Spider Monkey Optimization to Solve Traveling Salesman Problem

Safial Islam Ayon[1], M. A. H. Akhand[1], S. A. Shahriyar[1], and N. Siddique[2]
[1] Dept. of CSE, Khulna University of Engineering & Technology, Khulna, Bangladesh
[2] School of Computing, Engineering and Intelligent Systems, Ulster University, United Kingdom

***Abstract*—Traveling Salesman Problem (TSP) is the most well-known combinatorial optimization real-world problem. TSP is also very popular to check proficiency in any newly developed optimization method. In addition, the optimization methods, which are developed for other tasks (e. g., numerical optimization), also test their proficiency in TSP. This study investigates a new technique to solve TSP based on a recently developed optimization technique stimulated through the foraging conduct of spider monkeys. Standard Spider Monkey Optimization (SMO) is established for numerical optimization which has six phases and each one has a different purpose. In this study, SMO is modified and updated to solve TSP; and Swap Operators (SOs), and Swap Sequence (SS) are considered to adapt SMO for TSP. In the proposed method, each spider monkey is considered as a TSP solution and SS is considered to update the solution. An SS is a group of SOs and everyone shows two positions in a visit which may be swapped. All SOs of a SS is applied on a specific tour maintaining order and thus ramifications of the SS change the TSP tour into another one. The SOs are generated using the experience of a specific spider monkey as well as the experience of other members (local leader, global leader, or randomly selected spider monkey) of the group. The proposed strategy has been examined on a huge number of benchmark TSPs and final consequences are compared to other prominent methods. Experimental consequences show that the proposed strategy is a decent technique to resolve TSP.**

Index Terms— Traveling Salesman problem, Swap Sequence, Swap Operator, Spider Monkey Optimization

I. INTRODUCTION

Traveling Salesman Problem (TSP) is the most well-known combinatorial optimization real-life problem. TSP is the problem faced by a salesperson who begins from a specific city and travels all of the different cities in the shortest conceivable path. The salesman cannot visit any city twice before returning to the starting city [1]. TSP demonstrates all the parts of combinatorial optimization and comes under the set of NP-hard. If one can easily find a proposed solution for a specific problem with polynomial time then it is called *NP* problem. And if it as difficult as any NP problem then it is called *NP*-hard problem [2]. There are numerous real-life applications of TSP such as vehicle routing, computer wiring, X-Ray crystallography, printed circuit panel, order selecting problem in warehouses [1].

TSP is also very popular to check proficiency of optimization methods. Nearly every year new method for resolving engineering and optimization problems has been verified on the TSP as a standard test bench and interest has been grown-up in recent times to resolve it in modern methods. Ant Colony Optimization (ACO) [3] is the pioneer one to resolve TSP and is shown to perform well. In addition, the optimization methods which are developed for other optimization tasks (e.g., numerical optimization) are also modified to solve TSP [4, 10, 11]. A number of methods are also developed based on Particle Swarm Optimization (PSO) [4, 10, 11]. Among the methods, Swap Sequence (SS) based PSO methods are shown to perform well to solve TSP.

This paper explores a new method to resolve TSP, based on a currently evolved optimization approach using Spider Monkey Optimization (SMO) [6]. In recent years, researchers develop the SMO algorithm in many ways. Several modifications are different perturbation rate, different probability scheme, local search ability, modified position update strategy, fitness-based position, and etc. [8]. There are different types of applications of SMO like automatic generation control, CDMA multiuser detection, wire sensors networks, image segmentation, numerical classification, antenna optimization, optical power flow, and etc. [8]. In this study, SMO is modified and updated to solve TSP; and Swap Operators (SOs) and SS are considered to adapt SMO for TSP.

