An Improved Particle Swarm Optimization Algorithm For A Variant of TSP

Dr. Nitesh M Sureja[1], Dr. Sanjay P Patel[2]

[1]*Director, OM Engineering College, Junagadh, Junagadh Dist, Gujarat, India*
[2]*Assistant Professor, Government Engineering College, Patan, Patan Dist, Gujarat, India*

Abstract *— Particle swarm optimization algorithm is one of the nature inspired algorithms based on the flocking behaviour of a swarm of the birds. The standard Particle swarm optimization algorithm has been successfully used to solve many engineering problems. Each and every algorithm has its own merits and demerits like stagnation and fall in premature convergence in searching space. It is always necessary to handle the issues of exploitation and exploration of the search space. Excessive exploitation leads to premature convergence, while excessive exploration slows down the convergence. In this paper, an improved particle swarm optimization algorithm is proposed to solve the Random Traveling Salesman Problem. Random TSP is a type of the TSP where the TSP problems are defined randomly. The results obtained from this algorithm are compared with the results obtained with other optimization algorithms like GA, MA and ACO. Results shows that the Particle swarm optimization (PSO) algorithm performs very well to solve most of TSP problems, but it can be trapped into local optimum solutions for some of the problems.*

Keywords — *Particle Swarm Optimization, Nature Inspired Algorithms, Random Traveling Salesman, Optimization Introduction*

I. INTRODUCTION

In a Traveling salesman problem (TSP), a salesman has to travel all the cities in a tour exactly once [1][2]. He has to reach back to the starting city to complete the tour. Starting and ending city is same here. The salesman has to complete the tour with the minimum time and cost. So, he has to find the shortest path of a tour to minimize cost and time. The problem becomes a problem of finding the shortest distance path problem. TSP falls in to the category of the combinatorial optimization problem. TSP is very easy to understand but very difficult to solve. It is very difficult to find exact solution for a TSP. The complexity of the problem increases with the increase in the number of cities. Many exact and approximation methods have been proposed to address the traveling salesman problem. Exact methods starts to fail with the increase in the number of cities. So, approximation methods, known as optimization methods are used to solve the TSP.

G. Dantzig et al [3] presented a cutting plane method to solve TSP in 1954. G. laporte [4] presented a branch and bound method for solving TSP in 1992. A simulated Annealing algorithm [5] for TSP is proposed by David Bookstaber in 1997. The algorithms based on neural networks to solve TSP are presented by [6][7]. This problem is also solved by [8][9][10] using the concept of biological evolution (Genetic Algorithms). Memetic algorithm is developed by [11] [12] to solve the TSP. The foraging searching behavior of ant is modeled by [13][14] to address the TSP problem. Bee colony approaches are presented by [15][16][17] for the same. A monkey search algorithm to solve TSP is presented by [18]. A discrete cuckoo search algorithm is proposed by Jati GK et al for TSP in 2012[19]. Algorithms based on the flashing behavior of firefly are presented by [20][21]. A Bat algorithm is proposed by [22] for solving TSP. Artificial immune algorithms to solve TSP are proposed by [23][24][25]. Algorithms based on the flocking behaviors of birds for solving traveling salesman problem are proposed by [26][27][28]. Particle Swarm Optimization (PSO) algorithm is developed by Kennedy and Eberhart in 1995[29]. This is a stochastic optimization algorithm which simulates the foraging behaviors of a swarm of birds. In this algorithm, every solution is considered as a particle, and the combination of particles makes the entire swarm. The concepts of velocity and position are used by the PSO to find the optimal point(s) in the search space.

Mainly PSO algorithm is characterized in two steps namely initialization step and cycle step. The cycle step is divided into four steps :(1) fitness evaluation (2) finding p^{best} and g^{best} ,(3) new value generation of velocity and position and (4) inertia weight Updating.

In this paper, an algorithm to solve Random Traveling Salesman Problem (RTSP) using Particle Swarm Optimization is proposed. The paper is organized as follows.

1. Section I: Introduction
2. Section II: Random Traveling Salesman(RTSP)
3. Section III:PSO Algorithm Introduction
4. Section IV: Proposed PSO-RTSP Algorithm
5. Section V: Implementation and Results of Proposed PSO-RTSP
6. Section VI: Conclusion

Finally, paper ends with the references.

