Arabian Journal for Science and Engineering (2019) 44:3669–3679
https://doi.org/10.1007/s13369-018-3617-0

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Discrete Sine-Cosine Algorithm (DSCA) with Local Search for Solving Traveling Salesman Problem

Mohamed A. Tawhid[1] · Poonam Savsani[1,2]

Received: 25 February 2018 / Accepted: 28 October 2018 / Published online: 11 November 2018

Abstract

One of the new population-based optimization algorithms, named sine-cosine algorithm (SCA), is introduced to solve continuous optimization problems. SCA utilizes the sine and cosine functions to recast a set of potential solutions to balance between exploration and exploitation in the search space. Many researchers have developed and introduced a modified version of SCA to solve engineering problems, multi-objective version of SCA to solve multi-objective engineering design problems, and a binary version of SCA to deal with datasets. Our goal from this work to propose discrete SCA (DSCA) to solve the traveling salesman problem (TSP). The TSP is one of the typical NP-hard problems. DSCA works on the basic concepts of exploration and exploitation. To balance the exploration and exploitation in DSCA, it uses two different mathematical expressions to update the solutions in each generation. DSCA is combined with 2-opt local search method to improve exploitation. To enhance the exploration heuristic crossover, it is united with the proposed DSCA. A benchmarks problem selected from TSPLIB is used to test the algorithm, and the results show that the DSCA algorithm proposed in this article is comparable with the other state-of-the-art algorithms over a wide range of TSP.

Keywords Traveling salesman problems · Discrete sine-cosine algorithm · NP-hard combinatorial optimization problem · Metaheuristic

1 Introduction

In 2016, Mirjalili [1] introduced a new population-based optimization algorithm, named sine-cosine algorithm (SCA), to solve continuous optimization problems. SCA employs the sine and cosine functions to adjust a set of potential solutions in order to have a balance between exploration and exploitation in the search space. Since then, many researchers have utilized the SCA to solve various continuous optimization problems. For example, Tawhid and Savsani [2] developed a multi-objective sine-cosine algorithm (MO-SCA) to solve multi-objective engineering design problems. As well, Elaziz et al. [3] developed a modified SCA to solve engineering problems. Other researchers developed a binary version for SCA to deal with datasets [4,5], scheduling [6], and unit commitment problem [7]. As such, there is a demand for a discrete version of SCA in order to deal with various combinatorial optimization and complex discrete problems such as TSP, vehicle routing problems, and others. Our goal from this work is to introduce the discrete SCA (DSCA)in order to solve TSP. TSP comprises of a set of cities and the distances between each pair of cities. The aim is to obtain the shortest possible route that visits each city exactly once and returns to the origin city. The tour including each city once is labeled as the Hamiltonian circuit (HC), and the shortest Hamiltonian circuit is the optimal Hamiltonian circuit (OHC). This is for Euclidean TSP. Previously, TSP has been proven to be NP-complete [8]. As such, TSP has been extensively studied in the fields of combinatorial mathematics, graph theory and computer science due to its theoretical and practical values [9]. That being said, there are no polynomial algorithms for the NP-complete problems unless NP = P [10]. As such, the research on competent algorithms for TSP is still one of the most important topics researched today. Over the past years, TSP has become one of the best

✉ Poonam Savsani
poonam.savsani@gmail.com

Mohamed A. Tawhid
mtawhid@tru.ca

1 Department of Mathematics and Statistics, Faculty of Science, Thompson Rivers University, Kamloops, BC V2C 0C8, Canada

2 Department of Industrial Engineering, Pandit Deendayal Petroleum University, Gandhinagar, Gujarat, India

platforms to evaluate the performance of different kinds of algorithms because of its hardness. At present, all the methods used to solve TSP can be divided into two categories, one is the exact methods that guarantee the optimal solution, and the other is an approximation algorithm [11]. Using the exact methods guarantees you the optimal solution, but with the expansion of the problem's scale, the solving time required increases exponentially. As such, it is difficult to apply these methods to solve large-scale problems. Some of the common exact methods include dynamic programming [12], branch and bound [13], depth-first search graph algorithm [14], and the integer programming methods [15] (which are feasible to tackle the TSP with less than 1000 cities [16]). Using powerful turning machines, they can find the OHC within an acceptable computation time. When the scale of TSP becomes larger, the approximate algorithms demonstrate their good performance. Approximation algorithms can also be divided into two categories, the local search algorithms and heuristic optimization methods. Local search algorithms are related to the characteristics of the problems, such as the 2-opt [17], 3-opt [18], LK [19], the LKH [20] and the inver over [21]. These local algorithms can make effective use of the relevant characteristics of the problem in order to find the local optimal solution to the problem. Like their counterparts, when the scale of the problem increases, the computation greatly increases as well. Some of the heuristic optimization methods that have been developed so far, in order to search for the nearly optimal solution for TSP include the ant colony algorithm [22] (ACO), the genetic algorithm (GA) [23], the simulated annealing algorithm (SA) [24], particle swarm optimization(PSO) [25,26], the artificial neural networks (ANN) [27,28] and the artificial immune algorithm(AIS) [29]. Since these heuristic algorithms do not depend on the problem itself, they have a strong global search capability, while still falling into the local optimum. In recent years, many scholars have combined the local search with metaheuristic algorithms to produce a new hybrid algorithm for solving TSP. Examples of these include the mixing of the genetic operators [30] and LK, and memetic algorithm [31]; Yang et al., 2008 [32] proposed a method that combines the ant colony algorithm and mutation strategy; Samanlioglu et al., 2006 [33] proposed a method of combining genetic algorithm with a 2-opt; Gang et al. [34] proposed an improved complete 2-opt (complete 2-opt, C2OPT); Marinakis et al. [35] proposed honey bee mating algorithm for the Euclidean traveling salesman problem optimization; Zhou et al. in [36] proposed a discrete Glowworm swarm algorithm (DGSO); Ouaarab et al. [37] proposed a discrete cuckoo search algorithm(DCS); and the improved genetic algorithm is adopted. Another example is the hybrid genetic algorithm with two optimization strategies O(n) and O(n^3) which is proposed in [38]. The author reported better performance of hybrid GA than GA with 2-opt and classical GA.

