Performance Analysis of Genetic Algorithm, Particle Swarm Optimization and Ant Colony Optimization for solving the Travelling Salesman Problem

Valarmathi B[1], Santhi K[2], Ravi Chandrika[3], Peeyush Goel[4], Bhagyashree Bagwe[5]

***Abstract*:** *The Travelling salesman problem also popularly known as the TSP, which is the most classical combinatorial optimization problem. It is the most diligently read and an NP hard problem in the field of optimization. When the less number of cities is present, TSP is solved very easily but as the number of cities increases it gets more and more harder to figure out. This is due to a large amount of computation time is required. So in order to solve such large sized problems which contain millions of cities to traverse, various soft computing techniques can be used. In this paper, we discuss the use of different soft computing techniques like Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and etc. to solve TSP.*

***Index Terms*:** Ant Colony Optimization, Genetic Algorithm, Particle Swarm Optimization, Soft Computing, Travelling Salesman Problem.

I. INTRODUCTION

The TSP is used to discover the optimal path travelled by the salesman from a starting location to an ending location with a constraint that she / he visits only a certain set of cities and returns to the starting location, by traversing through one city only once, along with a condition that the total distance travelled in between the starting point and the ending point is minimized. This problem is declared as an NP hard problem because it is cannot be solved exactly in polynomial time. A minimum of exponential time is required to obtain an optimal solution.

Till now, many heuristic algorithms in the field of operation research methodologies have been developed to optimize this minimization problem. TSP can be used in several fields such as traffic, vehicle routing and military. It also includes other typical applications like, computer wiring, where the smallest length of wires must be used to build the circuit, cutting wallpaper, where we must use the lowest possible area of sheets for the wallpaper and job sequencing, where the parameters such as priority of jobs, time taken, resources required play the major role. It also has statistical applications like combinatorial data analysis.

The minimization problem is based along the distance between a two cities. It is figured by using Euclidean Distance and it is shown in the following equation 1:

$$d = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2} \qquad (1)$$

where (a1, b1) and (a2, b2) are the coordinates of city 1 and city 2 respectively.

- TSP comes under the category of permutation problem in which the main objective is to find the route of the shortest path taking by considering the constraint that every city must be visited only once. [3]
- The objective of this is to explore the various algorithms that can be used along with GA which will help in solving TSP.
- Another objective is improvising the existing algorithm into a better and more optimal version.

II. RELATED STUDIES

TSP is encountered in combinatorial problems which come under a general category of classic NP-Complete problem. Here, NP stands for "non-deterministic polynomial time". Different methods such as an integer linear programming, neural network approach which is based on the self-organized feature map model, branch and bound based method for solving the large-scale Multiple TSP (MTSP) and etc. are used to solve the algorithm. Some more approaches to TSP are GA, evolutionary simulated annealing, island-based GA, bat algorithm, imperialist competitive algorithm, firefly algorithm, Variable Neighbourhood Search (VNS) algorithm and etc. A Discrete Water Cycle Algorithm (DWCA) was proposed by Eneko Osaba et al. [2] for Asymmetric TSP (ATSP). DWCA can prove to be more reliable in solving TSP because it has a better performance in terms of optimality and we get more stable quality solutions by the DWCA. DWCA also provides with the best results with an outstanding performance in 73.68% of the TSP datasets. Samrat Hore et al. [3] proposed a new

Revised Manuscript Received on July 05, 2019.

Valarmathi B, Associate Professor, Department of Software Engineering, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India.

Santhi K, Associate Professor, Department of Analytics, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India.

Ravi Chandrika, B.Tech.(Information Technology) Student, Department of Information Technology, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India.

Peeyush Goel, B.Tech.(Information Technology) Student, Department of Information Technology, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India.

Bhagyashree Bagwe, B.Tech.(Information Technology) Student, Department of Information Technology, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India.

algorithm, the VNS algorithm, which was integrated with a stochastic approach for solving the TSP with an optimal solution. To evaluate the hardness / difficulty to solve the different cases of TSP, Miguel Cárdenas-Montes et al. [1] used direct, indirect measures. His methodology worked on histograms, where he fitted the normalized areas of the Weibull probability distribution, generated from the application of the Dirichlet tessellation for the TSP instances. A model was established between both variables whenever a high correlation was found. Yongbo Chen et al. [5] proposed an algorithm called Modified Two-part Wolf Pack Search (MTWPS), which used both two-part individual encoding approach and the transposition and extension (TE) operation for the MTSP.