II. RANDOM TRAVELING SALESMAN PROBLEM

There are many types of traveling salesman problem (TSP) described in the literature based on their characteristics. To list a few:

1. Symmetric TSP
2. Asymmetric TSP
3. Dynamic TSP
4. Random TSP.
5. Spherical TSP

and many more.

Symmetric TSP is a tsp where distance between the cities is same from the either side. Asymmetric TSP is a TSP where distance between the cities from the either side is not same. Dynamic TSP is a TSP where the problem changes itself at run time. In a spherical TSP all cities lie on a sphere. Random TSP is a TSP where all the city problems are defined randomly before starting to find its solution. This is done to remove the problem of local optima during the solution finding. A random instance generator is used to generate the city problems randomly. This paper presents an algorithm to solve Random TSP. Particle Swarm Optimization algorithm is used here to solve the Random TSP. all city problems are generated in the range of 10 to 100.

III. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is an intelligent optimization algorithm based on the Swarm Intelligence. It is based on a simple mathematical model, developed by Kennedy and Eberhart in 1995, to describe the social behavior of birds and fish [29]. The model relies mostly on the basic principles of self-organization which is used to describe the dynamics of complex systems. Subsequently PSO was applied to solve many continuous and discrete optimization problems by various authors. PSO is swarm intelligence based meta-heuristic which is inspired by the group behavior of bird flocking or fish schooling. Similarly to genetic algorithms (GAs), it is a population-based method, that is, it represents the state of the algorithm by a population, which is iteratively modified until a termination criterion is satisfied. In PSO algorithms, the population of the feasible solutions is often called a swarm. The feasible solutions are called particles. The PSO method views the set of feasible solutions as a "space" where the particles "move".

Unlike GAs, PSOs do not change the population from generation to generation, but keep the same population, iteratively updating the positions of the members of the population (i.e., particles). PSOs have no operators of "mutation", "recombination", and no notion of the "survival of the fittest". On the other hand, similarly to GAs, an important element of PSOs is that the members of the population "interact" or "influence" each other.

Each particle *i* has its neighbourhood *Ni* (a subset of *P*). The structure of the neighbourhoods is called the swarm topology, which can be represented by a graph.

A. Characteristics of particle i at iteration t

- $x_i^{(t)}$ … the position (a d-dimensional vector)
- $p_i^{(t)}$ … the "historically" best position
- $l_i^{(t)}$ … the historically best position of the neighbouring particles; for the fully connected topology it is the historically best known position of the entire swarm
- $v_i^{(t)}$ … the speed; it is the step size between $x_i^{(t)}$ and $x_i^{(t+1)}$
- At the beginning of the algorithm, the particle positions are randomly initialized, and the velocities are set to 0, or to small random values.

B. Algorithm Parameters

- $w^{(t)}$ … inertia weight; a damping factor, usually decreasing from around 0.9 to around 0.4 during the computation
- φ_1, φ_2 … acceleration coefficients; usually between 0 and 4.

C. Update Velocity and positions of the particles

Particle Velocity update takes place using equation 1.

$$v_i^{(t+1)} = w^{(t)} vi^{(t)} + \varphi_1 u_1 (p_i^{(t)} - x_i^{(t)}) + \varphi_2 u_2 (l_i^{(t)} - x_i^{(t)}) \quad (1)$$

The symbols u_1 and u_2 represent random variables with the $U(0,1)$ distribution. The first part of the velocity formula is called "inertia", the second one "the cognitive (personal) component", the third one is "the social (neighborhood) component". Position of particle i changes according to equation 2.

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \quad (2)$$

The goal of the algorithm is to have all the particles locate the optima in a multi-dimensional hyper-volume. This is achieved by assigning initially random positions to all particles in the space and small initial random velocities. The algorithm is executed like a simulation, advancing the position of each particle in turn based on its velocity, the best known global position in the problem space and the best position known to a particle. The objective function is sampled after each position update. Over time, through a combination of exploration and exploitation of known good positions in the search space, the particles cluster or converge together around optima, or several optima.

The algorithm is terminated after a given number of iterations, or once the fitness values of the particles (or

the particles themselves) are close enough in some sense.

