54	-1.86	0/100
6	***kroA100***	-0.05	0.00	34/100	-4.84	-0.40	0/100
7	***kroB100***	-0.54	0.00	1/100	-5.72	-1.12	0/100
8	***eil101***	-2.03	-0.79	0/100	-7.62	-4.77	0/100
9	***bier127***	-1.05	-0.37	0/100	-6.82	-0.95	0/100
10	***ch130***	-1.35	-0.38	0/100	-6.34	-1.47	0/100
11	***ch150***	-1.79	-0.43	0/100	-7.53	-2.90	0/100
12	***kroA150***	-1.72	-0.63	0/100	-6.96	-2.38	0/100
13	***kroA200***	-1.80	-0.86	0/100	-6.25	-1.66	0/100
14	***lin318***	-4.13	-2.67	0/100	-7.05	-4.25	0/100
	Avg	-1.10	-0.44		-6.38	-1.72	

Table 11. Comparison of ASol results of the proposed method with different meta-heuristics from the literature

	Compared Test Instances				
Solution Methods	**Eil51**	**Berlin52**	**St70**	**Eil76**	**KroA100**
BKS	426.00	7542.00	675.00	538.00	21282.00
Proposed 2-Opt_cHS	426.07	7542.00	675.00	542.50	21293.10
Basic DCS [20]	439.00	7836.40	696.90	565.70	22419.90
Improved DCS [21]	426.00	7542.00	675.00	538.00	21282.00

References

[1] Laporte G, The Traveling Salesman Problem: An overview of exact and approximate algorithms, European Journal of Operational Research, Vol. 59, pp. 231-247, (1992).

[2] Croes GA, A Method for Solving Travelling-Salesman Problems. Operations Research, Vol. 6, No. 6, pp. 791-812, (1958).

[3] Holland JH, Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, Mass., USA, (1975).

[4] Kirkpatrick S, Gelatt JrCD, Vecchi MP, Optimization by simulated annealing. Science, Vol. 220, No. 4598, pp. 671–680, (1983).

[5] Geem ZW, Kim JH, Loganathan GV, A new heuristic optimization algorithm: harmony search. Simulation, Vol. 76, No. 2, pp. 60–68, (2001).

[6] Yang XS, Harmony Search as a Metaheuristic Algorithm. in Music-Inspired Harmony Search Algorithm. 2009, Springer Berlin / Heidelberg. pp. 1-14, (2009).

[7] Sureja N, New Inspirations in Nature: A Survey. International Journal of Computer Applications & Information Technology, pp.21-24, (2012).

[8] Abdel-Raouf O, Metwally MAB, A Survey of Harmony Search Algorithm. International Journal of Computer Applications, pp.17-26, (2013).

[9] Freisleben B, Merz P, Genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. in Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC ’96), pp. 616–621, (1996).

[10] Chowdhury A, Ghosh A, Sinha S, Das S, Ghosh A, A novel genetic algorithm to solve travelling salesman problem and blocking flow shop scheduling problem. International Journal of Bio-Inspired Computation, Vol. 5, No. 5, pp. 303-314, (2013).

[11] Wang Y, Tian D, An improved simulated annealing algorithm for traveling salesman problem. in Proceedings of the International Conference on Information Technology and Software Engineering, Vol. 211 of Lecture Notes In Electrical Engineering, pp. 525–532, (2013).

[12] Fiechter CN, A parallel tabu search algorithm for large traveling salesman problems. Discrete Applied Mathematics, Vol. 51, No. 3, pp. 243–267, (1994).

[13] Stüzle T, Hoos H, MAX-MIN Ant system and local search for the traveling salesman problem. Reference Future Generations Computer Systems, Vol. 16, No. 8, pp. 889–914, (1997).

[14] Randall M, Montgomery J, The accumulated experience Ant colony for the traveling salesman problem. International Journal of Computational Intelligence and Applications, Vol.3, No.2, pp. 189–198, (2003).

[15] Chu SC, Roddick JF, Pan JS, Ant colony system with communication strategies. Information Sciences, Vol. 167, No. 1–4, pp. 63–76, (2004).

[16] Wang KP, Huang L, Zhou CG, Pang W, Particle swarm optimization for traveling salesman problem. in Proceedings of the International Conference on Machine Learning and Cybernetics, pp. 1583–1585, (2003).

[17] Yang XS, Nature-Inspired Metaheuristic Algorithms. Luniver Press, (2010).

[18] Yang XS, Deb S, Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation, Vol. 1, No. 4, pp. 330-343, (2010).

[19] Ouyang X, Zhou Y, Luo Q, Chen H, A novel discrete cuckoo search algorithm for spherical traveling salesman problem. Applied Mathematics & Information Sciences, Vol. 7, No. 2, pp. 777–784, (2013).

[20] Ouaarab A, Ahiod B, Yang, XS, Discrete Cuckoo Search Algorithm for the Travelling Salesman Problem. Neural Computing and Applications, Vol.24, No.7-8, pp 1659-1669, (2014a).

[21] Ouaarab A, Ahiod B, Yang XS, “Improved and Discrete Cuckoo Search for Solving the Travelling Salesman Problem”, X.-S. Yang In Cuckoo Search and Firefly Algorithm: Theory and Applications, Vol. 516, pp. 63-84, Switzerland: Springer(2014b).