Yuzhe Yan et al. [6] modified the ACO for the TSP, where strategies like adaptive tour construction and pheromone updating strategies were used to make the balance in the search space more intense and diverse. Xin Chen et al. [7] proposed the spherical TSP algorithm where all the cities were represented as points and solutions as paths, that resided on the surface of a sphere. A hybrid algorithm combining the Glow-worm Swarm Optimization and the 2-opt algorithm was proposed. In it, glow-worms carrying the luciferin between cities are converted to edges, by modifying the probability formula and the luciferin updation formula.

Yu Lin et al. [8] Developed a hybrid algorithm of simulated annealing and tabu search algorithm. By gathering enough properties of the hybrid algorithm, he was able to improve the search efficiency of the dynamic neighbourhood structure by decreasing the randomness of the conventional 2-opt neighbourhood. Mostafa Mahi et al. [9] Proposed a new hybrid method which used PSO to the ACO parameters that affect its performance. Also, to improve the local solutions, he added the 3-Opt heuristic method to the given method. Victer Paul et al. [10] Worked on permutation-coded GA and proposed different population growing techniques such as random, Gene Bank, Nearest Neighbour, Selective Initialization (SI), Sorted Population (SP), and etc. Bihter Avsar et al. [11] researched on divide-and-conquer problem with a parallelized approach by using a Self-Organizing Map (SOM) to solve the Euclidean TSP.

Manuel López-Ibánez et al. [12] supported this intuition by providing an example based on the TSP with time windows. From the above context, it was clear that to minimize the travel time, the problem variant should be well studied, while, to minimize the make span, the problem variant should be less studied. Contributions in dynamic environments were made by Michalis Mavrovouniotis et al. [13] where he proposed an ACO framework where he used different immigrant schemes such as random immigrants, memory-based immigrants and elitism-based immigrants. All these immigrants were integrated in the ACO to solve the dynamic problems. An effective an metaheuristic hybridized method was proposed by Zaniar Ardalan et al. [14] where he focused on local search procedure. Inspired from the social and political global search strategy, he proposed algorithm based on an imperialist competitive algorithm. All these algorithms provide an insight on how the problem can be solved.

III. PROPOSED ALGORITHMS

Different algorithms have different computing power and have some limitations. These limitations can be based on different parameters. Our proposed method works on comparing of one of these parameters, number of cities, for 3 different algorithms, which are GA, PSO and ACO. The additional work included is the addition of ACO algorithms in our paper, which competes with GA and PSO for 5 different set of number of cities. Performance wise results are listed out which can help us in building efficient hybrid systems for a certain size of datasets (number of cities).

A.GA

The GA is a method which learns from the natural selection process, that drives biological evolution. It is applied for solving both forced and unforced problems. At each step, the population of individual solutions is repeatedly updated and random selection for the position of parents from the current population is made using GA. The next generation is produced by using these parents to produce children. As we pass multiple generations, the population "evolves" toward an optimal solution. GA can solve a variety of problems that cannot be solved even by standard algorithms. These problems may include the major objective function as discontinuous, non-differentiable, stochastic, or highly nonlinear. The GA can also prove effective for mixed integer programming problems, which includes some integer-valued constrained components.

The three main types of rules that change the current population to next generation at every step in GA are:

a) Selection rules are applied to select the individuals, called parents, that aid in the generation of the next level population.

b) Crossover rules are applied to combine two parents to form children for the next generation.

c) Mutation rules are given to apply random changes to individual parents to make children.

The steps required in GA are Initial population, Calculate fitness, Check stopping condition, Selection, Crossover and Mutation.

Step 1: Generate initial random population.
Step 2: Calculate the fitness of individuals.
Step 3: Satisfy the stop criteria.
Step 4: Selection of the individuals.
Step 5: We can only shuffle the route using the crossover operator.
Step 6: Mutation Operator.
Step 7: Go to step 2.
Step 8: End.