There is a plethora of different versions of PSOs, which usually modify the formula for the change of velocity (e.g., instead of u1 and u2 they use diagonal matrices U1 and U2, in other variants they use no inertia, but enforce an upper limit on the particle speed, there is the so-called "fully informed" PSO, and there is also a popular modification using a "constriction coefficient".

There exist versions of the PSO for constrained optimization, for discrete optimization, and for multi-objective optimization.Pseudo code of the Particle Swarm Optimization algorithm is given below.

For each particle
  Initialize particle
End
Do
  For each particle
    Calculate fitness value
    If the fitness value is better than the best fitness value(p^{best}) in history
    Set current value as the new P^{best}
  End
  Choose the particle with the best fitness value of all the particles as the g^{best}
  For each particle
    Calculate velocity according equation (1)
    Update particle position according equation (2)
  End
While maximum iterations or minimum error criteria is not attained

Fig. 1 Pseudocode of Particle Swarm Optimization

IV. PROPOSED PSO-RTSP MODEL

Now, basic PSO can't be applied directly to the TSP as TSP is a discrete optimization problem. The equation 1 and 2 are slightly modified here. We introduce the concept of probability here. In the TSP, next city will be chosen based on the probability (X) and a random number (R) defined. The value of random number is kept in between 0 and 1. During each updation, if R <= X then the corresponding edge is selected for further operation.

A new Parameter u_3 is also used here to avoid the local optima or to explore the search space very efficiently. So, the formula 2 is converted into formula 4 here and the new updated equation looks like as follows,