As mentioned, in this paper, a discrete sine-cosine algorithm (DSCA) is proposed to solve TSP. Benchmark problems selected from TSPLIB are used to test the algorithm, and results show that the proposed algorithm in this work can achieve near OHC and has strong sturdiness.

The rest of the paper is structured as follows: Section 3 describes a basic TSP and its mathematical forms. Section 4 introduces the basic SCA. Section 5 proposes DSCA and explains in detail. Section 6 presents results and discussions. Finally, the paper is concluded.

2 The Traveling Salesman Problem

The traveling salesman problem (TSP) is one of the most cited NP-hard combinatorial optimization problems because it is so perceptive and easy to understand but difficult to solve. In graph theory, it is described as the weighted graph (WG), $G = (V, E, d)$, where $V = (v_1, v_2, \ldots v_m)$ is the vertex set and $E = |e_{ij}|_{m\times m}$ are the edges set. $v_i (1 \le i \le m)$ is the vertex, and $e_{ij} (1 \le i, j \le n)$ is the edge connecting the two vertices v_i and v_j. $d : E \rightarrow R^+$ is weight function. The weight d is often taken as distance, cost, etc., for various kinds of TSP. If $d_{ij} = d_{ji}$, then the TSP is called symmetric. If $d_{ij} \ne d_{ji}$ for some $i \ne j$, TSP is an asymmetric. In this work, we consider symmetrical TSP.

The purpose of the TSP is to find optimal Hamilton circuit (OHC) that all the vertices are visited once and only once. Given the HC including m vertices, it is represented as HC = $(v_1, v_2, \ldots, v_m, v_1)$, such that minimize f(HC) the sum of all the Euclidean distances d between each city from same path HC and the computational model of symmetrical TSP is given in Eq. 1.

$$f(\mathrm{HC}) = \sum\nolimits_{i=1}^{m-1} d(v_i, v_{i+1}) + d(v_m, v_1) \tag{1}$$

The Euclidean distance d, between any two cities with coordinate (x_1, y_1) *and* (x_2, y_2), is calculated by Eq. 2.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2}$$

As symmetrical TSP is hard to solve, in the present work, symmetrical benchmark problems are taken from TSPLIB to check the performance of proposed DSCA.

3 Basic Sine-Cosine Algorithm

Recently, Mirjalili [1] proposed a sine-cosine algorithm (SCA) as a population-based algorithm, which starts the search process with a random set of solutions named population. All the stochastic optimization algorithm stresses the exploitation and exploration of the search space. SCA balances the exploration and exploitation by utilizing two different mathematical expressions to update the solutions in each generation. These expressions are defined as follows:

$$X_j^{g+1} = X_j^g + r_1 \sin(r_2)|r_3(Xb)_i^g - X_j^g| \quad (3)$$

$$X_j^{g+1} = X_j^g + r_1 \cos(r_2)|r_3(Xb)_i^g - X_j^g| \quad (4)$$

where X_j^g denotes the current solution with g-th generation and j-th dimension. Xb describes the position solution. Normally, the position solution is the best solution from the population. r_1, r_2 and r_3 are the random numbers which man-

Table 1 Results of the DSCA algorithm for symmetric TSP instances from TSPLIB

	Best	Worst	Mean	Std	PDbest (%)	PDavg (%)	Optimal
eil51	**426**	427	426.4	0.54772	0	0.23474	426
berlin52	**7542**	7542	7542	**0**	0	0	7542
st70	**675**	675	675	**0**	0	0	675
pr76	**108,159**	108,159	108,159	**0**	0	0	108,159
eil76	**538**	539	538.2	0.44721	0	0.18587	538
kroA100	**21,282**	21,282	21,282	**0**	0	0	21,282
kroB100	**22,141**	22,141	22,141	**0**	0	0	22,141
kroC100	**20,749**	20,749	20,749	**0**	0	0	20,749
kroD100	**21,294**	21,309	21,300	8.21584	0	0.07044	21,294
kroE100	**22,068**	22,068	22,068	**0**	0	0	22,068
eil101	**629**	630	629.2	0.44,721	0	0.15898	629
lin105	**14,379**	14,379	14,379	**0**	0	0	14,379
pr107	**44,303**	44,303	44,303	**0**	0	0	44,303
pr124	**59,030**	59,030	59,030	**0**	0	0	59,030
bier127	**118,282**	118,282	118,282	**0**	0	0	118,282
ch130	6111	6140	6124	12.69	0.016364	0.47455	6111
pr136	96,928	97,378	97164.6	187.097	0.160944	0.46426	96,928
pr144	**58537**	58,537	58,537	**0**	0	0	58,537
ch150	**6528**	65,28	6528	**0**	0	0	6528
kroA150	**26524**	26,528	26525.4	1.51658	0	0.015081	26,524
kroB150	**26130**	26,154	26134.8	10.73313	0	0.091848	26,130
pr152	**73682**	73,682	73682	**0**	0	0.000000	73,682
rat195	**2323**	2332	2327.2	4.0249	0	0.38743	2323
d198	15788	15807	15799.8	7.0852	0.050671	0.12034	15,788
kroA200	**29368**	29368	29368	**0**	0	0	29,368
kroB200	29447	29,497	29467.4	18.2975	0.033959	0.16980	29,447
ts225	**126643**	126,810	126709.8	91.4697	0	0.13187	126,643
tsp225	**3916**	3919	3917	1.4142	0	0.07661	3916
pr226	**80369**	80410	80380.4	17.9388	0	0.05101	80,369
gil262	2384	2388	2386.6	1.9494	0.251678	0.16779	2384
pr264	**49135**	49135	49135	**0**	0	0	49,135
a280	**2579**	2588	2584.6	3.3615	0	0.34897	2579
pr299	48250	48,355	48306.8	48.1009	0.122280	0.21762	48,250
lin318	42167	42,284	42221.4	55.5905	0.327270	0.27747	42,167
rd400	15408	15,453	15422.6	18.1604	0.824247	0.29206	15,408
fl417	11920	11,943	11933.4	8.9610	0.494966	0.19295	11,920
pr439	107326	108,152	107588.2	331.3302	0.101560	0.76962	107,326
rat575	6881	6923	6898.6	18.1191	1.569539	0.61038	6881
rat783	9343	9465	9402.4	53.9194	5.747619	1.30579	9343
pr1002	272,323	273,345	272739.6	515.1304	4.875828	0.37529	272,323
nrw1379	60,247	61,230	60,653.4	501.0886	5.990340	1.63162	60,247

