Research Reports on Computer Science
https://ojs.wiserpub.com/index.php/RRCS/

Research Article in Special Issue: Selected Papers from the 4th International Conference on Machine Learning, Image Processing, Network Security and Data Sciences (MIND-2022)

A Comparative Study on Genetic Algorithm and Reinforcement Learning to Solve the Traveling Salesman Problem

Agash Uthayasuriyan[1], Hema Chandran G[1], Kavvin UV[1], Sabbineni Hema Mahitha[1], Jeyakumar G[2*]

[1]Department of Electrical and Electronics Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India
[2]Department of Computer Science and Engineering, Amira School of Computing, Coimbatore, Amrita Vishwa Vidyapeetham, India
E-mail: g_jeyakumar@cb.amrita.edu

Received: 8 March 2023; **Accepted:** 28 March 2023

Abstract: Machine Learning (ML) and Evolutionary Computing (EC) are the two most popular computational methodologies in computer science to solve learning and optimization problems around us, respectively. It is of research interest in the literature, for exploring these two methodologies and to formulate algorithmic frameworks with '*EA for ML*' and '*ML for EA*' where EA stands for Evolutionary Algorithm. The objective of this paper is on exploring this dimension of research. The Traveling Salesman Problem (TSP) is one of the NP-hard (nondeterministic polynomial time hard) problems in combinatorial optimization problems. The solution for a TSP is the shortest path covering all the nodes in a given city. This paper compares two algorithms, "Genetic Algorithm (GA)" of the EC domain and "Epsilon-Greedy Q-Learning Algorithm (EQLA)" of the ML domain on solving TSP. The detailed design methodology involved in both these algorithms is discussed in this paper. The experiments are carried out on two different data sets (*random* and *ATT48*) to compare the speed and accuracy of the algorithms. The comparative results reveal that the GA could solve the TSP more effectively than EQLA. The obtained inferences along with the limitations are presented in this paper.

Keywords**:** GA, EA, reinforcement learning, EQLA, TSP

1. Introduction

The Evolutionary Algorithms (EAs), the potential optimization algorithms of the Evolutionary Computing (EC) paradigm, use the mechanisms drawn from nature to solve optimization problems by simulating the actions behind the principle of *survival of the fittest*. The EAs are widely used as a ready-made tool for solving a variety of benchmarking problems and real-world optimization problems. Though the applicability of the EAs started with solving optimization problems, there are research attempts and success stories of extending them to solving learning problems too.

The Machine Learning (ML) paradigm of computer science provides us with a large list of algorithms for solving the challenging prediction problems around us [1-3]. These algorithms imitate how human learns and increases the accuracy of prediction over the experience [4, 5]. The research attempts on integrating ML and EA paradigms are showing an increasing trend in the literature. Towards this, this paper is a research attempt to showcase how the same problem can be modeled and solved as an optimization problem as well as a learning problem.

The Genetic Algorithm (GA), a subset of EA, is widely used for solving complex optimization problems that involve single or multi, or many objective functions [6-8]. Epsilon-Greedy Q-Learning Algorithm (EQLA) is a type of Reinforcement Learning (RL) algorithm. The EQLA is more suitable to solve the optimization problem than other reinforcement models. This paper solves the Traveling Salesman Problem (TSP) by GA and EQLA, along with an attempt to study the working of a bio-inspired optimization algorithm and a traditional ML algorithm with thorough comparison and formulating valuable inferences.

The remaining part of the paper is organized as follows. Section 2 presents the work related to solving TSP by using GA and RL. Section 3 explains the design of the experiments carried out in this study. The results recorded and the inferences obtained are presented in Section 4, and finally, Section 5 concludes the paper.

2. Related work

A few recent studies that solve TSP using GA and RL conducted by various authors have been summarised in this section. The TSP is solved based on two different models using GA, by employing the mutation and elitism procedures presented in [9]. The results show that the model which uses 0 and 1 probabilities shows better performance than the other model, which is based on different probabilities when traversing from one node to another. In [10], the authors have used a new crossover technique that has the ability to calculate an approximate optimal path for solving TSP using GA and compare it with the existing algorithm - Sequential Constructive Crossover (SCX). It is found that in terms of cost and time, the new crossover technique is better compared and hence provides an optimal solution.

To increase population diversity and optimize mutation characters, an improved GA combining random crossover and dynamic mutation is proposed in [11]. The Traditional GA (T-GA), Adaptive Crossover Probability Genetic Algorithm (ACP-GA), Improved Selection Genetic Algorithm (IS-GA), and Random Crossover and Dynamic Mutation Genetic Algorithm (RCDM-GA) are employed to solve the classical TSP. The comparison of the simulated results of four algorithms demonstrates that the convergence rate and optimal solution of RCDM-GA are better compared to those of the other algorithms. The study presented in [12] explains the ability of GA to optimize in order to arrive at a viable solution for TSP. According to the authors, heuristic methods and genetic approaches are the best ways to solve TSP, and different possible combinations of selection, crossover, and mutation methodologies can be used to find multiple optimal solutions. One such combination of local optimization heuristics (reduces the search domain) and phenotype GAs (eliminates the generation of invalid tours) is followed in [13] and applied to TSP. An efficient GA for solving the Generalized Traveling Salesman Problem (GTSP) has been presented in [14] based on local connections and global connections.