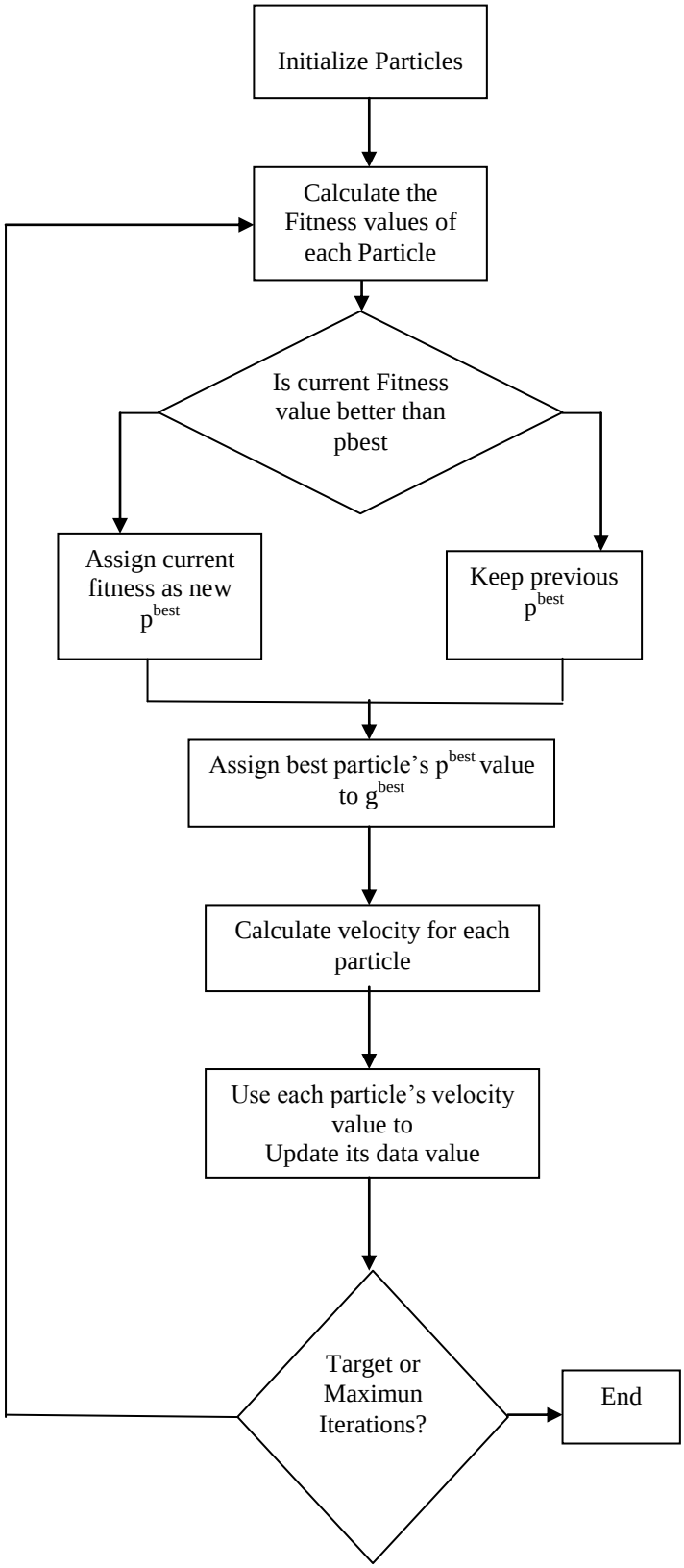

Fig. 2 Flowchart of the Proposed PSO-RTSP Model

(1)

$$v_i^{(t+1)} = w^{(t)} vi^{(t)} + \varphi_1 u_1 (p_i^{(t)} - x_i^{(t)}) + \varphi_2 u_2 (l_i^{(t)} - x_i^{(t)})$$

$$x_i^{(t+1)} = v_i^{(t+1)} \square \varphi_3 u_3 * x_i^{(t)} \quad (2)$$

Fitness function is the only standard of judging whether an individual is "good" or not. We take the reciprocal of the length of each path as the fitness function. Length the shorter, fitness value the better. Flowchart of the proposed PSO-RTSP model is given in the Figure 2.

V. IMPLEMENTATION AND RESULTS

Proposed PSO-RTSP model is implemented in Matlab -12. Program developed is run on a dual core machine with four GB RAM. As mentioned, all tsp problems are generated randomly in the range of 10 to 100. Algorithm runs itself based on the termination criteria. Termination criteria used here is number of iterations. Table 1 shows the results obtained by the algorithm. Results are also shown graphically in the figure 3 to 12.

TABLE I. RESULTS OBTAINED FROM PROPOSED PSO-RTSP MODEL

City Problem	PSO-RTSP		
	Length	*Time*	*Iterations*
10	**281.6978**	**16.9741**	**200 (Termination Criteria)**
20	**406.4415**	**19.0053**	
30	**452.2936**	**22.0370**	
40	**505.4753**	**25.8914**	
50	**609.8611**	**30.9500**	
60	**720.5503**	**38.9281**	
70	**785.1615**	**43.9166**	
80	**902.1146**	**51.5915**	
90	**943.2347**	**60.2499**	
100	**980.1618**	**68.5272**	