[22] Jati GK, Suyanto S, Evolutionary Discrete Firefly Algorithms for Travelling Salesman Problem. Proceedings of the 2nd International Conference on Adaptive and Intelligent Systems, pp. 393-403, Klagenfurt, Austria: Springer, (2011).

[23] Kumbharana SN, Pandey GM, Solving Travelling Salesman Problem using Firefly Algorithm. International Journal for Research in Science & Advanced Technologies, pp. 53-57, (2013a).

[24] Ismail MM, Che H, Mohd H, Mohamad SH, Jaafar HI, A Preliminary Study on Firefly Algorithm Approach for Travelling Salesman Problem. In: Science & Engineering Technology National Conference 2013, 3-4 July 2013, Kuala Lumpur, Malaysia, (2013).

[25] Wang L, Xu Y, Mao Y, Fei M, A Discrete Harmony Search Algorithm. Communications in Computer and Information Science, pp. 37-43, (2010).

[26] Pan QK, Wang L, Gao L, A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers. Applied Soft Computing, Vol.11(8), pp. 5270-5280, (2011).

[27] Yuan Y, Xu H, Yang J, A hybrid harmony search algorithm for the flexible job shop scheduling problem. Applied Soft Computing, Vol.13(7), pp. 3259-3272, (2013).

[28] Jian H, Qi-yuan P, Diversity Maintaining Harmony Search and Its TSP Solution. Application Research of Computer, pp. 3583-3586, (2013).

[29] Weyland, D, A critical analysis of the harmony search algorithm—How not to solve Sudoku. Operations Research Perspectives, Vol. 2, pp. 97–105, (2015).

[30] Rechenberg I., Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart, (1973).

[31] Geem ZW, Novel derivative of harmony search algorithm for discrete design variables. Applied Mathematics and Computations, Vol.199, pp. 223-230, (2008).

[32] Pang W, Wang KP, Zhou CG, Dong LJ, Fuzzy discrete particle swarm optimization for solving traveling salesman problem. in Proceedings of the 4th International Conference on Computer and Information Technology (CIT '04), pp. 796–800, (2004).

[33] Thamilselvan R, Balasubramanie P, A genetic algorithm with a Tabu search (GTA) for traveling salesman problem. International Journal of Recent Trends in Engineering, Vol. 1, No. 1, pp. 607-610, (2009).

[34] Kaveh A, Talatahari S, Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Computers and Structures, Vol. 87, No. 5-6, pp. 267–283, (2009).

[35] Yan Y, Zhao X, Xu J, Xiao, Z, A mixed heuristic algorithm for traveling salesman problem. in *Proceedings of the 3rd International Conference on Multimedia Information Networking and Security (MINES '11)*, pp. 229–232, (2011).

[36] Chen SM, Chien CY, Parallelized genetic ant colony systems for solving the traveling salesman problem. *Expert Systems with Applications*, Vol. 38, No. 4, pp. 3873–3883, (2011a).

[37] Chen SM, Chien CY., Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications*, Vol. 38, No. 12, pp. 14439–14450, (2011b).

[38] Yun HY, Jeong SJ, Kim KS, Advanced Harmony Search with Ant Colony Optimization for Solving the Traveling Salesman Problem. *Journal of Applied Mathematics*, pp. 1-8, (2013).

[39] Kumbharana SN, Pandey GM, A comparative study of ACO, GA and SA for solving traveling salesman problem. *International Journal of Societal Applications of Computer Science*, Vol. 2, No. 2, pp. 224–228, (2013b).

[40] Reinelt, G, TSPLIB95. Retrieved Oct., 2013 from https://www.iwr.uni-heidelberg.de/groups/comopt/software. (2013).

[41] Kay, M, Matlog: Logistics Engineering Matlab Toolbox. Retrieved Apr., 2014 from. www.ise.ncsu.edu/kay/matlog/ (2014).

Kenan Karagul studied industrial engineering for his Bachelor degree and business administration for MSc and PhD degrees. His field of study includes operations research, logistics, vehicle routing problems, metaheuristics and quantitative models. He worked at various firms between 1996 and 2001. He has been working at Pamukkale University as an instructor since 2001. He was awarded the best PhD thesis on Graduate Tourism Students Congress in Kuşadası (2014).

Erdal Aydemir is an assistant professor in Dept. of Industrial Engineering at Suleyman Demirel University (SDU), Isparta, Turkey. He was received the BSc degree in Industrial Engineering from Selcuk University in 2005 and the MSc degree in Industrial Engineering from SDU in 2009. In 2013, he was awarded a PhD degree in Mechanical Engineering (ME) at SDU. His research interests are EPQ/EOQ models, vehicle/inventory routing problems, uncertainty modelling, grey system theory, decision modelling of production, service systems and its adaptation with artificial intelligence etc. on the various fields.

Sezai TOKAT received the BSc and PhD degrees in control and computer engineering from Istanbul Technical University, Istanbul, Turkey, in 1994 and 2003, respectively. He received his MSc degree in systems and control engineering from Bogazici University, Istanbul, in 1997. Since 2011 he has been with Pamukkale University, Turkey as associate professor. His research interests include intelligent control techniques, robust control, nonlinear control, optimization, vehicle routing problems