The Biased Random-Key Genetic Algorithm (BRKGA) is applied to the TSP in [15], it is explained that BRKGA is an efficient methodology for solving problems involving combinatorial optimization, but it needs its parameters tuned to ensure that the algorithm's intensification and diversification are balanced. In [16], for solving TSP, the authors used RL, specifically Q-Learning, as a ML technique in Adaptive Operator Selection (AOS) inside the Iterated Local Search (ILS) meta-heuristic. The ILS incorporates multiple local searches and perturbation technicians, and Q-Learning has been incorporated into the ILS algorithm on two different levels: (1) Choosing suitable local search operators and (2) Choosing suitable perturbation operators at every phase of the search process. RL is also a potential method to work out the Traveling Salesman Problem With Refuelling (TSPWR). [17] explains that an RL-TSPWR algorithm and statistical methods are applied for tuning RL parameters (reinforcement function, α, ε, and γ).

In [18], the authors have presented a hybrid system that employs a RL agent to improve the TSP performance of a GA. At each generation, the agent employs Q-Learning, which influences the selection of both crossover and mutation operators, as well as the selection of individuals for reproduction. This hybrid system learns fasters and results in an overall better solution. While examining the connection between the best possible mutation rate and problem complexity, the authors of [19] present that they are inversely proportional. A learning-based method for solving the Multiple Travelling Salesman Problem (MTSP) is presented in [20], a two-stage approach is implemented where RL is used to train the model and a standard optimization technique is used to resolve the single-agent TSP related to each agent.

In [21], the author has solved TSP using GA and compared the solution with NNA (Nearest Neighbor Algorithm) and the Simulated Annealing (SA) technique. The author also presents that GA performs better than NNA and SA. Similarly, the study presented in [22] explains that solving TSP using GA and comparing it with NNA has proven that GA outperforms the NNA solution when the number of cities is increased. The author in [23] explains that an ant colony TSP optimization approach performs better when compared to NNA. In [24], dynamic programming with fuzzy logic is used to solve TSP and mentions that a multivariate problem can be solved in simple ways by decomposing it. The approach also explains that the fuzzy logic used is also able to perform appreciably well in comparison with other optimization algorithms

After analyzing the prior works, this paper compares the performance of GA and RL in solving TSP toward formulating novel integrated frameworks of GA and RL.

3. Design of experiments

This paper investigates and compares the GA alongside EQLA on an optimization problem like TSP. In TSP, given a set of cities with coordinates, a salesman needs to choose the shortest path to cover all the cities and return to the initial city. The problem could be solved through the brute-force method, but the complexity of the problem increases and requires more time to obtain the solution. To overcome this limitation and to obtain optimal results that are computationally effective and achieved quickly, algorithms such as the GA and EQLA are chosen to obtain a quasi-optimal solution. The next subsection provides a short summary of the algorithmic structures of GA and EQLA used for solving TSP.

3.1 *Structure of GA*

The GA is well known for providing fast and optimal solutions. It solves complex optimization problems by iterative random search of solutions. GA works based on evolutionary principles inspired by nature. GA approaches the TSP as an optimization problem by evaluating the routes and improving the search with various techniques. The working methodology is as in Figure 1.

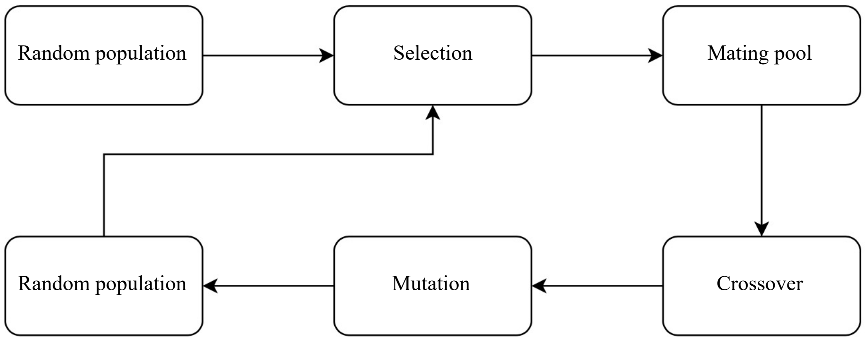

Figure 1. A simple structure of GA

The GA model starts with a random set of the population (routes). This set of routes has a predefined size. For each individual in the population set, the evaluation is based on its fitness value. In TSP, the fitness value is considered to be the inverse of the distance taken to traverse through all the cities in the given path. A lower distance value implies a higher fitness score and a higher distance value implies a lower fitness score. The population set is then ranked based on these fitness values ranging from the scores with the highest values to the least. This ranked set of individuals is used for the selection process. In selection, a particular number of best individuals (elitism) are passed directly to the mating pool as shown in Figure 2. With this, the best performing routes are automatically carried over to the next generation, ensuring the routes with possible solution set persists. A mating pool is a collection of parents that are used for generating a new population for the next generation. The other individuals (except the elite individuals) are made to crossover and mutate.