age the exploitation and exploration of the search space. r_1 changes linearly from a constant value a as follows:

$$r_1 = a - a\left(\frac{g}{g_m}\right) \tag{5}$$

where g denotes the current generation, g_m is the maximum number of generations and a is a constant defined by the user. r_2 is a random number in the interval $[0, 2\pi]$. r_3 is a random number which can take a value less than or greater than 1. So, r_3 utilizes the following expression:

$$r_3 = (b)R_1 \tag{6}$$

where R_1 is a random number in the interval [0,1] and b $>$ 1 is a given constant by the user.

The solution is updated by utilizing the probability P_r as follows:

$$X_j^{g+1} = X_j^g + r_1 \sin(r_2)|r_3(Xb)_i^g - X_j^g| \quad if\ P_r < R_2 \tag{7}$$

$$X_j^{g+1} = X_j^g + r_1 \cos(r_2)|r_3(Xb)_i^g - X_j^g| \quad if\ P_r > R_2 \tag{8}$$

where R_2 is a random number varying in the interval [0,1].

As the equations mentioned above show, there exist four main parameters in SCA, namely, r_1, r_2, r_3, and r_4. The parameter r_1 *determines* the next position's region (or movement direction) which could be either outside the space, between the destination and solution, or in it. The parameter r_2 describes how far the movement should be outwards or toward the destination. The parameter r_3 brings a random