SMO is developed stimulated through the foraging conduct of spider monkeys. The social conduct of spider monkeys is an instance of a Fission-Fusion Social Structure (FFSS) [7]. The spider monkey lives in a large group and shows intelligence foraging behavior. The foraging conduct of the spider monkey is divided into four stages. In the first stage, they start looking for food and estimate their distance from the food. In the second stage, they update their position according to the distance from the food. In the third stage the local-leader modifies its superior position in his group and if the position is not modified at a predetermined time, then the group members start searching for food in different directions. At the final stage, the global-leader also modifies its best position and when the number of groups reached a maximum number, then the main group leader combines all the subgroups into one large group or else it splits the group into some subgroups. Their intelligence technique for searching food is the source of motivation to develop the algorithm. All the members search for food and interconnected with each other via voice messaging.

Standard SMO is developed for numerical optimization [6], and the components of SMO are updated and hence a new method is proposed to solve TSP in this study. In the proposed method, each spider monkey is considered as a TSP result and SS is applied to update the solution. An SS is a group of SOs and everyone indicates two positions in a visit which may be swapped. All SOs of an SS is applied on a particular tour keeping order and consequently, ramifications of the SS change the TSP tour into another one. The SOs are generated using the experience of a specific spider monkey as well as the experience of other members (local leader, global leader, or spider monkey selected by randomly in the particular group) of the group.

Rest of the paper has been organized as follows. Section II describes the proposed method to solve TSP using SMO algorithm. Section III presents the experimental outcomes of the proposed strategy and compares the performance with

978-1-5386-9111-3/19/$31.00 ©2019 IEEE

other methods. Finally, Section IV presents a brief conclusion of the paper.

II. Spider Monkey Optimization (SMO) to Solve TSP

This section explains the proposed SMO based method for solving TSP. Algorithm 1 displays the phases of the standard SMO algorithm for numerical optimization. There are six phases in SMO and in the algorithm, each stage has a different purpose to complete the algorithm. local-leader-phase and global-leader-phase are responsible for producing a newly updated position for every spider monkey in the group. If the position is better than the previous position, the position is updated to the new position or, it retains the existing solution. global-leader and local-leader are selected in the global-leader-learning-phase, and local-leader-learning-phase respectively. In the local-leader-decision-phase and global-leader-decision-phase, local-leader initializes all the group's members again and the global-leader takes the decision to divide the group into subgroups or join all the groups into one group. This study, SMO operators are modified to resolve

Algorithm 1: SMO Algorithm

Step 1: Initialization
Step 2: Select **Local-Leader** and **Global-Leader**
Step 3: For each Spider Monkey
a. Generate a new position
b. Calculate the probability
c. Based on the probability update position again
d. Select new **Local-Leader** and **Global-Leader**
e. Redirect all the members of the group if Local-Leader is not updating for a specific time.
f. Splits the group into subgroups or combine all the groups into one group if Global-Leader is not updating for a specific time.
Step 4: Step 3 continue until a termination criterion is met
Step 5: G**lobal supreme** will be the final outcome

TSP. In the proposed system, SS is used to update a solution. The following subsections explain the operators in detail to make the paper independent.

A. Swap Operator in TSP

A SO [4, 9-10] is a couple of indexes that imply two cities which may be swapped in a tour solution. Assume a solution in TSP has four cities in a sequence as defined by S = (4 – 1 – 3 – 2) is a solution and a *SO* is *SO* (1,3), then the modified solution S' is defined by:

$$S' = (4 – 1 – 3 – 2) + SO\ (1,3) = (3 – 1 – 4 – 2)$$

Here '+' means applying the swap sequence to the solution.

An SS [4, 9-10] is a set of one or additional SO(s) that applied to a specific solution in a sequence, which is expressed by

$$SS = (SO_1, SO_2, SO_3, \ldots, SO_n)$$

Here SO_1, SO_2, SO_3, …, SO_n are *SOs*. The *SS* can also be applied to solutions, e.g. S_1 and S_2, in the following from way

$$SS = S_2 – S_1 = (SO_1, SO_2, SO_3, \ldots, SO_n)$$

Here '-' means want to apply SO_s of *SS* to solution S_1 to S_2. Suppose two solutions are S_1 = (1 – 2 – 3 – 4 – 5) and S_2 = (2 – 3 – 1 – 5 – 4) then *SS* = *SO* (1,2), *SO* (2,3), *SO* (4,5).