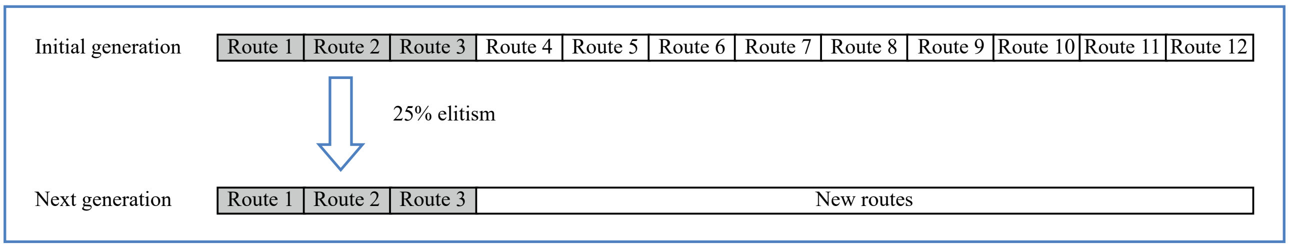

Figure 2. A sample for elitism in GA

Generally, in the crossover, two random parents are chosen and a crossover point is chosen. The parents are then spliced and joined together to form a new child. However, in TSP, since all the cities must be visited only once, a random subset of parent 1 is considered and the remaining genes (cities) are filled from parent 2 without any duplication of the cities present in the subset of parent 1, forming a new child (new route) as depicted in Figure 3.

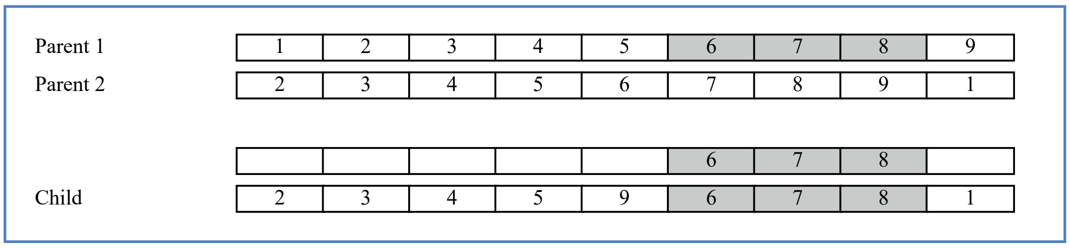

Figure 3. A sample GA crossover

The mutation is carried out for the children produced as a result of crossover. It acts as an important function in GA by introducing novel routes and the possible chance of local convergence is ruled out. For every individual, based on the specified low probability of cities in a route, swap mutation takes place resulting in the swapping of cities in a route. These distinct individuals along with the best individuals from the previous generation form a new population set and then the process is repeated. In this experiment, a population of 100 routes has been considered, for the algorithm to work on various individuals at once, and the elitism value is kept as 25% of the population size. The mutation value was set to 0.001 based on [19].

3.2 *Structure of EQLA*

Although TSP is an optimization problem, it is converted into an RL problem by mapping the elements in the problem to the parameters involved in the algorithm. The cities in the problem are denoted as states and, the agent at each state can perform certain actions which are represented in the Q-Table as state-action pairs. In each state, the agent chooses any of the possible actions and receives a reward according to that action. The reward is the negative of the distance between the current state and the next state. The main aim of the agent is to maximize the reward obtained. So, maximizing the reward will naturally decrease the distance and hence solve the TSP. Since Q-Learning is a model-free RL algorithm, it works on the given toy problem through a trial-and-error approach. The working of the algorithm is depicted below.

Q-Table Generation
Input: alpha: learning rate, gamma: discount factor, epsilon: random number
Output: A Q-Table containing $Q(S, A)$ pairs defining estimated optimal policy

```
1. Initialize $Q(s, a)$ table arbitrarily
/*For each step, calculate Q-value and update the Q-Table */
2. For each step:
2.1 Initialize state $S$; /* Initialize current city $S$ /*
2.2 For each step in iteration do
      Do
       /* Choose the next city from the current city using the epsilon-greedy policy */
       2.2.1 $A \leftarrow$ Select action(Q, S, epsilon);
       /* Take action $A$, then observe reward $R$ and next state $S$; */
       2.2.3 $Q(s, a) = Q(s, a) +$ learning rate*[$R$ + gamma*max $Q(s', a) - Q(s, a)$];
       2.2.4 $S = S'$;
     while $s$ is not terminal;
     end
end
```

Epsilon-Greedy Action Selection

```
Input: Q-Table generated so far, S current state
Output: Best route
Function Q-Learning (Q, S, epsilon)
1. $N \leftarrow$ uniform random number between 0 and 1;
2. If $N <$ epsilon then
          2.1 $A \leftarrow$ random action /* update Q-Table if best found so far is improved */
          else
          2.2 $A \leftarrow$ current best action (best route achieved)
return selected action $A$;
```

The EQLA consists of three parameters - *Alpha*, *Gamma*, and *Epsilon*. *Alpha* is the learning rate and is a real-valued variable in the range of [0,1]. The learning rate determines the extent of updating the Q-Table. Here, 0 makes the model not learn anything and 1 makes the model learn only from the most recent action. *Gamma* or discount factor is a parameter that determines the importance of the reward provided in each epoch, valued in the range of [0,1]. A *Gamma* value of 0 makes the model focus only on short-term rewards and 1 makes the model aim for high rewards in the long run. For indeterministic problems such as TSP, without a terminal state, having a *Gamma* value of 1 may lead to infinite reward and the solution might not converge.