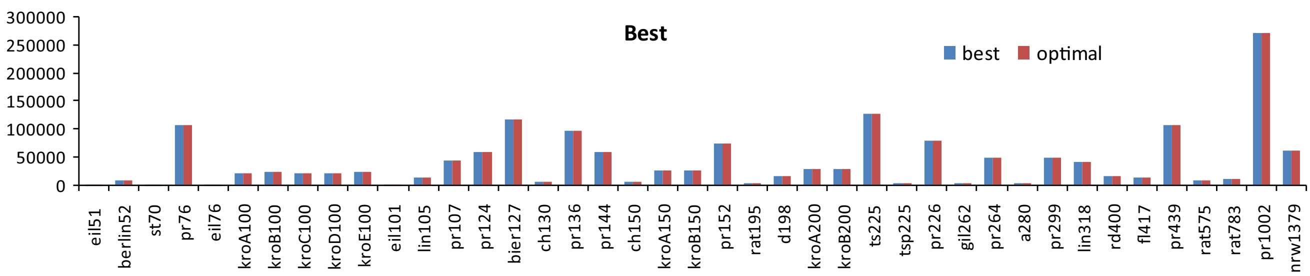

Fig. 1 Comparison of obtained best solutions with the known optimum solutions

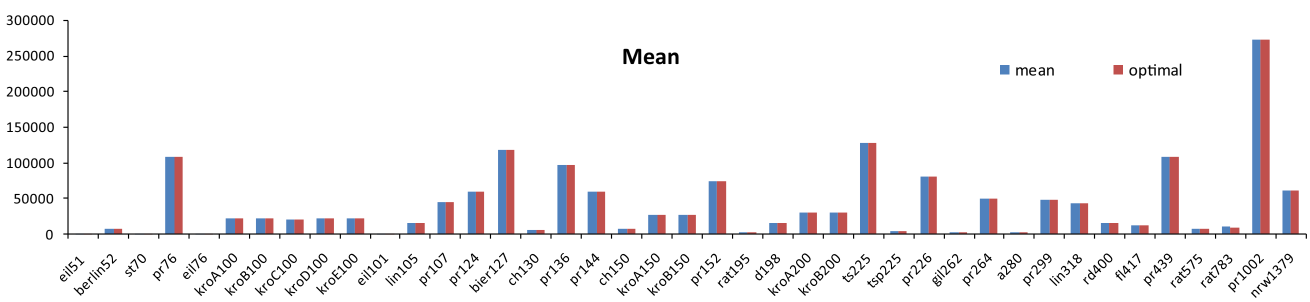

Fig. 2 Comparisons of mean value to the known best solutions

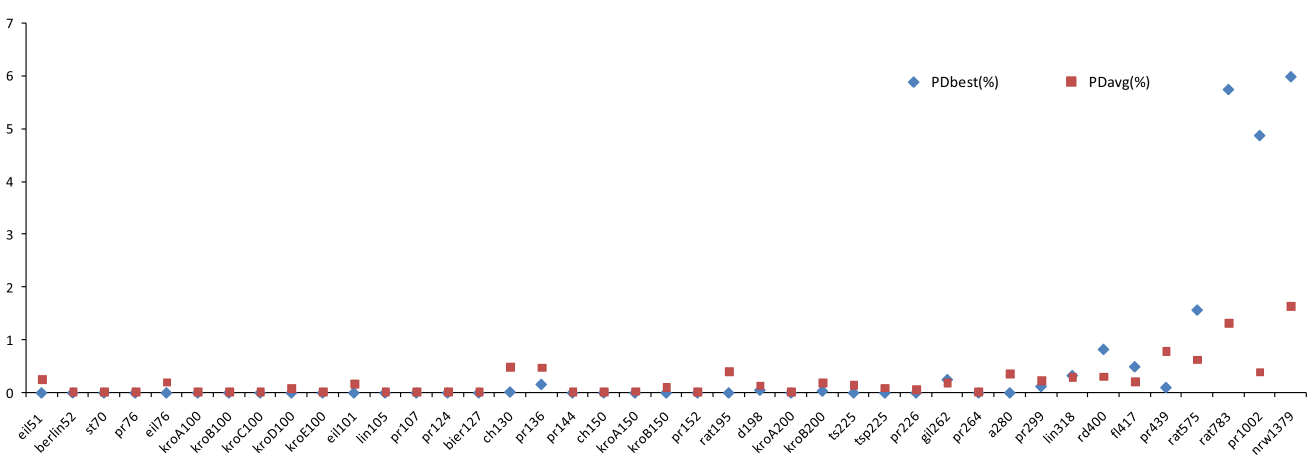

Fig. 3 Percentage deviation for the mean and the best solutions

weight for the destination in order to stochastically maximize ($r_3 > 1$) or minimize ($r_3 < 1$) the effect of destination in defining the distance. Finally, the parameter r_4 equally swaps between the cosine and sine components. This algorithm is called the sine-cosine algorithm (SCA) because of the use of sine and cosine in this formulation. The cyclic pattern of sine and cosine function updates the existing solution near to the optimal solution. This can ensure the exploitation of the defined space between two solutions. For exploring the search space, the solutions should be able to search outside the space between their corresponding destinations as well. This can be attained by changing the range of the cosine and sine functions.

The procedure for the basic SCA is given in Algorithm 1.

Algorithm 1. Basic sine-cosine algorithm (SCA)

Initialize parameters a, b, P_r

Generate initial Population

For $i = 1{:}N$

Calculate r_1, r_2, r_3 *and* R_1

Identify best solution Xb

If $R_1 < P_r$

$$X_j^{g+1} = X_j^g + r_1 \sin(r_2)\left|r_3(Xb)_i^g - X_j^g\right|$$

Else

$$X_j^{g+1} = X_j^g + r_1 \cos(r_2)\left|r_3(Xb)_i^g - X_j^g\right|$$

End if

End for

4 Discrete Sine-Cosine Algorithm (DSCA) with Heuristic Crossover and 2-opt Local Search for TSP

This section describes the extension version of SCA for the TSP. DSCA operates on the same principle as SCA.

In the DSCA, the various HCs, which is made up of different cities, are updated with the help of the following equations:

$$\mathrm{HC}_j^{g+1} = \mathrm{HC}_j^g + r_1 \sin(r_2)|r_3(HCb)_i^g - \mathrm{HC}_j^g| \quad if\ P_r < R_2 \tag{9}$$

$$\mathrm{HC}_j^{g+1} = \mathrm{HC}_j^g + r_1 \cos(r_2)|r_3(\mathrm{HC}b)_i^g - \mathrm{HC}_j^g| \quad if\ P_r > R_2 \tag{10}$$

The parameters r_1, r_2, r_3, and r_4 are calculated in the same manner as in the basic SCC. In DSCA, the length of HC depicts the objective function value, and HCb represents the best HC in the generation. Since the DSCA is a metaheuristic approach, it is strong at exploration rather than exploitation. As such, there is a need to make it suitable for combinatorial problems such as TSP. Hence, it is logical to add local search to make it more effective. Two such local searches are the heuristic crossover [39] and the 2-opt [17]. These local searches are operated in a probabilistic manner. Either of the local searches is imposed on the best solution by the following conditions: If $R_{l1} < R_{l2}$

Operate Heuristic crossover

Else

Operate 2-opt local search

Endif

where R_{l1} and R_{l2} are random numbers between 0 and 1. We use probability with the local search in order to give them separate chances to update the best solution in each generation. As such, if the best solution is not updated in a particular generation, it still gets the chance to be updated by another local search.

The procedure for the heuristic crossover and 2-opt is explained in Algorithms 2 and 3, respectively.

Algorithm 2. Heuristic crossover

C=number cities

Assume heuristic factor (HF)

for k=1:C

select $HC_k, HC_j \quad k \neq j, j = 1$

for i = HF:C-1

$$u_k = HC_k(i)$$

$$u_{k+1} = HC_k(i+1)$$

$$u_j = HC_j(p), \qquad p = position\left(u_k\ in\ HC_j\right)$$

$$u_{j+1} = HC_j(p+1), \qquad if\ p = C \to (p+1) = 1$$

if $d(u_k, u_{k+1}) > d(u_j, u_{j+1})$

$$HC_k(i+1) = HC_j(p+1)$$

end if

end for

end for

Algorithm 3. 2-opt local search

Assume 2-opt factor(2OF)

2-opt(HC, i, j)

If $d(v_i, v_j) + d(v_{i+1}, v_{j+1}) < d(v_i, v_{i+1}) + d(v_j, v_{j+1})$

(v_i, v_{i+1}) *and* (v_j, v_{j+1}) *will be deleted*

(v_i, v_j) *and* (v_{i+1}, v_{j+1}) *will be added*

Reverse the part between cities v_{i+1}, v_j

New_HC= HC[1:i-1]+reverse of HC[i:j]+HC[j:C];

end if

repeat until no improvement is made

start_again:

best_distance = calculateTotalDistance(existing_route)

for i = 0:C-2OF

for j = i + 1:C

if $F(HC_j) > F(HC_k)$

new_HC = 2OF(HC_k*, i, j)*

new_distance = calculateTotalDistance(new_route)

else

new_HC = 2opt(HC_j*, i, j)*

new_distance = calculateTotalDistance(new_route)

end if

i f new_distance<best_distance

HC_j*= new_HC*

goto start again

end if

end for

end for

end while

The procedure for DSCA for TSP is explained in Algorithm 4

Algorithm 4. DSCA with local search

Randomly initialize a population of HC

while termination criterion is not met do

for each HC do

identify HCb^g

Generate new solution $({HC_i}^{g+1})$ using equation 9 or 10

if F$({HC_i}^{g+1})$ < F(HC_i^g) then

Identify the best solution $(HC_{best})^{g+1}$ and $HC_{best} = (HC_{best})^{g+1}$