It is to be noticed that various *SSs* applied to the same solution may generate the same result. All these SSs are called the equivalent set of *SSs*. Among all the *SSs*, the *SS* which has the minimum *SOs* is called the Basic Swap Sequence (BSS).

Suppose two *SS* are: SS_1 = *SO* (2,3), *SO* (3,2), *SO* (4,1), *SO* (5,3), *SO* (3,4) and SS_2 = *SO* (4,1), *SO* (5,3), *SO* (3,4) they are applied to S_1 = (e-a-b-c-d) independently, giving the outcome S_2 = (c-a-e-d-b). Consequently, SS_2 is the BSS. Using $S_2 – S_1$ it will also be found.

B. Modified SMO for TSP

There are six phases in SMO and all are modified to solve TSP. The modified phases are described in the following sections. However, at first initialization is explained and the complete algorithm is presented finally.

1) Initialization

First, initialize N random solutions. Then divide the population of the spider monkey into n groups. In every group, select the local leader. Among all the group members, select the global leader. There are four manipulate parameters in the SMO algorithm: Local-Leader-Limit, Global-Leader-Limit, Maximum-Group (*MG*) size, and Perturbation-Rate (*pr*), which are initialized at the start of the algorithm.

2) Local-Leader-Phase (LLP)

In the LLP, each spider monkey modifies its solution based on its current involvement, local leader involvement and the involvement of randomly selected group members of that group. The solution update equation for a spider monkey in this phase is

$$SS = U[0\text{-}1]*(LLk – SMi) + U[0\text{-}1]*(RSMr – SMi) \quad (1)$$

$$BSS = BasicSwapSequence\ (SS) \quad (2)$$

$$SMnewi = SMi + BSS \quad (3)$$

At first, it calculates the SS between the LL_k and SM_i, and *SS* of randomly selected spider Monkey (RSM_j) and SM_i. Use a random portion of the *SS* using *U[0-1]*. Then add those two *SSs* into one *SS* using Eq. (1). Using Eq. (2) calculate the BSS of the generated *SS*. The BSS is then applied to the solution of the SM_i as seen in Eq. (3). If the cost of the new solution is less than the previous solution's cost, then the previous solution is replaced by the newly generated solution. Otherwise, the previous solution is retained.

3) Global-Leader-Phase (GLP)

In the GLP, all the spider monkeys modify their solution using its current involvement, global-leader involvement and randomly selected local group member involvement. But in this phase, an SM decides based on the value of probability if the SM will update or not. Eq. (4) shows the formula for calculating the probability of *ith SM*.

$$prob(i) = 0.9 * \frac{min\_cost}{cost(i)} + 0.1 \quad (4)$$

In Eq. (4), *cost(i)* is the tour cost of ith *SM* and *min_cost* is the minimum tour cost by the best SM of that group. The solutions are updated in this phase according to

$$SS = U[0,1]*(GL - SM_i) + U[0,1]*(RSM_r - SM_i) \quad (5)$$

$$BSS = BasicSwapSequence\ (SS) \quad (6)$$

$$SMnew_{ij} = SM_{ij} + BSS \quad (7)$$

Other procedures like find *SS*, merge them together using Eq. (5), calculate the BSS using Eq. (6) and apply the BSS to the solution using Eq. (7) are same as Local Leader phase. Here also, the newly generated tour cost and the current tour cost are compared and finally select the best one.

4) Local-Leader-Learning (LLL) Phase

This phase compares the minimum tour cost and tour cost of a local leader in every group. If the new tour cost is less than the local leader tour cost, then the new solution is elected as the updated local-leader of the group. Otherwise, the local-leader retains its current solution. If the local-leader is not modified, then Local-Limit-Count is augmentation by 1.