Epsilon hyperparameter is based on the randomness proposed to the model where it explores or exploits based on the value. When the *Epsilon* value is equal to 0, the agent uses the previously obtained knowledge and when the value is equal to 1, the model explores and then arrives at a solution. So, the *Epsilon* is chosen (close to zero) so it never settles with the local optimal solution.

The first step of the algorithm is to initialize the parameters. The Q-Learning algorithm uses a state-action table called Q-Table. The Q-Table contains all the possible states and the corresponding actions possible at each state. At each epoch, the agent is at a certain state and the possible actions that can be performed from that state are given by the Q-Table. The Q-Table is initialized to a 2-dimensional matrix of dimensions $N \times N$ where N represents the number of cities in the defined TSP. In each epoch, the agent selects an action from the available set of actions based on the epsilon-greedy approach.

In the epsilon-greedy algorithm, there is an exploration vs exploitation trade-off. The agent can either perform exploration and aim for higher rewards as time progresses or can exploit based on the knowledge gained and can aim for short-term rewards. This trade-off can be addressed using the *Epsilon* value and is demonstrated in Figure 4.

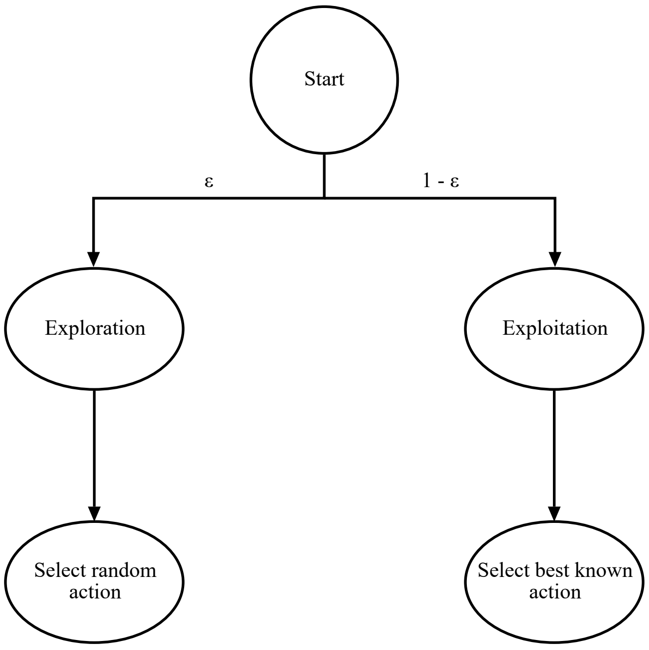

Figure 4. Exploration-exploitation trade-off

The parameter *Epsilon* is used to overcome the exploitation vs. exploration trade-off. The epsilon-greedy approach takes a random value between 0 and 1. If the generated value is less than *Epsilon*, then the model performs exploration. If the generated value is greater than *Epsilon*, then the agent performs exploitation. This algorithm returns the best action that can be performed at a given stage. After attaining the best possible action, the Q-Learning algorithm performs an update of the Q-Table. The updating of the Q-Table is based on the Bellman equation which is given by Equation (1), where the variables S_t and A_t refer to current state and current action respectively and Q (S_t, A_t) represents the Q-Table corresponding to state-action pair for state S_t and action A_t. Other parameters such as α denote the learning rate, γ refers to the importance of reward, and R is the reward for performing action A_t form state S_t.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \, [R_{t+1} + \gamma \max Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{1}$$

The Bellman Equation takes into account the current value from the Q-Table and increments it with a value determined by the parameters of the model such as learning rate (α) and discount factor (γ). After each epoch, the distance is calculated and stored in an array. The best of the distances is obtained from the array after the completion of the pre-set number of epochs.

4. Results and discussions

In the view of comparing and evaluating the algorithms extensively, two data sets were considered. One is a randomly generated x and y coordinates and the other consisting a set of 48 state capitals of the United States (*ATT48* – obtained from Traveling Salesman Problem Library (TSPLIB) [25]) and is visualized in Figure 5.