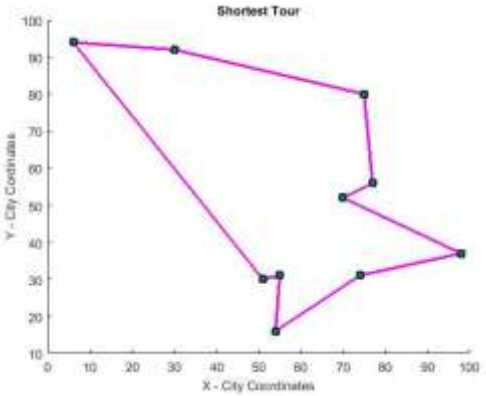

Fig. 3 10 City Problem Results

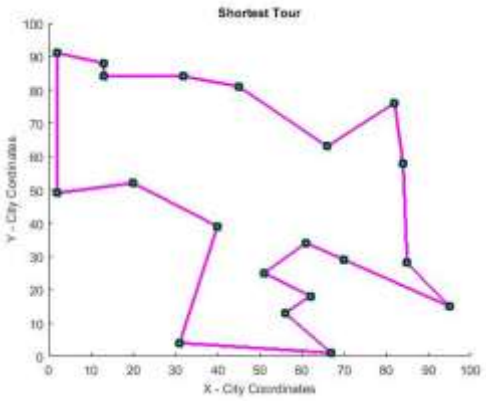

Fig. 4 20 City Problem Results

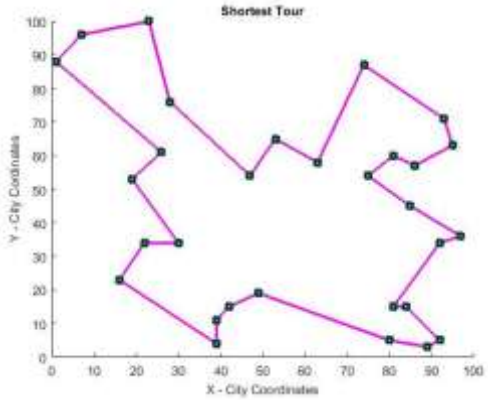

Fig. 5 30 City Problem Results

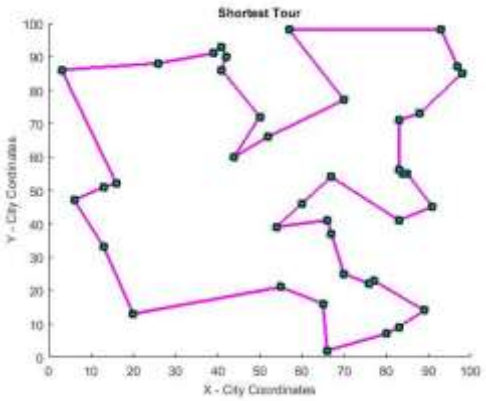

Fig. 6 40 City Problem Results

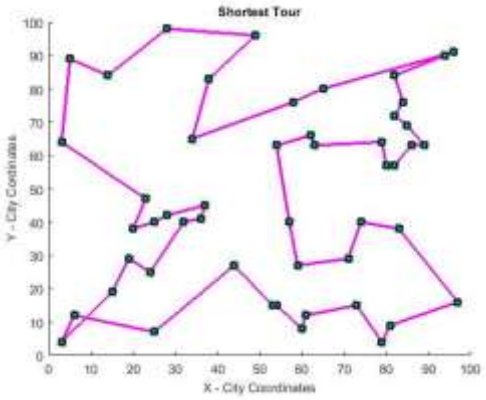

Fig. 7 50 City Problem Results

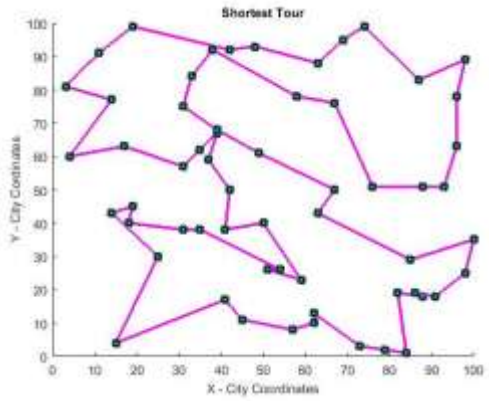

Fig. 8 60 City Problem Results

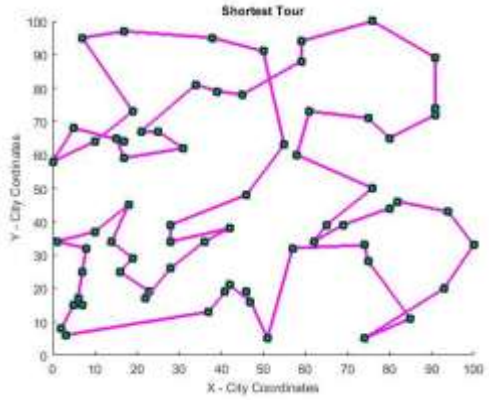

Fig. 9 70 City Problem Results

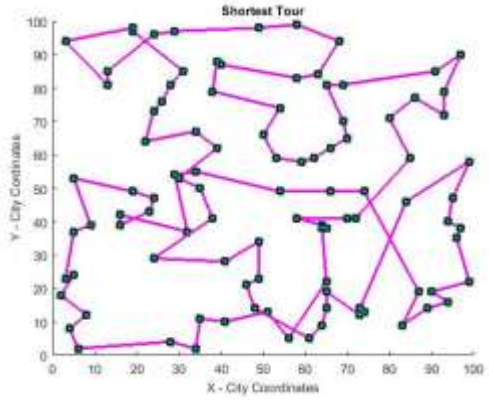

Fig. 10 80 City Problem Results

VI. CONCLUSION

A PSO algorithm based on the flocking behaviors of the birds has been introduced to solve the TSP. The model has been tested on a set of randomly generated TSP problems. As this is the initial implementation of PSO on Random TSP, we hope to improve the model further to achieve optimal values for the list of randomly generated TSP problems. This algorithm generates very good results in terms of time as well as distance for the small to average size of the TSP problems. The algorithm starts to detoriate in terms of quality as the size of the problem increases. It is observed that with the some parameter settings the quality can be improved for the big size problems also.

ACKNOWLEDGMENT

The authors would like to thank OM Engineering college-Junagadh, Government Engineering college-Patan and all those who have supported directly or indirectly to carry out this research.

REFERENCES

[1] M. M. Flood, ―*The Traveling Salesman Problem, Operations Research*, 1956

[2] Gerhard Reinelt. "*The Traveling Salesman: Computational Solutions for TSP Applications.*", Springer-Verlag, (1994),Berlin, Heidelberg .

[3] G. Dantzig, R. Fulkerson, S. Johnson, *Solution of a Large-Scale Traveling Salesman Problem*, J. Oper. Res. Soc. 2 (1954) 393–410.

[4] G. Laporte, *The vehicle routing problem: an overview of exact and approximate algorithms*, Eur. J. Oper. Res. 59 (1992) 345–358.

[5] D. Bookstaber, "*Simulated Annealing for Traveling Salesman Problem*", Spring, 1997.

[6] S. M. Abdel-Moetty, "*Traveling salesman problem using neural network techniques*", IEEE The 7th International Conference on Informatics and Systems (INFOS), 2010, Cairo, Egypt.