B.PSO

PSO mimics the behaviors of bird flocking. In PSO, each single solution is a "bird" which is named "particle" in the search space. The majority of the particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles. PSO is initialized with a group of random particles and then searches for optima by updating generations. In every iteration, each particle is updated by subsequent two "best" values. The first one is the best solution it has achieved so far. This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer

is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. The steps applied in PSO are specified below.

Step 1: Population Initialization.
Step 2: Do 3-8 steps till we meet ending condition.
Step 3: Find fitness.
Step 4: Calculate pBest.
Step 5: Calculate gBest.
Step 6: Calculate velocity.
Step 7: Update current solution.
Step 8: Update current solution with pBest.
Step 9: Check for stopping condition.

C.ACO

ACO comes under the class of algorithms which mimics the actions of an ant colony. It includes artificial 'ants', also called simulation agents, which locates optimal solutions by representing all the possible solutions while moving through a parameter space. As real ants lay down a pheromone trail wherever they go, while searching their environment in order to direct each other towards the resources, similarly, the simulated 'ants' store their locations and the standard of their solutions, which helps them in locating better solutions in the later simulation iterations. The steps involved in ACO are given below:

Step 1: Initialization.
Step 2: Determine the probability of traversing from one city to another.
Step 3: Finding the best route.
Step 4: Update current solution with best cost and best path.
Step 5: Check for stopping condition

IV. EXPERIMENTAL STUDIES

We put forward a series of comparative experiments on GA, PSO and ACO. In each of the comparative experiments, all the three algorithms share the same configurations, and the same irrelevant variables, including same original population / particle swarm and the same city location

A.GA

The code for solving GA has been written and the output for 500 generations for 20 cities was obtained. Fig 4.1(a) shows the output for the first 5 generations and Fig 4.1(b) shows the output for the last 5 generations. Next, we obtained the output for 500 generations for 52 cities. Fig 4.2(a) shows the output for the first 5 generations and Fig 4.2(b) shows the output for the last 5 generations.

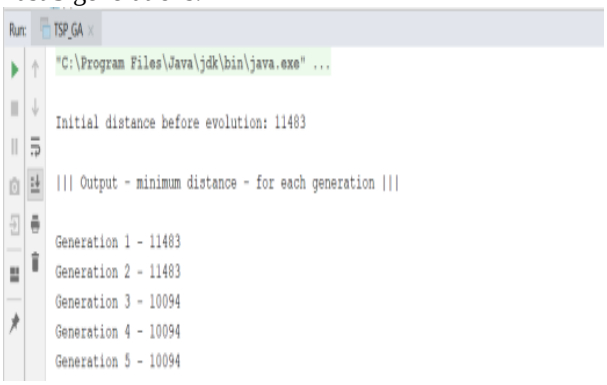

Fig 4.1 (a) Output for the first 5 generations for 20 cities using GA

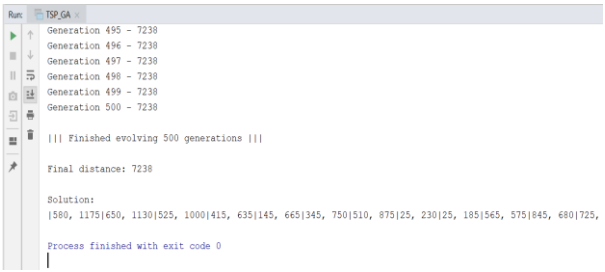

Fig. 4.1(b) Output for the last 5 generations for 20 cities using GA along with the final solution

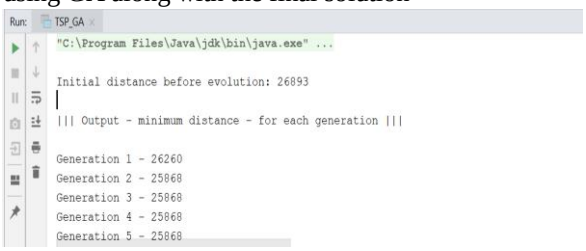

Fig 4.2. (a) Output for the first 5 generations for 52 cities using GA

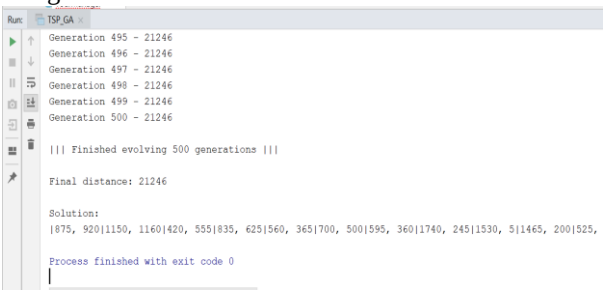

Fig 4.2. (b) Output for the last 5 generations for 52 cities using GA along with the final solution