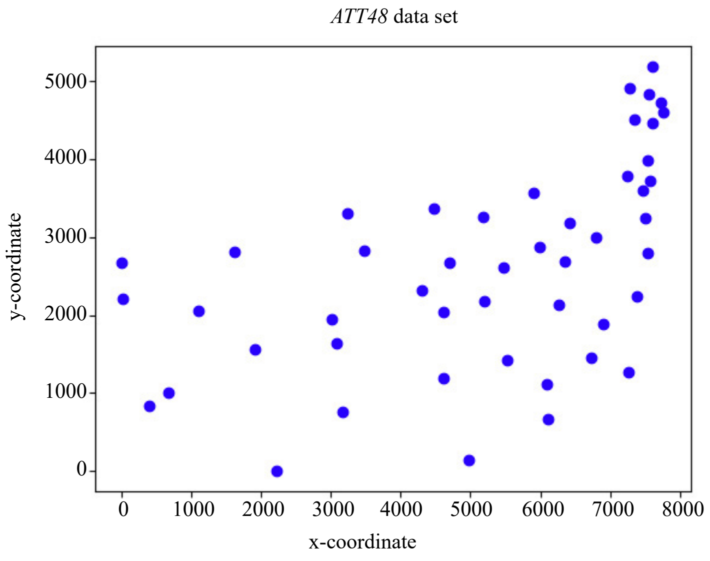

Figure 5. Visualization of *ATT48* data set

GA initially tries to find the route with the highest fitness value for each generation from a randomly generated population and then improvises the search based on elitism, mutation, and crossover. A possible error of converging in local minima is avoided by this method. However, the algorithm is run for about 10 iterations and the average of all iterations is considered as the obtained solution for a fair comparison with the Q-Learning technique.

The distance vs. generation graph for the last iteration showcases the development in search and fitness values of the algorithm over generations. The results obtained from data sets using GA are tabulated and graphed. The results (the initial distance and optimized distance) for the randomly generated data set are presented in Table 1. For a sample run, over 500 generations, the convergence of the shortest path distance is visualized in Figure 6. Similarly, the results and the convergence graphs obtained for the *ATT48* data set are presented in Table 2 and Figure 7, respectively.

Table 1. Results of GA solving TSP with a random data set

Iteration	Initial distance	Optimized distance
1	1,456.13	551.99
2	1,456.13	513.84
3	1,456.13	513.84
4	1,456.13	513.84
5	1,456.13	513.84
6	1,456.13	513.04
7	1,456.13	503.77
8	1,456.13	503.77
9	1,456.13	496.53
10	1,456.13	496.53
Average value	**1,456.13**	**512.1**

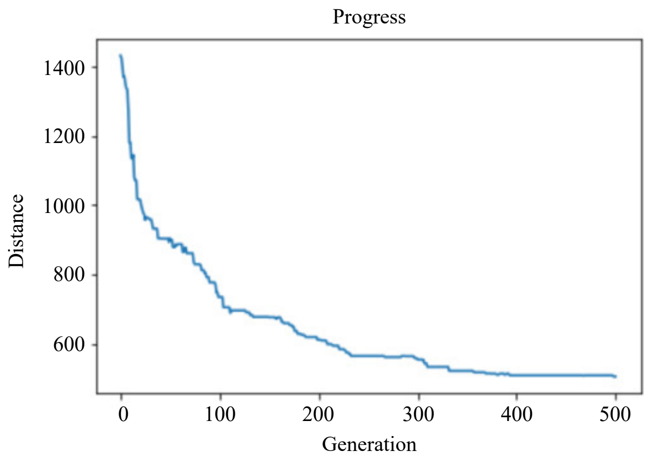

Figure 6. Convergence graph of GA for TSP with a random data set

Table 2. Results of GA solving TSP with the *ATT48* data set

Iteration	Initial distance	Optimized distance
1	129,870.83	37,136.8
2	129,870.83	37,136.8
3	129,870.83	37,136.8
4	129,870.83	37,136.8
5	129,870.83	37,136.8
6	129,870.83	37,136.8
7	129,870.83	37,136.8
8	129,870.83	37,136.8
9	129,870.83	37,136.8
10	129,870.83	36,535.61
Average value	**129,870.83**	**37,076.68**

