A Comparative Analysis of Genetic Algorithms, Ant Colony Optimization, Q-Learning and SARSA for Solving the Travelling Salesman Problem

Muhammad Najeeb Jilani
Department of Computer Science
Habib University
Karachi, Pakistan
mj06879@st.habib.edu.pk

Sajeel Nadeem Alam
Department of Computer Science
Habib University
Karachi, Pakistan
sa06840@st.habib.edu.pk

***Index Terms*—Genetic Algorithms, Ant Colony Optimisation, Q-Learning, SARSA Learning, Travelling Salesman Problem**

I. Introduction:

In the field of Artificial Intelligence there have been numerous techniques and attempts to solve complex problems. One such problem is the Travelling Salesman Problem (TSP). Given a set of cities and the distance between every pair of cities, the objective is to find the shortest route from a starting city, visiting every city and then returning to the starting point. Although this problem may sound simple, in theory it is an NP-hard problem due its vast search space, thus it cannot be solved in polynomial time [1].

TSP is classical problem and has many applications such as vehicle routing, manufacturing and logistics. Due to its wide variety of application, multiple AI based techniques have been employed and compared to solve this optimisation problem such as Evolutionary Algorithms and Reinforcement Learning [2]. In order to provide a more diverse comparison, an extensive background study of four different strategies: Genetic Algorithms, Ant Colony Optimisation, Q-Learning and SARSA Learning was performed. After understanding their theoretical foundations from existing literature, all four methods were used to solve the qa194 dataset of the travelling salesman problem obtained from World TSP [3]. The core of our research lies in the comparative analysis which was carried out on the obtained results. The performance of each algorithm was assessed based on the optimum results they achieved along with the number of iterations taken. The results are illustrated with the help of graphs which allowed us to gain valuable insights on the strengths and limitations of each technique. The overall goal of the research was to gain a better understanding of existing Artificial Intelligence based techniques that can be used to solve complex problem that have various real life applications such as the Travelling Salesman Problem.

II. Literature Review:

Over the past few years, multiple techniques and implementations have been used to attempt to solve the TSP. A 2017 paper by Mohammed Alhanjouri explores several optimisation techniques such as genetic algorithm, simulated annealing, particle swarm optimization, ant colony optimization, bacteria foraging optimization, and bee colony optimization. After performing a comparative analysis, the results stated that simulated annealing, particle swarm optimization, and ant colony optimization were successful in solving the problem whereas genetic algorithm required substantial memory to display its potential and reach an optimal solution [4]. A more recent paper also explored genetic algorithms, ant colony optimization and particle swarm optimisation and its results presented that although ant colony optimization has the longest running time, it achieved the most optimal solution whereas genetic algorithm is more efficient when the size of the problem is increased and provides a better solution than particle swarm optimization [5].

Another paper compares genetic algorithms and epsilon-greedy Q-learning (EQLA) by running them on two different datasets and comparing their speeds and accuracy with regards to solving the problem. The results stated that genetic algorithm takes less time and reaches a more optimal solution as compared to EQLA and is thus a better option in the case of optimising the TSP [6]. Moreover, a 2021 paper which primarily explores the performances of two Reinforcement Learning techniques: Q-Learning and SARSA, found that fine tuning the value of ϵ to 0.01 in the ϵ - greedy policy achieves the best solution in 15 out of 16 cases [7]. After going over existing literature, Genetic Algorithms and Ant Colony Optimisation were selected as Evolutionary Algorithms techniques whereas Q-learning and SARSA Learning were selected as Reinforcement Learning techniques to be included in our comparative analysis for solving the travelling salesman problem.