B.PSO

The code for solving PSO has been written and the output for 20 cities was obtained. Fig 4.3(a) shows the pBest for the last 3 particles along with the final gBest for 20 cities and Fig 4.3(b) shows the graph(path travelled from one city to another) for 20 cities. Next, we obtained the output for 52 cities. Fig 4.4(a) shows the pBest for the last 3 particles along with the final gBest for 52 cities and Fig 4.4(b) shows the graph(path travelled from one city to another) for 52 cities.

Fig 4.3 (a) Final Output for 20 cities using PSO

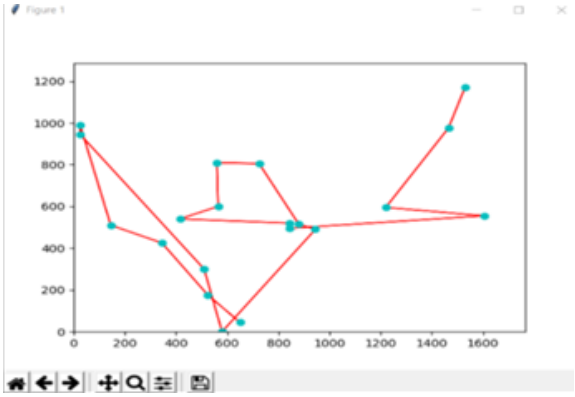

Fig 4.3 (b) Graph for 20 cities using PSO

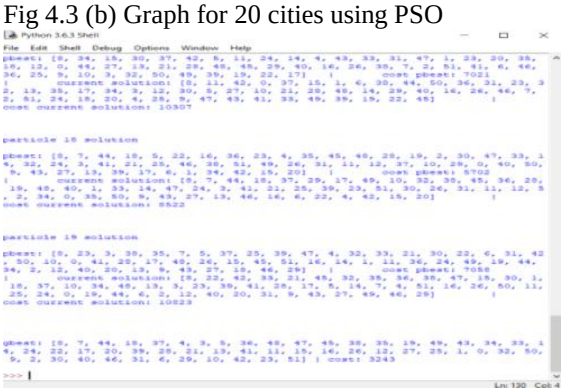

Fig 4.4 (a) Final Output for 52 cities using PSO

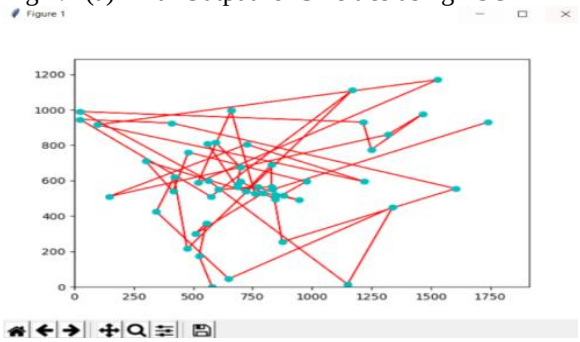

Fig 4.4 (b) Graph for 52 cities using PSO

C.ACO

The code for solving ACO has been written and the output for 500 generations for 20 cities was obtained. Fig 4.5(a) shows the output for the last 15 generations and the final cost for 20 cities and Fig 4.5(b) shows the graph(path travelled from one city to another) for 20 cities. Next, we obtained the output for 500 generations for 52 cities. Fig 4.6(a) shows the output for the last 5 generations and the final cost for 52 cities and Fig 4.6(b) shows the graph(path travelled from one city to another) for 52 cities.