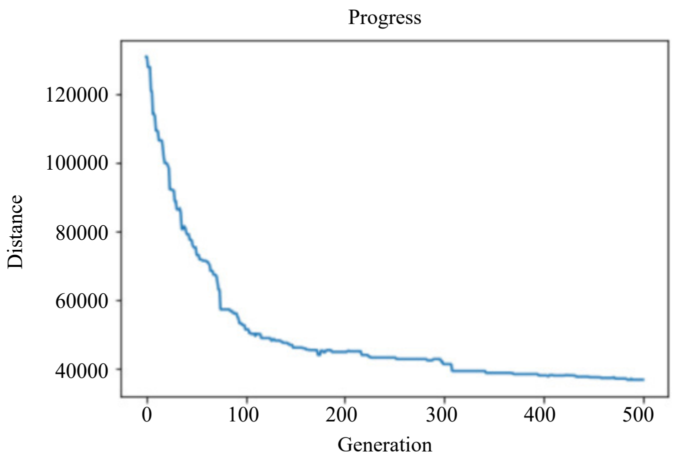

Figure 7. Convergence graph of GA for TSP with *ATT48* data set

It is observed from the experimental results that the GA performs well and improves the solutions in every generation. The time taken to find the solution in each generation is found to be minimal and the algorithm also manages to find solutions all over the search space. The implemented EQLA, initially, tries to obtain the Q-Table with the best routes to the cities (minimum distance). The concept of RL is that the agent tries to discover its environment which purely depends on trial and error. To overcome the exploration-exploitation trade-off, a variety of *Epsilon* values are considered for better understanding and to compare the algorithm with GA. Table 3 presents the distance value obtained for each *Epsilon* value considered, for the TSP with a randomly generated data set. Here, the algorithm is run for 10 iterations, and an average of all the iterations is considered as the obtained solution. Figure 8 indicates the reduction in the distance as the number of iterations increases (depicting distance vs. the number of iterations). Similarly, the results obtained by solving the TSP with the *ATT48* data set using the EQLA are presented in Table 4 and Figure 9.

Table 3. Results of GA solving TSP with a random data set

Iteration	***Epsilon* = 0**	***Epsilon* = 0.2**	***Epsilon* = 0.5**
1	551	576	554
2	521	535	575
3	536	568	549
4	540	549	554
5	529	537	577
6	544	535	545
7	519	505	559
8	510	541	538
9	506	539	558
10	532	536	535
Sum	5,288	5,421	5,544
Average	**528.8**	**542.1**	**554.4**

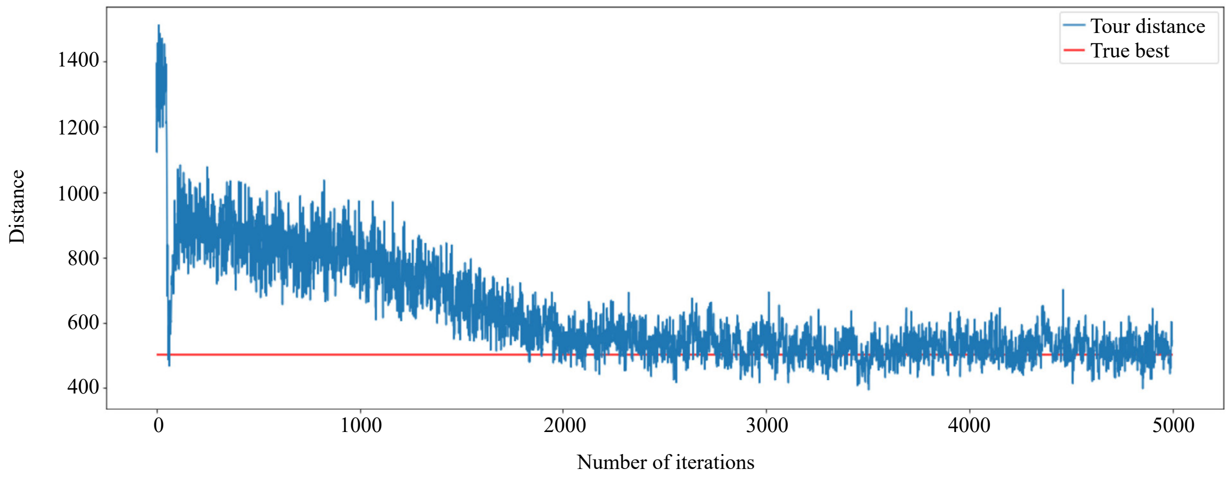

Figure 8. Convergence graph of EQLA for TSP with a random data set

Table 4. Results of EQLA solving TSP with *ATT48* data set

Iteration	*Epsilon* = 0	*Epsilon* = 0.2	*Epsilon* = 0.5
1	41,876	44,768	45,392
2	45,537	46,127	47,634
3	46,678	44,969	49,101
4	41,382	44,707	48,155
5	45,401	45,840	46,887
6	44,727	46,395	42,475
7	41,922	44,325	45,736
8	44,540	46,464	45,830
9	41,693	45,834	44,487
10	43,317	44,719	45,489
Sum	437,073	454,148	461,186
Average	**43,707.3**	**45,414.8**	**46,118.6**