Else $HC_{best} = (HC_{best})^g$

If $R_1 < R_2$

Update HC_{best} by heuristic crossover

Else

Update HC_{best} by 2-opt local search

end while

5 Results and Discussion

In order to test the performance of proposed DSCA, 41 experimental test cases of symmetrical TSP are taken from the TSPLIB library http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95. Each test case in the simulation is operated independently 20 times. The DSCA is coded in MATLAB R2014a, and the results are obtained on CORE i3, 2.27 GHz processor. For the evaluation of DSCA on TSP, population sizes of 15 and 200 generations are considered. Table 1 shows the numerical results of the DSCA algorithm for different TSP instances. The numbers shown in bold indicate that DSCA reaches OHC for that particular instance. These results are obtained by calculating the Euclidean distances between the vertices. The first column in the table represents the name of the symmetric TSP benchmark, ended by the number of cities. The second column depicts the best result obtained by DSCA. As for the third column, it indicates the average solution for 20 runs. The fourth column shows the standard deviation of solutions obtained by DSCA algorithm over 20 independent runs. The fifth and sixth column shows the percentage best solution length found over the optimal length of 20 runs "PDbest%" and the percentage average solution length found over 20 runs "PDavg%," respectively. Finally, the last column gives best known from TSPLIB.

As discussed above, the percentage deviation of a solution in contrast to the optimal solution is calculated by the following formula:

Table 2 Comparison of average tours of DSCA with other state-of-the-art algorithms for different TSP instances

	Eil51	Berlin52	Eil76	St70	KroA100	Lin105	KroA200	Ch150	Eil101
Best Known Solution	426	7542	538	675	21,282	14,379	29,368	6528	629
Proposed -DSCA	**426.4**	**7542**	**538.2**	**675**	**21282**	**14379**	**29368**	**6568**	**629.2**
ACOMAC [40]	430.68		555.7		21457				
ACOMAC + NN [40]	430.68		555.9		21433.3				
RABNET-TSP [41]	438.7	8073.97	556.1		21868.47	14702.17	30257.53	6753.2	654.83
Modified RABNET-TSP [42]	437.47	7932.5	556.33		21522.73	14400.7	30190.27	6738.37	648.63
GSA ACO PSO [43]	427.27	**7542**	540.2		21370.3	14406.37	29738.73	6563.7	635.23
IVRS + 2-opt [44]	431.1	7547.23			21498.61				648.67
ACO + 2-opt [44]	439.25	7556.58			23441.8				672.37
HACO [45]	431.2	7560.54							
CGAS [46]		7634	542		21437		29,946		
WFA with 2-opt [47]	426.65	**7542**	541.22		**21282**	**14379**	29654.03	6572.13	639.87
WFA with 3-opt [47]	426.6	**7542**	539.44		21282.8	14459.4	29646.5	6700.1	633.5
ACO with Taguchi [48]	435.4	7635.4			21567.1	14475.2			655
ACO with ABC [49]	443.39	7544.37	557.98	700.58	22435.31			6677.12	683.39
HGA+2 local [38]	429.19	7544.37	546.06	677.39	21312.45	14422.89	29458.81	6557.69	644.82
PSO–ACO–3-opt [50]	426.45	7543.2	538.3	678.2	21445.1	14379.15	29646.05	6563.95	632.7
ACE [51]	426.818	7543	538.31	676.41	21298.6	14385.5		6550	633.619

Table 3 Comparison of DSCA with neuro-immune network [41], GCGA with local search [52], Massutti and Castro's method [42], GSA ant colony system with PSO [43], HGA [38], ACE [51] and improved BA [53]

	optimum	DSCA	neuro-immune-network [41]	GCGA with local search [52]	Massutti and Castro's method [42]	GSA ant colony system with PSO [43]	HGA [38]	ACE [51]	Improved BA [53]
eil51	426	**426.4**	438.7	430	437.47	427.27	429.19	426.818	428.1
berlin52	7542	**7542**	8073.97		7932.5	**7542**	7544.37	7543.04	**7542**
st70	675	**675**		678			677.39	676.418	679.1
pr76	108159	**108159**		108942			108255.9	108251	
eil76	538	**538.2**	556.1	551	556.33	540.2	546.06	538.311	548.1
kroA100	21282	**21282**	21868.47	21543	21522.73	21370.47	21312.45	21298.6	21445.3
kroB100	22141	**22141**	22853.6	22542	22661.47	22282.87			22506.4
kroC100	20749	**20749**	21231.6	21025	20971.23	20878.97	20812.22		21050
kroD100	21294	**21300**	22027.87	21809	21697.37	21620.47	21344.67		21593.4
kroE100	22068	**22068**	22815.5	22379	22715.63	22183.47			22349.6
eil101	629	**629.2**	654.83	646	648.63	635.23	644.82	633.619	646.4
lin105	14379	**14379**	14702.23	14544	14400.17	14406.37	14422.89	14385.5	
pr107	44303	**44303**		44909			44341.67		44793.8
pr124	59030	**59030**		59141			59094.13		59412.1
bier127	118282	**118282**	121780.3	120412	120886.3	119421.8			
ch130	6110	**6124**	6231.77		6282.4	6205.63	6130.277	6153.96	
pr136	96772	97164.6		99505			**97019.29**		99351.2
pr144	58537	**58537**		58564			58537.22		58876.2
ch150	6528	**6528**	6753.2		6738.37		6557.69	6550	
kroA150	26524	**26525.4**	27346.43	27298	27355.97	26899.2	26597.78		
kroB150	26130	**26134.8**	26752.13	26682	26631.87	26448.33	26335.85		
pr152	73682	**73682**		74582			73765.7	73766.8	74676.9

Table 3 continued

	optimum	DSCA	neuro-immune-network [41]	GCGA with local search [52]	Massutti and Castro's method [42]	GSA ant colony system with PSO [43]	HGA [38]	ACE [51]	Improved BA [53]
rat195	2323	**2327.2**		2420			2356.02		
d198	15780	**15799.8**		16084			15963	15813.3	
kroA200	29368	**29368**	30257.53	29910	30190.27	29738.73	29458.81		
kroB200	29437	**29467.4**	30415.6	30627	30135	30035.23	29583.38		
ts225	126643	**126709.8**		128016			128295.7		
tsp225	3916	**3917**					3892.88		
pr226	80369	**80380.4**		80969			80534.39		
gil262	2378	**2386.6**		2515					
pr264	49135	**49135**		50344			49163.26		50908.3
pr299	48191	**48306.8**		50812			49757.66		49674.1
lin318	42029	**42221.4**	43704.97	44191	43696.87	43002.9	42877.24		
rd400	15281	**15422.6**		16420			16143.96		
fl417	11861	**11933.4**		12243					
pr439	107217	**107588.2**		113787			111210		
rat575	6773	**6898.6**	7125		7115.67	6933.87			
rat783	8806	9402.4	9326		9343.77	**9079.23**			

$$\text{PD solution\%} = \frac{\text{solution length} - \text{optimal solution length}}{\text{optimal solution length}} \times 100$$

According to the values displayed in Table 1, DSCA gives OHC for 27 TSP instances, and for the remaining, it gives near OHC. From the table, it is observed that DSCA gives zero standard deviation in 16 TSP. As well, results in Table 1 indicate that the DSCA is very efficient in solving both small-scale and large-scale TSP. Also, the variation of the obtained best solution is compared with the known optimum solutions graphically. This is shown in the form of bar charts in Fig. 1. From the results, one can state that for small-scale problems less than 300 cities, DSCA is capable of finding the optimum solutions efficiently, and after 300 cities, DSCA can still find the near OHC. As DSCA is a heuristic approach to TSP instances, the effects of the algorithms can be precisely judged from the mean value of the best solutions obtained in separate runs. Such variation is presented in Fig. 2, and it can be noted that as the number of cities increases, the effectiveness of DSCA decreases. The percentage deviation for the best and the mean solutions is presented in Fig. 3. It can be noted from results that as the number of cities increases, the performance of the DSCA decreases; however, DSCA is still capable of attaining close to the global solution for the TSP instances.

The results of the proposed method are compared with the other well-known state-of-the-art algorithms available in the literature. This includes different variants of ACO including ACO with multiple ant clans (ACOMAC), ACOMAC with the nearest neighborhood, ACO with simulated annealing, ACO with 2-opt, hybrid ACO, ACO with Taguchi, and finally, ACO with ABC. The algorithms are also compared with other methods such as the hybrid GA, GA with ACO, the water flow algorithm, the hybrid PSO, the RABNET, and IVRS. The results are given in Table 2, which presents the average value of the solutions. It is observed from the results that DSCA is superior when compared with the other algorithms for all the tested problems.

Table 4 DSCA comparison with DIWO [54]

	PD % average	
	DIWO [54]	proposed DSCA
eil51	0.6999	0.23474
berlin52	0.0313	0
st70	0.3125	0
kroA100	0.0375	0
kroB100	0.8816	0
pr107	0.4837	0
pr136	0.94	0.46426
kroA150	0.778	0.015081
kroB150	0.3229	0.091848
d198	0.6691	0.12034
tsp225	2.3949	0.07661
pr226	0.2238	0.036084
rd400	2.4229	0.29206
pr1002	3.1873	0.37529

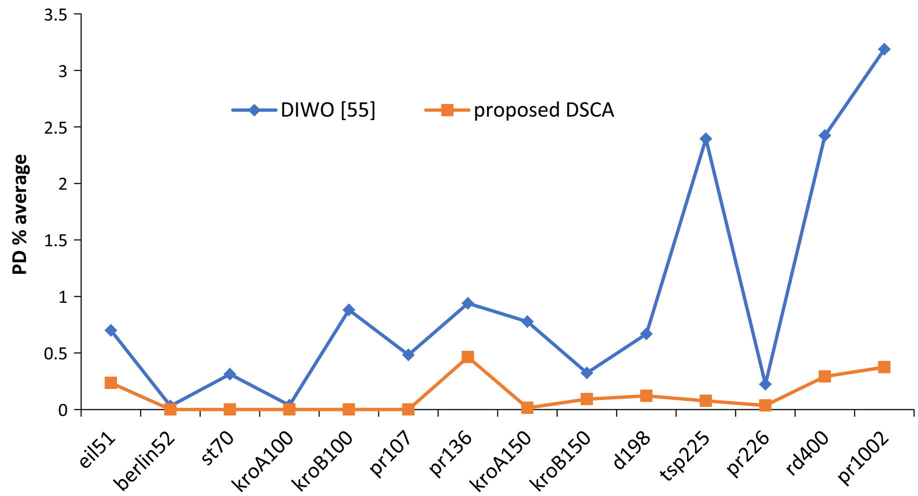

Fig. 4 Graphical representation of comparisons of proposed DSCA and DIWO [54]

Furthermore, to enhance the comparison of the proposed algorithm, we compare it with a wide range of problems, such as other researcher's work. Comparison is made for 38 TSP instances ranging from 51 cities to 783 cities with neuro-immune network [41], GCGA with local search [52], Massutti and Castro's method [42], GSA ant colony system with PSO [43], HGA [38], ACE [51] and improved BA [53]. Table 3 depicts the outperformance of DSCA over all other algorithms. The results indicate that HGA [38] performs better in the performance of pr136 and rat783, while DSCA is better in the remaining 36 TSP instances.

DIWO [54] has presented results for various TSP instances in terms of %PD average, for TSP instances ranging from 51 cities to 1002 cities. Comparison of DSCA with DIWO [54] is depicted in Table 4, which shows the superior performance of DSCA for all the considered TSP instances over DIWO [54]. Figure 4 shows the graphical comparison of DSCA and DIWO [54].

6 Conclusions and Future Work

In this work, we propose a discrete sine-cosine algorithm (DSCA) that can be used to solve traveling salesman problems. We test the performance of the DSCA by applying it to 41 different benchmark problems obtained from the TSP library. Results indicate that our proposed algorithm gives the optimal HCs for 27 instances of TSP and near OHCs for the remaining instances. Furthermore, we also compare the DSCA with other state-of-the-art algorithms derived from the genetic algorithm, ant colony optimization, particle swarm optimization, artificial bee colony optimization, and other methods in the same category. Our comparison demonstrates the competitive performance of DSCA over the other state-of-the-art methods. As such, this work provides us with areas to investigate and new directions to pursue as future works. For example, we intend to apply our proposed algorithm to solve other combinatorial optimization problems such as mixed-integer programming problems, vehicle routing [55], and scheduling [56]. Finally, we would like to generalize this work to solve other types of TSP instances such as generalized TSP [57], spherical [58] and asymmetric [59,60].