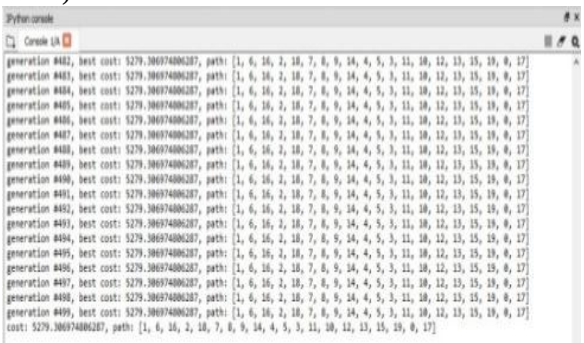

Fig 4.5 (a) Final Output for 20 cities using ACO

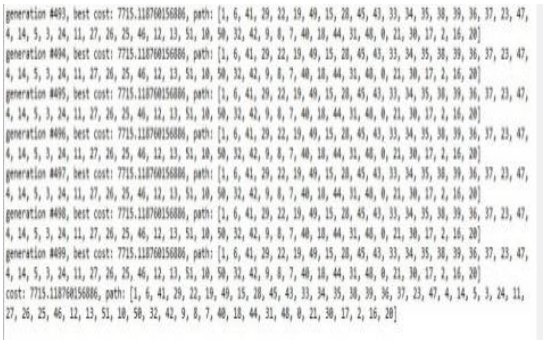

Fig 4.6 (a) Final Output for 52 cities using ACO

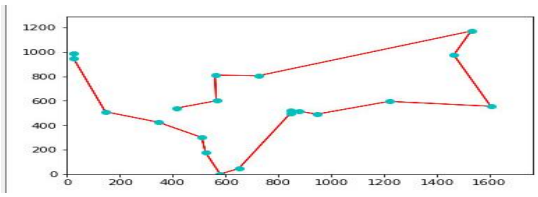

Fig 4.5 (b) Graph for 20 cities using ACO

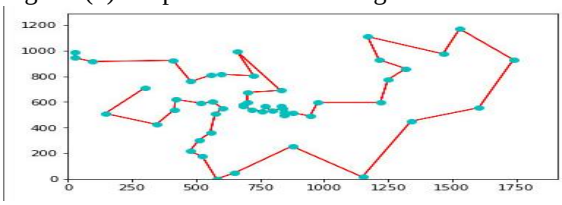

Fig 4.6 (b) Graph for 52 cities using ACO

V. RESULTS AND CONCLUSIONS

Results obtained from the above output screens(section 4) are presented in a tabular form in Table 1 where we put forward the no. of cities/dataset size(chromosome size) and the best cost obtained by PSO, ACO and GA algorithms for these number of cities. The above table is converted in graphical format (Fig 5.) for easier representation and interpretation of the values.

Table 1 Comparison table for performance of PSO, ACO and GA at 10, 20, 30, 40 and 52 number of cities.

SIZE	COST		
	PSO	ACO	GA
10	8002	2821	2823
20	7597	5280	8680
30	7084	6379	12485
40	4121	6742	15750
52	3243	7715	22385

Table 2 Rank-wise comparison for GA,PSO and ACO for 10,20,30,40 and 52 cities

Size	First Best	Second Best	Third Best
10	ACO	GA	PSO
20	ACO	PSO	GA
30	ACO,PSO	ACO,PSO	GA
40	PSO	ACO	GA
52	PSO	ACO	GA

GA, PSO and ACO are used with 5 different sizes of datasets for each of the algorithms, and produced the results from it. The produced output was compared with different dataset size. Furthermore

study and research can be carried out between these algorithms on the basis of other parameters, like no. of iterations, or the parameters passed, etc. These comparisons can help us understand the weak points of these algorithms which will help in producing better hybrid systems by combining 2 or more algorithms for a particular dataset size in future.

Fig.5. Comparison graph for performance of PSO, ACO and GA at 10, 20, 30, 40 and 52 number of cities

Summary of experiments are given below and the rank-wise comparison from Table 2 gives us the following key points:

1. At very low values of a number of cities in a chromosome, i.e, 10, ACO performs the best, followed by GA, followed by PSO.
2. At lower values of a number of cities in a chromosome, i.e, 20, ACO performs the best, followed by PSO, followed by GA.
3. As the medium value of a number of cities in a chromosome, i.e, 30, ACO and PSO both perform the best, followed by GA.
4. At higher values of a number of cities in a chromosome, i.e., 40 and 52, PSO performs the best, followed by ACO, followed by GA.