5) Global-Leader-Learning (GLL) Phase

The minimum tour of all the SM is found in this GLL phase. Then it is compared with the present global-leader tour cost. If the tour cost is less than the current cost, then the new solution is selected as the updated global-leader. Further, if the cost of global-leader is not updated then GlobalLimitCount is augmentation by 1.

6) Local-Leader-Decision (LLD) phase

If the local-leader position is not modified for a certain period called Local-Leader-Limit then all the members of that group modify their position using either random initialization or combining the involvement of global-leader and local-leader using Eq. (8).

$$SMnew_i = SM_i + U[0,1]*(GL - SM_i) + U[0,1]*(SM_i - LL_k) \quad (8)$$

Here, *GL* is the global leader and LL_k is the local leader of the *kth* group. Following the Eq. (8), *SMi* moves forward to the global leader and detaches from the local leader.

7) Global-Leader-Decision (GLD) phase

If the position of global leader is not modified for a certain period of time known as Global-Leader-Limit, then the global-leader splits the population into small groups until *MG* is reached. Each time *GLD* phase splits the group, *LLL* phase selects the new local leader. When the highest number of groups are created and the global-leader is not modified, then the global-leader joins all the groups into one group.

8) SMO algorithm for TSP

Algorithm 2 displays the phases of the modified SMO for TSP solve. Like other population-based algorithms, SMO initializes the solution randomly. At first, initially selects the LL_k of every group and *GL* (Step 2). At each iteration step, every SM_i will update its solution using *SS* (Step 3.a and 3.c). *LL* and *GL* will update in every iteration if the newly generated solution is better than the current solution (Step 3.d). LL_k will initialize all the SM_i if the LL_k is not updated for a predetermined time (Step 3.e). *GL* will split the population into some small sets or combine all the groups into one single set if the *GL* is not updating a specific time (Step 3.e). The iteration will continue until a termination criterion will be met. Finally, *GL* is considered a final outcome (Step 5).

Algorithm 2: SMO Algorithm for solve TSP

Step 1: Initialization

Step 2: Select *GL* and LL_k of the k^{th} group

Step 3: For each solution SM_i in the group

a. Calculate *SS* using Eq. (1), BSS using Eq. (2), and modify SM_i using Eq. (3)
b. Compute the *prob(i)* using Eq. (4)
c. Based on the *prob(i)* calculate *SS* using Eq. (5), BSS using Eq. (6), and update SM_i using Eq. (7)
d. Update LL_k and *GL*
e. Initialize all the members either randomly or using Eq. (8)
f. Divide the population or combine all groups

Step 4: If (termination condition is met), Goto **Step 3**

Step 5: The global Supreme Leader *GL* as a solution

III. Experimental Studies

The outcomes of the proposed method to solve TSP using SMO are compared with two prominent methods ACO and Velocity Tentative PSO (VTPSO) are discussed in this section. ACO was the pioneer and well-studied technique for TSP, and VTPSO was the recently developed best-suited method for TSP.

A. Benchmark TSP Data and Experimental Methodology

In this examination, a set of 15 benchmark problems are considered from TSPLIB [12] where the number of cities is between 14 to 101. The experiment was implemented on MATLAB R2017a. For appropriate understanding, tests have been led on a desktop computer (HP ProBook 450G, Intel® Core ™ i5-4200M, CPU @ 3.60 GHz, RAM 8 GB) with Windows10. For a reasonable examination, the investigational methodology was selected deliberately.

B. Experimental Analyses varying Population and Total Iteration

In this section, the experimental analyses of SMO are compared to ACO and VTPSO for a selected problem Eli51. In analyses, variation impact of population size and total iteration were presented. The outcomes were the average of 10 separate runs.

Figure 1 demonstrates the tour cost for various population sizes. Here, the population size varies from 5 to 500. For ACO, the number of ants was equivalent to the number of cities (i.e., 51) as it was desired. For all the problems, the total number of iterations was fixed at 500. It was observable from Fig. 1 that all the approaches give poorest tour costs at small population size (e.g. 10, 20). Since population size was permanent in ACO for a specific problem, it demonstrated invariant performance in the figure and the performance was worst among the three methods. On the other hand, the results have improved with the increase in population size in case of VTPSO and SMO. It was noticeable from the figure that SMO was better than in general.