Acknowledgements We would like to thank the reviewers for their thoughtful comments and efforts toward improving our manuscript. Also, we are grateful to Marium Tawhid for editing this paper. The research of the Mohamed A. Tawhid is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). NSERC also supports the postdoctoral fellowship of the Poonam Savsani by NSERC.

References

1. Mirjalili, S.: SCA: a sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. **96**, 120–133 (2016)
2. Tawhid, M.A.; Savsani, V.: Multi-objective sine-cosine algorithm (MO-SCA) for multi-objective engineering design problems. Neural Comput. Appl. (2017). https://doi.org/10.1007/s00521-017-3049-x
3. Elaziz, M.A.; Oliva, D.; Xiong, S.: An improved opposition-based sine cosine algorithm for global optimization. Expert Syst. Appl. **90**, 484–500 (2017)
4. Li, S.; Fang, H.; Liu, X.: Parameter optimization of support vector regression based on sine cosine algorithm. Expert Syst. Appl. **91**, 63–77 (2018)
5. Kumar, V.; Kumar, D.: Data clustering using sine cosine algorithm: Data clustering using SCA. In: Hassanien, E., Gaber, T. (eds.) Handbook of Research on Machine Learning Innovations and Trends, pp. 715–726. IGI Global (2017)
6. Das, S.; Bhattacharya, A.; Chakraborty, A.K.: Solution of short-term hydrothermal scheduling using sine cosine algorithm. Soft Comput. **22**(19), 6409–6427 (2018)
7. Reddy, K.S.; Panwar, L.K.; Panigrahi, B.K.; Kumar, R.: A new binary variant of sine-cosine algorithm: development and application to solve profit-based unit commitment problem. Arab. J. Sci. Eng. **43**(8), 4041–4056 (2018)