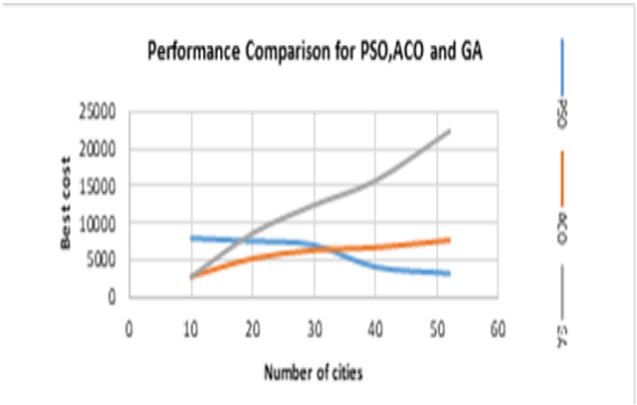

Fig 5. Comparison graph for performance of PSO, ACO and GA at 10, 20, 30, 40 and 52 number of cities

Summary of experiments are given below and the rank-wise comparison from Table 2 gives us the following key points:
1. At very low values of a number of cities in a chromosome, (i.e,) 10, ACO performs the best, followed by GA, followed by PSO. 2. At lower values of a number of cities in a chromosome, (i.e,) 20, ACO performs the best, followed by PSO, followed by GA. 3. As the medium value of a number of cities in a chromosome, i.e, 30, ACO and PSO both perform the best, followed by GA. 4. At higher values of a number of cities in a chromosome, i.e., 40 and 52, PSO performs the best, followed by ACO, followed by GA.

REFERENCES

[1] Cárdenas-Montes, "Creating hard-to-solve instances of travelling salesman problem", Applied Soft Computing, vol. 71, pp. 268-276, 2018.
[2] Osaba, E., Del Ser, J., Sadollah, A., Bilbao, M. N., & Camacho, "A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem", Applied Soft Computing, vol. 71, pp. 277-290, 2018.
[3] Hore, S., Chatterjee, A., & Dewanji, "Improving variable neighborhood search to solve the traveling salesman problem", Applied Soft Computing, vol. 68, pp. 83-91, 2018.
[4] Zhou, H., Song, M., & Pedrycz, "A comparative study of improved GA and PSO in solving multiple traveling salesmen problem", Applied Soft Computing, vol. 64, pp. 564-580, 2018.
[5] Chen, Y., Jia, Z., Ai, X., Yang, D., & Yu, "A modified two-part wolf pack search algorithm for the multiple traveling salesmen problem", Applied Soft Computing, vol. 61, pp. 714-725, 2017.
[6] Yan, Y., Sohn, H. S., & Reyes, "A modified ant system to achieve better balance between intensification and diversification for the traveling salesman problem", Applied Soft Computing, vol. 60, pp. 256-267, 2017.
[7] Chen, X., Zhou, Y., Tang, Z., & Luo, "A hybrid algorithm combining glowworm swarm optimization and complete 2-opt algorithm for spherical travelling salesman problems", Applied Soft Computing, vol. 58, pp. 104-114, 2017.
[8] Lin, Y., Bian, Z., & Liu, "Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing–tabu search algorithm to solve the symmetrical traveling salesman problem", Applied Soft Computing, vol. 49, pp.937-952, 2016.
[9] Mahi, M., Baykan, Ö. K., & Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem", Applied Soft Computing, vol. 30, pp. 484-490, 2015.
[10] Paul, P. V., Moganarangan, N., Kumar, S. S., Raju, R., Vengattaraman, T., & Dhavachelvan, "Performance analyses over population seeding techniques of the permutation-coded genetic algorithm: An empirical study based on traveling salesman problems", Applied Soft Computing, vol. 32, pp. 383-402, 2015.
[11] Avşar, B., & Aliabadi, D. "Parallelized neural network system for solving Euclidean traveling salesman problem", Applied Soft Computing, vol. 34, pp. 862-873, 2015.
[12] López-Ibáñez, M., Blum, C., Ohlmann, J. W., & Thomas, B. "The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization", Applied Soft Computing, vol. 13, pp. 3806-3815, 2013.
[13] Mavrovouniotis, M., & Yang, "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors", Applied Soft Computing, vol. 13, pp. 4023-4037, 2013.
[14] Ardalan, Z., Karimi, S., Poursabzi, O., & Naderi, "A novel imperialist competitive algorithm for generalized traveling salesman problems", Applied Soft Computing, vol. 26, pp. 546-555, 2015.