III. Background Study

A. Genetic Algorithms

Genetic Algorithms were introduced by John Holland in 1975. In artificial intelligence, a genetic algorithm is a generic population based meta heuristic optimisation algorithm which is inspired by natural evolution. Initially a population is randomised where each member of the population is called a chromosome and is a candidate solution to the problem. In order to determine how good or bad a solution is, each chromosome is assigned a fitness value which is calculated by a fitness function. The goal of the fitness function is to guide the entire process, by setting the criteria for an optimal solution. In order to do so, it considers a number of parameters according to the heuristic which varies with the problem. From the population, pairs of chromosomes are selected using some selection scheme (tournament selection, random, best two etc) which are then used as parents for the next generation. Each pair of parents goes through recombination (crossover) where new off-springs are generated. These off-springs are mutated (small change) with a mutation rate and added to the population. In order to maintain the population size, a survivor selection scheme is implemented which determines the chromosomes that are sent to the next generation. This process is repeated for a set number of generations until the algorithm converges to an optimal value [8]. In genetic algorithms it is essential to select schemes such that both exploration and exploitation are balanced. Exploration makes sure that we are exploring the entire search space finding promising solutions whereas exploitation makes sure that we eventually converge to an optimal point. If there is too much exploration then the algorithm will not converge, and if there is too much exploitation then the algorithm will reach an early convergence, not reaching an optimal solution.

In the case of TSP, each chromosome is a possible tour that is represented as a list of n cities. Each gene in the chromosome represents a city of a total of n cities, and the order of the genes determines the order in which the cities are visited on tour. The initial population of candidate solutions is a nested list of such chromosomes. The fitness value of a tour is inversely proportional its distance cost. In each generation, pairs of parents are selected and single-point crossover is carried out which produces two off-springs that can be mutated. Truncation survivor selection is used where the tours with the lowest distance cost and highest fitness values are sent to the next generation. This is run for a set number of generations until a path is found which has the optimal distance cost.

B. Ant Colony Optimisation

On the other hand, Ant Colony Optimisation was introduced by Marco Dorigo in 1992. It is an optimisation technique inspired by the foraging behaviour of ants when they search for food. Ants communicate with each other using a trace called pheromones which they leave behind while taking a path to search for food. The shorter the path, the more frequent it is tread upon thus increasing its pheromone concentration. The path with higher pheromone concentration is more desirable for other ants hence eventually all the ants start following the shortest path to the food.

In the case of TSP, a set number of ants is determined and each ant is given a random starting city. The initial pheromone concentrations (τ) for every edge (path from one city to another) is randomised and stored in a table called the τ table. While calculating the probability of determining the next city to visit, the ant takes into the consideration the pheromone concentration (τ) of the edge connecting the current city to the next city and the desirability (η) of the next city which is inversely proportional to its distance from the current city. The effect of the pheromone concentration (τ) is determined by a constant alpha α whereas the effect of desirability (η) is determined by β. Thus the probability of going from the current city (i) to the next city (j), where the next city can only be from the set k of unvisited cities is calculated using the following formula:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{k \in \text{allowed}_k} \left[\tau_{ik}(t)\right]^{\alpha} \cdot \left[\eta_{ik}\right]^{\beta}} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases}$$

Using this process of selecting the next city iteratively, an ant is able to construct a complete tour. Once an ant completes its tour, the pheromone value of every edge is updated taking into consideration the edges visited by the ant and the evaporation rate. This ensures that the decisions taken by one ant influences the decisions taken by other ants while making their own tours hence establishing communication between them. The algorithm is terminated once all the cycles are completed or when we converge to an optimal solution. Following the trails of other ants while selecting the next city in the tour is exploitative as ants are more likely to follow the paths that previous ants found successful whereas taking into consideration the desirability of the next city which is essentially how close the next city is ensures exploration as ants are more likely to explore closer cities that may not yet have a strong pheromone trail. Similar to genetic algorithms, it is essential to balance exploration and exploitation by managing the effects of pheromones and desirability while calculating the probability of selecting the next city and this is done by selecting suitable values of alpha α and β [9].

C. Q-learning:

Q-learning is a well-known model-free off-policy reinforcement learning algorithm with a key goal of learning a policy which is optimal. The strategy which is used by the agent to take an action given the state of the agent in the environment is known as the policy. The policy which maximizes the expected value of the total reward is considered to be the optimal policy.

It uses a value function, called the Q-value which is used to estimate the expected cumulative rewards for choosing certain actions at each state. In Q-learning, there is a Q-table which contains values that are calculated by Q-function. The Q-function requires two parameters known as the state (s) and the action (a). The Q-value of a state-action pair (s, a), denoted as $Q(s, a)$, represents the expected cumulative reward the agent will receive by starting from state s, taking action a, and following an optimal policy thereafter.

As Q-learning is an off-policy RL technique, its behaviour policy is not the same as target policy. The policy that the agent uses to determine its action in a given state is known as the behaviour policy while the policy that the agent uses to learn from the rewards received for its actions to determine update Q-value is known as the target policy. Through the target policy, agents learn the desirability of an action in a given state.

The Q-learning technique uses an iterative approach to update the Q-values based on the agent's experience. At each time point, the action chosen by the agent is dependent on a strategy that balances exploration and exploitation, such as ϵ-greedy. In ϵ-greedy strategy, the probability of selecting a random action (exploration) is ϵ and the probability of selecting an action among all possible actions which maximizes the Q-value (exploitation) is $(1 - \epsilon)$.

The equation used to update the Q value for Q-Learning is defined as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

where s denotes the state at which the agent is currently at, a denotes the chosen action, r represents the immediate reward received after selecting action a at state s, s' is the next state after selecting action a, α denotes the learning rate ($0 < \alpha < 1$) which decides the weight assigned to the newly received information, and γ denotes the discount factor ($0 < \gamma < 1$) that lowers the effect of the future rewards and converge the reward. The Target policy and behaviour policies are different as our Q-learning technique is an off-policy technique so the Q-value update in Q-learning is based on the maximum Q-value of the next state, regardless of the action taken. Q-learning tends to be more explorative, as it updates Q-values depending on the maximum Q-value of the next state, even if the selected action was exploratory.

The Q-Learning algorithm continues to interact with the environment, updating the Q-values based on observed rewards and transitioning to new states until it converges to an optimal policy. Convergence is achieved when the Q-values stabilize and no further significant updates occur [10].

D. SARSA-Learning

SARSA (State-Action-Reward-State-Action) is also a model-free On-Policy RL technique that aims to learn a policy that is optimal. It uses a strategy which incorporates the exploration-exploitation trade-off during learning. Similar to Q-Learning, SARSA estimates the Q-values, that denote the expected cumulative rewards which are associated with selecting specific actions at each state. However, SARSA learns and updates its Q-values based on the state-action pair at which the agent is currently and the subsequent state-action pair. As it is an on-policy RL algorithm its behaviour policy is identical to its target policy.

The SARSA-learning algorithm also uses an iterative approach to update the Q-values based on the agent's experience. However, the behaviour policy is the same as the Target policy so if there is an ϵ-greedy strategy to select an action a in current state s at a time step, then for the subsequent state-action pair (s', a') the target policy will also be ϵ-greedy.

The SARSA update equation is defined as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma Q(s', a') - Q(s, a) \right]$$

where s is the state at which the agent is currently at, a is the chosen action, r denotes immediate reward received after selecting action a at state s, s' denotes the next state after taking action a, a' is the action chosen in the next state s', α is the learning rate ($0 < \alpha < 1$) which decides the weight assigned to the newly received information, and γ is the discount factor ($0 < \gamma < 1$) that discounts the future rewards.

The update equation calculates the difference between the current Q-value estimate and the updated Q-value estimate based on the observed reward, the Q-value of the next state-action pair. The learning process continues until convergence, where the Q-values stabilize and no significant updates occur.

In the case of TSP, the states S are all the location points or nodes, actions A are the move from the current state

to the next state s' and the rewards are the negative Euclidean distance from one state to another. For not visiting a state that has already been visited, there is a flag to restrict and only not visited states would be available for visit.

We trained the agent for both Q-Learning and SARSA-Learning using a fixed number of episodes. Also, we variate the parameters to see the behaviour of algorithms on different values of epsilon (ϵ), and learning rate (α) [10].

IV. Experiments/Analysis Conducted

In order to ensure a fair comparison, all four algorithms were run on the same dataset consisting of 194 cities. Firstly, each algorithm was analysed individually using different parameter values to check which set of parameters gave the best results for the specific algorithm. Once this was determined, the algorithms were run together on a set number of epochs (4000) and each algorithm was run using the set of parameters from which it got the best results.
In the case of genetic algorithms, the population size was kept at 10 and 30, whereas the mutation rate varied between 0.1, 0.3 and 0.7 resulting in six different cases. See Fig 1.

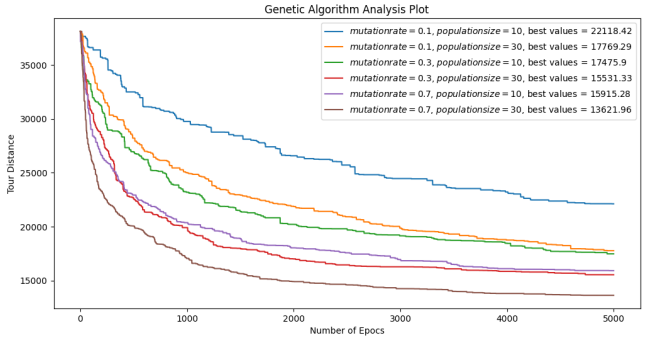

Fig. 1. Plot of Genetic Algorithm

As can be observed in Fig 1, a population size of 30 and a mutation rate of 0.7 gives the best results, achieving an optimal solution of 13621.

In the case of ant colony optimisation, the alpha α value was kept at 1 and 3, whereas the beta β value was kept at 10 and 7 resulting in four different cases. See Fig 2.

As can be observed in Fig 2, an alpha α value of 1 and a beta β value of 10 gives the best results, achieving an optimal solution of 10786.07.

In the case of Q-learning, the learning rate α was kept at $0.1, 0.5, 0.9$, whereas the ϵ was kept at 0.1 and 0.01 resulting in six different cases. See Fig 3 & 4.

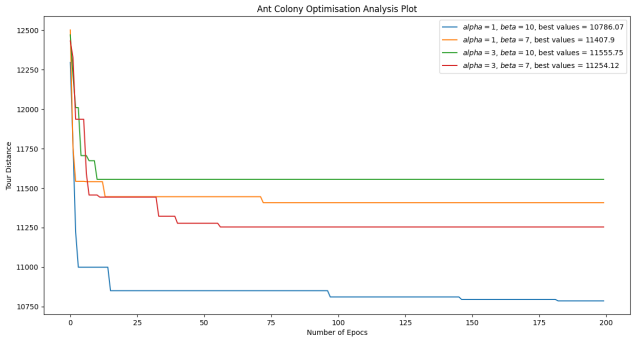

Fig. 2. Plot of Ant Colony Optimisation

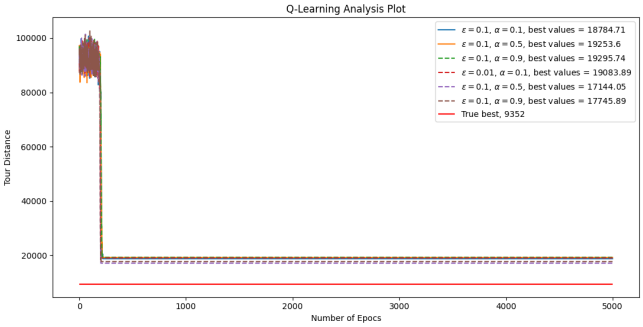

Fig. 3. Plot of Q-learning

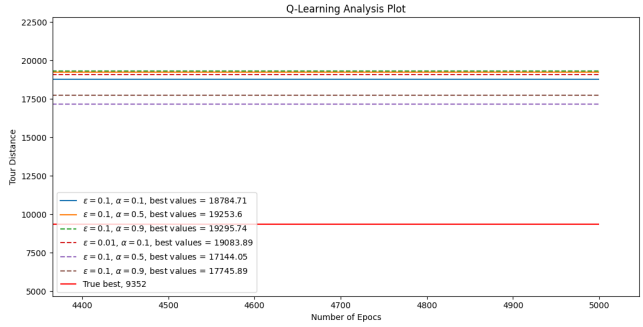

Fig. 4. Plot of Q-learning (asymptote zoom-in)

As can be observed in Fig 3 & 4, epsilon ϵ value of 0.1 and learning rate α value of 0.5 gives the best results, achieving an optimal solution of 17144.05. It can be observed that the α value of 0.9 with the same epsilon value is also very close but keeping a moderate learning rate α resulted in better.

In the case of SARSA-learning, the learning rate α was kept at $0.1, 0.5, 0.9$, whereas the ϵ was kept at $0.1\&0.01$ resulting

in six different cases. See Fig 3 & 4.

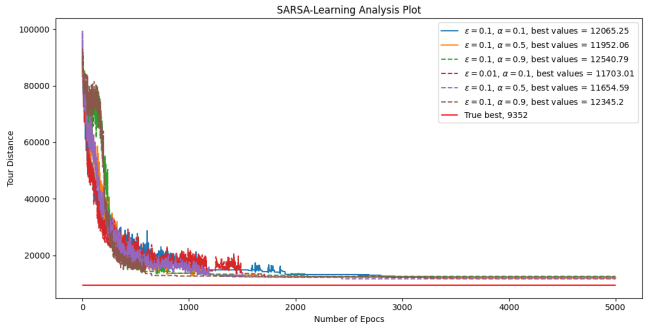

Fig. 5. Plot of SARSA-learning

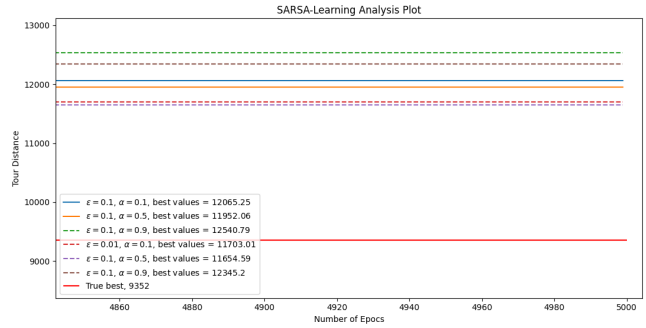

Fig. 6. Plot of SARSA-learning (asymptote zoom-in)

As can be observed in Fig 5 & 6, epsilon ϵ value of 0.1 and learning rate α value of 0.5 gives the best results, achieving an optimal solution of 11654.05. It can be observed that the α value of 0.1 and ϵ value of 0.01.

Now for a complete and fair comparison, we took the best parameters for each algorithm. These are the parameters for which each of them got the most optimal among the variations. See Fig 7 & 8:

As can be observed in Fig 7 & 8 that Q-learning obtained the optimal value of distance 20060.22, Genetic Algorithm obtained the optimal value of distance 14767.15, SARSA-learning obtained the optimal value of distance 12055.37, Ant Colony Optimization Algorithm obtained the optimal value of distance 10755.97 & the True best-value reported is 9352. As compared to Genetic & Sarsa, Ant Colony Optimization and Q-learning converged in fewer iterations.

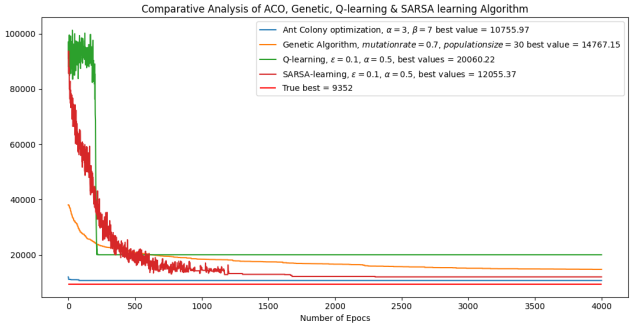

Fig. 7. Plot of Comparative analysis of ACO, Genetic Algorithm, Q-learning & SARSA learning algorithms

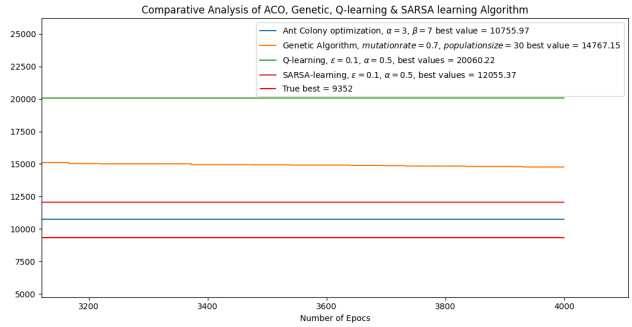

Fig. 8. Plot of Comparative analysis of ACO, Genetic Algorithm, Q-learning & SARSA learning algorithms (asymptote analysis)

According to our reflections, Ant Colony Optimisation has the best performance due its natural pheromone based appraoch that is well suited for optimisation problems specifically the TSP whereas Genetic Algorithm is a general optimisation approach which needs to be altered to solve the TSP. On the other hand, reinforcement learning techniques are generally sequential decision-making techniques that are not made for combinatorial optimisation problems such as the TSP. Moreover, ACO converges relatively quickly because ants communicate with each and thus build upon each other's tours, reinforcing the best tours whereas genetic algorithms can be slow to converge due to the random nature of its crossover and mutation operations.

V. Conclusion

In conclusion, we studied the theoretical implementations of four different AI based techniques: Genetic Algorithms, Ant Colony Optimisation, Q-Learning and SARSA performed a comparative analysis of all four by using their implementations to solve the Travelling Salesman Problem which is an NP hard problem. The comparative analysis was done in two parts. Firstly, each algorithm was analysed individually using different parameter values to check which set of parameters give the best results for the specific algorithm. Secondly, the algorithms were run together on a set number of epochs

and each algorithm was run using the set of parameters from which it got the best results. According to our results, Ant Colony Optimisation was able to achieve the most optimal solution in the least number iterations followed by SARSA Learning which took a considerably greater number of iterations. Genetic algorithm was also able to give a good optimal value however, it converged late as compared to the other algorithms. Lastly, Q learning gave the highest value (least optimal) however, it converges in fewer iterations as compared to Genetic Algorithms. This study can further be improved by implementing and analysing hybrid techniques between various Evolutionary Algorithms and Reinforcement Learning techniques.

Note: The implementation of Ant Colony Optimisation, Genetic Algorithm and Q-Learning were obtained from the sources [11, 12, 13] respectively, while SARSA was our implementation (modified code).

REFERENCES

[1] S. Sangwan, "Literature review on travelling salesman problem," *International Journal of Research*, vol. 5, p. 1152, 06 2018.

[2] A. Uthayasuriyan, H. G, K. UV, S. Mahitha, and G. Jeyakumar, "A comparative study on genetic algorithm and reinforcement learning to solve the traveling salesman problem," *Research Reports on Computer Science*, pp. 1–12, 05 2023.

[3] U. of Waterloo. (Year of Access) Tsp world - countries. [Online]. Available: https://www.math.uwaterloo.ca/tsp/world/countries.html

[4] M. Alhanjouri, "Optimization techniques for solving travelling salesman problem," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 7, pp. 165–174, 03 2017.

[5] Z. Wu, "A comparative study of solving traveling salesman problem with genetic algorithm, ant colony algorithm, and particle swarm optimization," in *Proceedings of the 2020 2nd International Conference on Robotics Systems and Vehicle Technology*, ser. RSVT '20. New York, NY, USA: Association for Computing Machinery, 2021, p. 95–99. [Online]. Available: https://doi.org/10.1145/3450292.3450308

[6] A. Uthayasuriyan, H. G, K. UV, S. Mahitha, and G. Jeyakumar, "A comparative study on genetic algorithm and reinforcement learning to solve the traveling salesman problem," *Research Reports on Computer Science*, pp. 1–12, 05 2023.

[7] A. Ottoni, E. Nepomuceno, M. Oliveira, and et al., "Reinforcement learning for the traveling salesman problem with refueling," *Complex Intelligent Systems*, vol. 8, p. 2001–2015, 2022.

[8] K. Man, K. Tang, and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 5, pp. 519–534, 1996.

[9] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents. ieee trans syst man cybernetics - part b," *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 26, pp. 29–41, 02 1996.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press Cambridge, 1998.

[11] R. Zhang. Ant colony optimization for traveling salesman problem. GitHub. [Online]. Available: https://github.com/ppoffice/ant-colony-tsp

[12] M. Asadolahi. Solving tsp with evolutionary genetic algorithm heuristic (python). GitHub. [Online]. Available: https://github.com/MohammadAsadolahi/Solving-TSP-with-Evolutionary-Genetic-Algorithm-Heuristic-Python

[13] M. Bagherpour. Tsp q-learning. GitHub. [Online]. Available: https://github.com/mehdibnc/TSP-Q-Learning