Figure 2 demonstrates the accomplished tour costs of ACO, VTPSO, and SMO for different iterations. Here the iteration number varied from 20 to 1000 and the population size was permanent at 100 for all methods. It was seen from the figure that ACO was invariant to iteration variation. On the other hand, tour costs were shown very high for a small

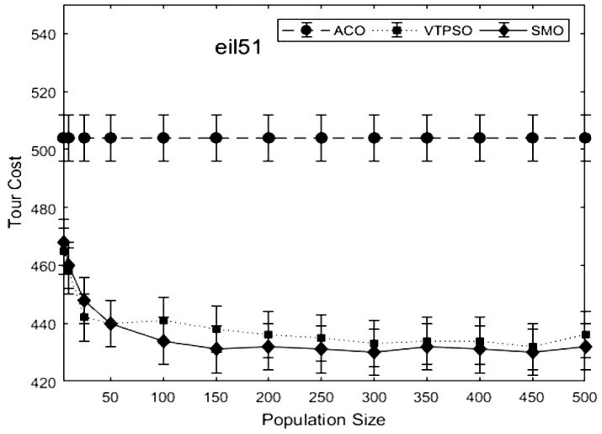

Figure 1. Different impact of population size on tour cost.

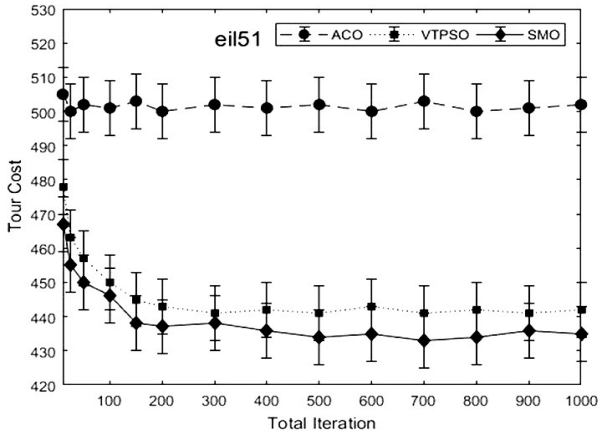

Figure 2. Different impact of total iteration on tour cost.

number of iterations (e.g., 20, 50) and improved with the increase of iteration for both SMO and VTPSO. Although both SMO and VTPSO were found competitive for a small number of iterations, SMO outperforms VTPSO in general. It was also notable that after that certain value, the changing result was not significant. At a look, proposed SMO has demonstrated the capacity to accomplish better result in fluctuating population size and iteration number.

C. Compare experimental results with ACO and VTPSO

This section compares the experimental results of SMO to ACO and VTPSO. For comparison, the iteration number was fixed at 500 and the population size was fixed at 100 for all the problems. As it was seen from Fig. 1 and Fig. 2 that when the population size was greater than 100 and the iteration number was greater than 500, the result did not change significantly.

Table I compares the performance of ACO, VTPSO, and SMO for 15 different benchmark TSPs. All the methods were run ten separate times for every problem. In Table I, the best tour cost (smallest value) among the 3 techniques was shown in bold-face type and the worst tour cost (biggest value) was indicated by underlined face type. A summary of the comparison was shown at the bottom of the table.

In Table I, the average tour cost for all the problems demonstrate that SMO was the best one and ACO was the poorest among the three methods. The average tour cost of SMO was 6354.45 for all 15 problems. The accomplished average tour cost of ACO and VTPSO was 7034.07 and 6466.04, respectively. In Best/Worst count for individual problems, VTPSO and SMO did not show any worst result. SMO showed the best results in eleven problems out of 15 problems and VTPSO showed the best result for six cases. In

TABLE I. COMPARISON BETWEEN THE PROPOSED SMO WITH ACO AND VTPSO TO RESOLVE BENCHMARK TSPS

SL	Problems	Average Tour Cost (Standard Deviation)			t-test eval. of SMO		Minimum Tour Cost		
		ACO	VTPSO	SMO	ACO	VTPSO	ACO	VTPSO	SMO
1	Burma14	31.32 (0.66)	**30.87** (0.0)	**30.87** (0.0)	++		31.21	**30.87**	**30.87**
2	Ulysses16	77.13 (0.0)	**74.02** (0.08)	74.04 (0.19)	++		77.13	**73.99**	**73.99**
3	Ulysses22	86.19 (0.36)	**75.41** (0.24)	**75.41** (0.31)	++		86.08	**75.31**	**75.31**
4	Bays29	9964.68 (0.0)	9120.20 (50.94)	**9078.05** (6.00)	++	++	9964.78	**9074.15**	**9074.15**
5	Eil51	503.41 (20.76)	442.33 (4.09)	**433.64** (3.63)	++	++	461.42	**428.98**	431.43
6	Berlin52	8061.64 (44.00)	7865.36 (114.35)	**7556.91** (9.35)	++	++	7870.45	**7544.37**	7545.14
7	St70	745.34 (5.38)	715.25 (18.28)	**689.11** (2.37)	++	++	734.19	**682.57**	686.12
8	Eil76	596.69 (7.93)	565.82 (5.13)	**559.48** (5.46)	++	++	583.28	560.44	**550.14**
9	Gr96	589.65 (2.92)	**537.10** (12.14)	539.76 (9.43)	++	-	564.37	**521.25**	526.25
10	Rat99	1369.47 (4.14)	1311.55 (37.87)	**1268.79** (7.37)	++	++	1366.3	1254.54	**1252.64**
11	Kroa100	24667.6 (98.00)	22138.5 (492.27)	**21593.2** (221.46)	++	++	24524.53	**21438.19**	21486.69
12	Krob100	25354.4 (268.57)	23228.3 (475.35)	**22864.0** (321.87)	++	++	24675.03	22648.29	**22499.02**
13	Kroc100	23300.1 (58.98)	**21830.6** (604.28)	21837.8 (547.08)	++	- -	23248.13	**21120.65**	21123.25
14	Rd100	9423.73 (160.42)	8382.91 (199.57)	**8042.39** (86.47)	++	++	9210.67	**7944.32**	7959.45
15	Eil101	739.53 (4.03)	**672.22** (2.43)	673.11 (5.94)	++	-	729.95	**663.28**	**663.28**
	Average	**7034.07**	**6466.04**	**6354.45**			**6941.83**	**6270.75**	**6265.18**
	Best/Worst	**0/15**	**6/0**	**11/0**	**15/0**	**9/3**	**0/15**	**12/0**	**8/0**

Technique	Pairwise Victory-Draw-Defeat Summary on Average Tour Cost and Minimum Tour Cost							
	ACO	VTPSO	SMO			ACO	VTPSO	SMO
ACO	-	15-0-0	15-0-0			-	15-0-0	15-0-0
VTPSO		-	9-2-4				-	3-5-7

pair Victory-Draw-Defeat comparison, SMO gives a better result than ACO in all the 15 cases and VTPSO for nine cases. On the other hand, SMO was worse than VTPSO for only four cases.

Pair two-tailed t-test result was directed to determine the meaning of a variation in results of SMO with VTPSO and ACO. For a particular problem, if the t-test gives a significantly better result than ACO/VTPSO, then it was indicated by a + (plus) symbol in the t-test evaluation column. Else if the t-test gives significantly worst result than ACO or VTPSO for a specific problem, then it was indicated by a - (minus) sign. If the mean tour cost difference for a specific problem was statistically significant with 95% confidence interval, then it was indicated by single plus/minus and a double plus/minus for 99% confidence interval. It was observed from Table I that SMO was significantly better than ACO for all 15 cases. SMO was significantly better than VTPSO for nine cases and worse for three cases.