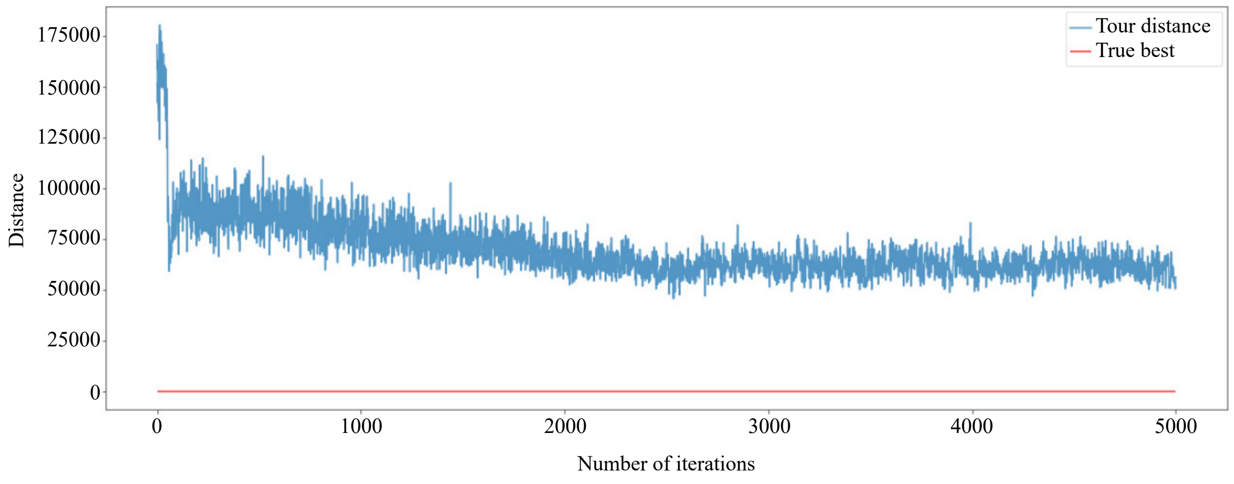

Figure 9. Convergence graph of EQLA for TSP with ATT48 data set

The comparative study carried out on comparing GA and EQLA in solving the classical TSP with two different data sets (*random* and *ATT48*), reveals that EQLA consumes more time and memory in comparison with GA. It is also inferred that the solutions found by EQLA are less accurate and GA finds more optimal solutions.

5. Conclusions

This work is an attempt to compare and understand the working of EC and ML approaches for solving an optimization problem and the following points are considered to be the highlighted contributions of this paper.

- Solving the optimization problem (TSP) through GA, a part of EAs, and obtaining the optimal result.
- Modeling optimization problem as a learning problem and solving with EQLA algorithm.
- Comparing and studying the inferences obtained from the solutions provided by both algorithms.

The GA performs well in terms of fastness in obtaining a quasi-optimal solution whereas the EQLA works by consuming a considerable amount of running time than GA and failing to obtain the optimal solutions near to that of GA. The experimental results obtained along with the inferences gained are presented in this paper. This study observed that GA is an effective approach for solving the GTSP. However, the limitation of GA is that the model might find it difficult to obtain a near-optimal solution when the number of cities in TSP becomes bigger. But, it can still manage to produce outputs in minimum time.

This study can be enhanced by hybridizing these two algorithms (GA and EQLA) to obtain cost-effective optimal solutions by leveraging the benefits of both algorithms.

Conflict of interest

The authors declare that there is no conflict of interest.

References

[1] Allaparthi SR, Jeyakumar G. An investigational study on ensemble learning approaches to solve object detection problems in computer vision. *Mathematical Statistician and Engineering Applications*. 2022; 71(3s): 399-412. https://www.philstat.org/special_issue/index.php/MSEA/article/view/219

[2] Aravind T, Reddy BS, Avinash S, Jeyakumar G. A comparative study on machine learning algorithms for predicting the placement information of under graduate students. In: *2019 Third International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. Palladam, India: IEEE; 2019. p.542-546. https://doi.org/10.1109/I-SMAC47947.2019.9032654

[3] Kartik PV, Sumanth KB, Sri Ram VN, Jeyakumar G. Decoding of graphically encoded numerical digits using deep learning and edge detection techniques. *Journal of Intelligent & Fuzzy Systems*. 2021; 41(5): 5367-5374. https://doi.org/10.3233/JIFS-189859

[4] Raj KS, Nishanth M, Jeyakumar G. Design of binary neurons with supervised learning for linearly separable Boolean Operations. In: Smys S, Tavares J, Balas V, Iliyasu A. (eds.) *Computational Vision and Bio-Inspired Computing: ICCVBIC 2019*. Cham: Springer; 2020. p.480-487. https://doi.org/10.1007/978-3-030-37218-7_54

[5] Sudharshan S, Jeyakumar G. Gradient-Based Versus Gradient-Free Algorithms for Reinforcement Learning. In: Tiwari A, Ahuja K, Yadav A, Bansal JC, Deep K, Nagar AK. (eds.) *Soft Computing for Problem Solving.* Advances in Intelligent Systems and Computing, vol 1392. Singapore: Springer; 2021. p.115-124. https://doi.org/10.1007/978-981-16-2709-5_10

[6] Pragadeesh C, Jeyaraj R, Siranjeevi K, Abishek R, Jeyakumar G. Hybrid feature selection using micro genetic algorithm on microarray gene expression data. *Journal of Intelligent & Fuzzy Systems*. 2019; 36(3): 2241-2246. https://doi.org/10.3233/JIFS-169935

[7] Abhishek S, Emmanuel SC, Rajeshwar G, Jeyakumar G. A genetic algorithm based system with different crossover operators for solving the course allocation problem of universities. In: Smys S, Iliyasu AM, Bestak R, Shi F. (eds.) *New Trends in Computational Vision and Bio-inspired Computing. ICCVBIC 2018*. Cham: Springer; 2020. p.149-160. https://doi.org/10.1007/978-3-030-41862-5_14

[8] Saketh KH, Jeyakumar G. Comparison of dynamic programming and genetic algorithm approaches for solving subset sum problems. In: Smys S, Tavares J, Balas V, Iliyasu A. (eds.) *Computational Vision and Bio-Inspired Computing. ICCVBIC 2019*. Advances in Intelligent Systems and Computing, vol 1108. Cham: Springer; 2020. p.472-479. https://doi.org/10.1007/978-3-030-37218-7_53

[9] Jafarian J. An experiment to study wandering salesman applicability on solving the travelling salesman problem based on genetic algorithm. In: *2010 International Conference on Educational and Information Technology*. Chongqing, China: IEEE; 2010. p.V1-183-V1-189. https://doi.org/10.1109/ICEIT.2010.5607762

[10] Dwivedi V, Chauhan T, Saxena S, Agrawal P. Travelling salesman problem using genetic algorithm. *IJCA Proceedings on Development of Reliable Information Systems, Techniques and Related Issues (DRISTI 2012).* 2012; 1: 25-30. https://www.ijcaonline.org/proceedings/dristi/number1/5926-1007

[11] Xu J, Pei L, Zhu RZ. Application of a genetic algorithm with random crossover and dynamic mutation on the travelling salesman problem. *Procedia Computer Science.* 2018; 131: 937-945. https://doi.org/10.1016/j.procs.2018.04.230

[12] Juneja SS, Saraswat P, Singh K, Sharma J, Majumdar R, Chowdhary S. Travelling salesman problem optimization using genetic algorithm. In: *2019 Amity International Conference on Artificial Intelligence (AICAI).* Dubai, United Arab Emirates: IEEE; 2019. p.264-268. https://doi.org/10.1109/AICAI.2019.8701246

[13] Pullan W. Adapting the genetic algorithm to the travelling salesman problem. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC'03*. Canberra, Australia: IEEE; 2003. p.1029-1035, vol 2. https://doi.org/10.1109/CEC.2003.1299781

[14] Matei O, Pop P. An efficient genetic algorithm for solving the generalized traveling salesman problem. In: *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*. Cluj-Napoca, Romania: IEEE; 2010 p.87-92. https://doi.org/10.1109/ICCP.2010.5606458

[15] Chaves AA, Lorena LH. An adaptive and near parameter-free BRKGA using Q-Learning method. In: *2021 IEEE Congress on Evolutionary Computation (CEC)*. Kraków, Poland: IEEE; 2021. p.2331-2338. https://doi.org/10.1109/CEC45853.2021.9504766

[16] Karimi-Mamaghan M, Pasdeloup B, Mohammadi M, Meyer P. A learning-based iterated local search algorithm for solving the traveling salesman problem. In: Dorronsoro B, Amodeo L, Pavone M, Ruiz P. (eds.) *Optimization and Learning. OLA 2021*. Communications in Computer and Information Science, vol 1443. Cham: Springer; 2021. p.45-61. https://doi.org/10.1007/978-3-030-85672-4_4

[17] Ottoni AL, Nepomuceno EG, Oliveira MS, Oliveira DC. Reinforcement learning for the traveling salesman problem with refueling. *Complex & Intelligent Systems.* 2022; 8(3): 2001-2015. https://doi.org/10.1007/s40747-021-00444-4

[18] Pettinger JE, Everson RM. Controlling genetic algorithms with reinforcement learning. In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. San Francisco, CA, United States: Morgan Kaufmann Publishers Inc; 2002. p.692. https://dl.acm.org/doi/abs/10.5555/2955491.2955607

[19] Piszcz A, Soule T. Genetic Programming: Analysis of Optimal Mutation Rates in a Problem with Varying Difficulty. In: *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*. Florida, USA: American Association for Artificial Intelligence; 2006. p.451-456.

[20] Hu Y, Yao Y, Lee WS. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. *Knowledge-Based Systems.* 2020; 204: 106244. https://doi.org/10.1016/j.knosys.2020.106244

[21] Tatavarthy SR, Sampangi G. Solving a reverse supply chain TSP by genetic algorithm. *Applied Mechanics and Materials.* 2015; 813-814: 1203-1207. https://doi.org/10.4028/www.scientific.net/AMM.813-814.1203

[22] Rao TS. A comparative evaluation of GA and SA TSP in a supply chain network. *Materials Today: Proceedings.* 2017; 4(2): 2263-2268. https://doi.org/10.1016/j.matpr.2017.02.074

[23] Rao TS. An ant colony TSP to evaluate the performance of supply chain network. *Materials Today: Proceedings.* 2018; 5(5): 13177-13180. https://doi.org/10.1016/j.matpr.2018.02.308

[24] Mythili V, Kaliyappan M, Hariharan S, Dhanasekar S. A new approach for solving travelling salesman problem with fuzzy numbers using dynamic programming. *International Journal of Mechanical Engineering and Technology.* 2018; 9(11): 954-966. https://iaeme.com/Home/article_id/IJMET_09_11_097

[25] Reinelt G. TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing*. 1991; 3(4): 376-384. https://doi.org/10.1287/ijoc.3.4.376