8. Zhang, W.; Korf, R.E.: A study of complexity transitions on the asymmetric traveling salesman problem. Artif. Intell. **81**(1–2), 223–239 (1996)
9. Rodríguez, A.; Ruiz, R.: The effect of the asymmetry of road transportation networks on the traveling salesman problem. Comput. Oper. Res. **39**(7), 1566–1576 (2012)
10. Berman, P.; Karpinski, M.: 8/7-approximation algorithm for (1, 2)-TSP. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, pp. 641–648. Society for Industrial and Applied Mathematics (2006)
11. He, J.; Yan, H.; Qiang, L.; Hong, Y.: Fat computational complexity and heuristic design for the TSP. J. Softw. **20**(9), 2344–2351 (2009)
12. Bellman, R.; Dreyfus, S.E.: Applied Dynamic Programming, vol. 2050. Princeton University Press, Princeton (2015). ISBN 1400874653, 9781400874651
13. Lawler, E.L.; Wood, D.E.: Branch-and-bound methods: a survey. Oper. Res. **14**(4), 699–719 (1966)
14. Gregor, D.; Lumsdaine, A.: The parallel BGL: a generic library for distributed graph computations. Parallel Object-Oriented Sci. Comput. **2**, 1–18 (2005)
15. Climer, S.; Zhang, W.X.: Cut-and-solve: An iterative search strategy for combinatorial optimization problems. Artif. Intell. **170**(8–9), 714–738 (2006)
16. Johnson, D.S.; McGeoch, L.A.: Experimental analysis of heuristics for the STSP. In: Gutin, G., Punnen, A.P. (eds.) The Traveling Salesman Problem and Its Variations. Combinatorial Optimization, vol. 12, pp. 369–443. Springer, Boston, MA (2007)
17. Croes, G.A.: A method for solving traveling-salesman problems. Oper. Res. **6**(6), 791–812 (1958)
18. Lin, S.: Computer solutions of the traveling salesman problem. Bell Syst. Techn. J. **44**(10), 2245–2269 (1965)
19. Lin, S.; Kernighan, B.W.: An effective heuristic algorithm for the traveling- salesman problem. Oper. Res. **21**(2), 498–516 (1973)
20. Helsgaun, K.: An effective implementation of the Lin–Kernighan traveling salesman heuristic. Eur. J. Oper. Res. **126**(1), 106–130 (2000)
21. Guo, T.; Michalewicz, Z.: Invor-Over Operator for the TSP-Proceedings of the 5th Parallel Problem Solving from Nature Conference (1998)
22. Junzhong, J.; Huang, Z.; Chunnian, L.: An ant colony algorithm based on multiple-grain representation for the traveling salesman problems. J. Comput. Res. Dev. **47**(3), 434–444 (2010)
23. Shu, J.L.; Zhao, Z.; Dai, Q.Y.: Genetic algorithm for TSP. Oper. Res. Manag. Sci. **13**(1), 17–22 (2004)
24. Kirkpatrick, S.; Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
25. Kennedy, J.; Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: Systems, Man, and Cybernetics, 1997. 1997 IEEE International Conference on, Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108. IEEE (1997)
26. Chen, W.N.; Zhang, J.; Chung, H.S.; Zhong, W.L.; Wu, W.G.; Shi, Y.H.: A novel set-based particle swarm optimization method for discrete optimization problems. IEEE Trans. Evolut. Comput. **14**(2), 278–300 (2010)
27. Liu, X.; Xiu, C.: A novel hysteretic chaotic neural network and its applications. Neurocomputing **70**(13), 2561–2565 (2007)
28. Han, F.; Ling, Q.H.; Huang, D.S.: An improved approximation approach incorporating particle swarm optimization and a priori information into neural networks. Neural Comput. Appl. **19**(2), 255–261 (2010)
29. Hunt, J.E.; Cooke, D.E.: Learning using an artificial immune system. J. Netw. Comput. Appl. **19**(2), 189–212 (1996)
30. Merz, P.; Freisleben, B.: Genetic local search for the TSP: new results. In: IEEE International Conference on Evolutionary Computation, 1997, pp. 159–164. IEEE (1997)
31. Bontoux, B.; Artigues, C.; Feillet, D.: A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. Comput. Oper. Res. **37**(11), 1844–1852 (2010)
32. Yang, J.; Shi, X.; Marchese, M.; Liang, Y.: An ant colony optimization method for generalized TSP problem. Prog. Nat. Sci. **18**(11), 1417–1422 (2008)
33. Samanlioglu, F.; Ferrell, W.G.; Kurz, M.E.: A memetic random-key genetic Algorithm for a symmetric multi-objective traveling salesman problem. Comput. Ind. Eng. **55**(2), 439–449 (2008)
34. Gang, P.; Iimura, I.; Nakayama, S.: An evolutionary multiple heuristic with genetic local search for solving TSP. Int. J. Inf. Technol. **14**(2), 1–11 (2008)
35. Marinakis, Y.; Marinaki, M.; Dounias, G.: Honey bees mating optimization algorithm for the Euclidean traveling salesman problem. Inf. Sci. **181**(20), 4684–4698 (2011)
36. Zhou, Y.Q.; Huang, Z.X.; Liu, H.X.: Discrete glowworm swarm optimization algorithm for TSP problem. DianziXuebao(Acta Electronica Sinica) **40**(6), 1164–1170 (2012)
37. Ouaarab, A.; Ahiod, B.; Yang, X.S.: Discrete cuckoo search algorithm for the traveling salesman problem. Neural Comput. Appl. **24**(7–8), 1659–1669 (2014)
38. Wang, Y.: The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem. Comput. Ind. Eng. **70**, 124–133 (2014)
39. Liu, W.; Zheng, J.; Wu, M.; Zou, J.: Hybrid crossover operator based on pattern, Seventh International Conference on Natural Computation (ICNC) 2011, vol. 2, pp. 1097–1100 (2011)
40. Tsai, C.F.; Tsai, C.W.; Tseng, C.C.: A new hybrid heuristic approach for solving large traveling salesman problem. Inf. Sci. **166**(1), 67–81 (2004)
41. Pasti, R.; De Castro, L.N.: A neuro-immune network for solving the traveling salesman problem. In: IJCNN'06. International Joint Conference on Neural Networks, 2006. pp. 3760–3766. IEEE (2006)
42. Masutti, T.A.; de Castro, L.N.: A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem. Inf. Sci. **179**(10), 1454–1468 (2009)
43. Chen, S.M.; Chien, C.Y.: Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. Expert Syst. Appl. **38**(12), 14439–14450 (2011)
44. Jun-man, K.; Yi, Z.: Application of an improved ant colony optimization on generalized traveling salesman problem. Energy Procedia **17**, 319–325 (2012)
45. Junqiang, W.; Aijia, O.: A hybrid algorithm of ACO and delete-cross method for TSP. In: 2012 International Conference on Industrial Control and Electronics Engineering (ICICEE), pp. 1694–1696. IEEE (2012)
46. Dong, G.; Guo, W.W.; Tickle, K.: Solving the traveling salesman problem using cooperative genetic ant systems. Expert Syst. Appl. **39**(5), 5006–5011 (2012)
47. Othman, Z.A.; Srour, A.I.; Hamdan, A.R.; Ling, P.Y.: Performance water flow-like algorithm for TSP by improving its local search. Int. J. Adv. Comput. Technol. **5**(14), 126 (2013)
48. Peker, M.; ŞEN, B.; Kumru, P.Y.: An efficient solving of the traveling salesman problem: the ant colony system having parameters optimized by the Taguchi method. Turk. J. Electr. Eng. Comput. Sci. **21**(Sup. 1), 2015–2036 (2013)
49. Gunduz, M.; Kiran, M.S.; Ozceylan, E.: A hierarchic approach based on swarm intelligence to solve the traveling salesman problem. Turk. J. Electr. Eng. Comput. Sci. **23**(1), 103–117 (2015)
50. Mahi, M.; Baykan, Ö.K.; Kodaz, H.: A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. Appl. Soft Comput. **30**, 484–490 (2015)

51. Escario, J.B.; Jimenez, J.F.; Giron-Sierra, J.M.: Ant colony extended: experiments on the traveling salesman problem. Expert Syst. Appl. **42**(1), 390–410 (2015)
52. Yang, J.; Wu, C.; Lee, H.P.; Liang, Y.: Solving traveling salesman problem using generalized chromosome genetic algorithm. Prog. Nat. Sci. **18**, 887–892 (2008)
53. Osaba, E.; Yang, X.S.; Diaz, F.; Lopez-Garcia, P.; Carballedo, R.: An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. Eng. Appl. Artif. Intell. **48**(1), 59–71 (2016)
54. Zhou, Y.; Luo, Q.; Chen, H.; He, A.; Wu, J.: A discrete invasive weed optimization algorithm for solving traveling salesman problem. Neurocomputing **151**, 1227–1236 (2015)
55. Toth, P.; Vigo, D. (eds.): Vehicle Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics. SIAM, Philadelphia (2014)
56. Chen, H.; Zhou, Y.; He, S.; Ouyang, X.; Guo, P.: Invasive weed optimization algorithm for solving permutation flow-shop scheduling problem. J. Comput. Theor. Nanosci. **10**(3), 708–713 (2013)
57. Snyder, L.V.; Daskin, M.S.: A random-key genetic algorithm for the generalized traveling salesman problem. Eur. J. Oper. Res. **174**(1), 38–53 (2006)
58. Ouyang, X.; Zhou, Y.; Luo, Q.; Chen, H.: A novel discrete cuckoo search algorithm for spherical traveling salesman problem. Appl. Math. Inf. Sci. **7**(2), 777 (2013)
59. Choi, I.C.; Kim, S.I.; Kim, H.S.: A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. Comput. Oper. Res. **30**(5), 773–786 (2003)
60. Cirasella, J.; Johnson, D.S.; McGeoch, L.A.; Zhang, W.: The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests. Algorithm Engineering an Experimentation, pp. 32–59. Springer, Berlin (2001)