The best result from the different runs may use the outcome of a method. Therefore, the minimum tour cost achieved from 10 runs were also compared in Table I for better understanding. On the basis of the minimum tour costs, SMO showed a better result than ACO and VTPSO. SMO demonstrated that the average minimum tour cost of the 15 problems (i.e. 6265.18) was lowest among the three methods. The average lowest tour costs of VTPSO and ACO were 6270.75 and 6941.83, respectively. Based on the Best/Worst summary, SMO achieves the best result for 8 cases, but none of the cases shown the worst result. On the other hand, VTPSO shows 12 best results. Furthermore, according to the Victory-Draw-Defeat summary, ACO was worse than SMO in all 15 cases. Also, SMO was found better than VTPSO for three cases but SMO was inferior for seven cases.

IV. Conclusions

A modified Spider Monkey Optimization (SMO) is proposed to solve TSP in this paper. Standard SMO is a function optimization algorithm, the algorithm is modified in such a way that it can solve TSP which is a popular combinatorial problem. In SMO, individual spider monkey holds a solution and tries to update it in every iteration. Swap Sequence (SS) related procedure is developed and applied to improve TSP solutions of individual monkeys. The proposed SMO outperformed ACO and VTPSO (the prominent TSP methods) while tested in solving a suite of a huge number of benchmark TSPs. Possible future investigation directions are also opened by this study. All the Swap Operators (SOs) of an SS are applied to a solution at a time at this study. There might be a chance to get better result considering solutions applying individual SOs.

References

[1] R. Matai, S.P. Singh, and M.L. Mittal, "Traveling Salesman Problem: An overview of Application, Formulations, and Solution Approaches," Traveling Salesman Problem, Theory and Applications, D. Davendra, InTech, pp. 1-24, 2010.

[2] K.L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling Salesman Problem," Encyclopedia of Operations Research and Management Science, pp. 1573-1578, 2013.

[3] Z. Chi, S.S. Hlaing, and M.A. Khine, "Solving Traveling Salesman Problem by using Improved Ant Colony Optimization Algorithm," International Journal of Information and Education Technology, Vol. 1, pp. 404-409, December 2011.

[4] K.P. Wang, L. Huang, C.G. Zhou, and W. Pang, "Particle Swarm Optimization for Traveling Salesman Problem," Proc. International Conference on Machine Learning and Cybernetics, pp. 1583-1585, November 2003.

[5] Z. Hua, and F. Huang, "A Variable Grouping based Genetic Algorithm for Large-Scale Integer Programming," Information Science, Vol. 176, pp. 2869–2885, 2006.

[6] J.C. Bansal, H. Sharma, S.S. Jadon, and M. Clerc, "Spider Monkey Optimization for Numerical Optimization," Memetic Computing, Vol. 47, pp. 31-48, 2014.

[7] M.M. Symington, "Fission-Fusion Social Organization in Ateles and Pan," International Journal of Primatology, Vol. 11, pp. 47-61, February 1990.

[8] V. Agrawal, R. Rastogi, and D.C. Tiwari, "Spider Monkey Optimization: a survey," International Journal of System Assurance Engineering and Management, Vol. 9, pp. 929-941, August 2018.

[9] X. Wei, Z. Jiang-Wei, and Z. Hon-lin, "Enhanced Self - Tentative Particle Swarm Optimization Algorithm for TSP," Journal of North China Electric Power University, Vol. 36, no. 6, pp. 69–74, 2009.

[10] J. Zhang and W. Si, "Improved Enhanced Self–Tentative PSO Algorithm for TSP," Sixth IEEE International Conference on Natural Computation, pp. 2638–2641, August 2010.

[11] M.A.H. Akhand, S. Akter, M.A. Rashid, and S.B. Yaakob, "Velocity Tentative PSO: An Optimal Velocity Implementation based particles Swam Optimization to Solve Traveling Salesman Problem," International Journal of Computer Science, Vol. 42, July 2015.

[12] TSPLIB – a library of sample instances for the TSP. Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB9/