[7] Budinich, M.: *A Self-Organizing Neural Network for the Traveling Salesman Problem That Is Competitive with Simulated Annealing*. Neural Computing 8, 416–424 (1996).

[8] Abid Hussain, Yousaf Shad Muhammad, M. Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry, and Showkat Gani, "*Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator*," Computational Intelligence and Neuroscience, vol. 2017, Article ID 7430125, 7 pages.

[9] Yong Deng, Yang Liu, and Deyun Zhou, *"An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP*," Mathematical Problems in Engineering, vol. 2015, Article ID 212794, 6 pages, 2015.

[10] Adewole, Philip & Taofiki, Akinwale & Otunbanowo, Kehinde. (2011). *A Genetic Algorithm for Solving Travelling Salesman Problem.* International Journal of Advanced Computer Sciences and Applications. 2. 10.14569/IJACSA.2011.020104.

[11] Gutin, G., & Karapetyan, D. (2009). *A memetic algorithm for the generalized traveling salesman problem*. Natural Computing, 9(1), 47–60.

[12] Arango Serna, M. D., & Serna Uran, C. A. (2015). *A Memetic Algorithm for the Traveling Salesman Problem.* IEEE Latin America Transactions, 13(8), 2674–2679.

[13] M. Dorigo, T. Stutzle, "*Ant Colony optimization*", A Bradford book, MIT Press Cambridge, Massachucetts,london, England (2004) .

[14] M. Dorigo, L. Gambardella, "*Ant colonies for the Traveling salesman problem*." Biosystems, 43 (1997): (73-81).

[15] Jiang, H., "*Solving Traveling Salesman Problem Using Artificial Bee Colony Algorithm*.", Computer Science and Technology. (2016) : (989-995)

[16] Li, L., Cheng, Y., Tan, L., & Niu, B. (2012). *A Discrete Artificial Bee Colony Algorithm for TSP Problem*. Lecture Notes in Computer Science, 566–573

[17] Wong, L.-P., Low, M. Y. H., & Chong, C. S. (2008). *A Bee Colony Optimization Algorithm for Traveling Salesman Problem.* 2008 Second Asia International Conference on Modelling & Simulation (AMS). doi:10.1109/ams.2008.27

[18] Ayon, S. I., Akhand, M. A. H., Shahriyar, S. A., & Siddique, N. (2019). "*Spider Monkey Optimization to Solve Traveling Salesman Problem*.", 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE).