International Journal of Control Theory and Applications

ISSN : 0974–5572

Volume 9 • Number 42 • 2016

A Study of Approaches to Solve Traveling Salesman Problem using Machine Learning

Manuj Aggarwal[a] Deepak Sharma[a] Mohd Nizam[a] and Naveen Yadav[a]

[a]Department of Computer Science, ARSD College, University of Delhi, India

E-mail: mmanuj.aggarwal@gmail.com, dsharma080@gmail.com, alinizam72@gmail.com, yadavnaveen357@gmail.com

Abstract: In Traveling Salesman Problem, a collection of cities and the cost of travel between each pair are given and the task is to find the cheapest way to visit all the cities exactly once and return to the starting city. For Traveling Salesman Problem, several methods have been suggested in literature that can find an optimal or near optimal solution. However, most of the traditional methods are lengthy and with the rise of machine learning, many techniques emerged out that gave near optimal solution. In this paper, two of the most available machine learning techniques: Q learning and Hopfield Neural Networks to solve traveling salesman problem have been studied. Comparative analysis of the techniques is done. It was observed that both the machine learning techniques produced near optimal results for problems of different size. As a part of future work, we will attempt to combine Q learning techniques with other metaheurisitic techniques such as Genetic Algorithms in order to obtain near optimal solutions.

Keyword: *Traveling Salesman Problem, Q Learning, Hopfield networks, Machine learning, metaheuristic optimization problem.*

1. INTRODUCTION

The Traveling Salesman Problem (TSP) is a classic optimization problem in the field of computer science. TSP describes a salesman who must travel through N cities. The order of visiting the cities is not important, as long as he is able to visit each city exactly once and comes back to starting city [1, 2]. Each city is connected to other city through some link. Each of the link between cities is weighted. The weight can be anything such as distance or cost of traveling to the connected city. The salesman wants to keep the distance and cost of travel as low as possible. In terms of graph theory, TSP can be understood as to find the shortest possible Hamiltonian Cycle in a graph, in which nodes of the graph represent cities and edges represent a path from one city to another [1].

TSP is easy to understand but difficult to solve. If there is way to divide the problem into smaller sub-problems then each of the sub-problems is as difficult to solve as the original problem. It has been studied for decades and yet no general solution has been found. In literature, TSP has been proved to fall in the category of NP-Complete problems [1, 3]. Fig. 1 shows an illustration of TSP.

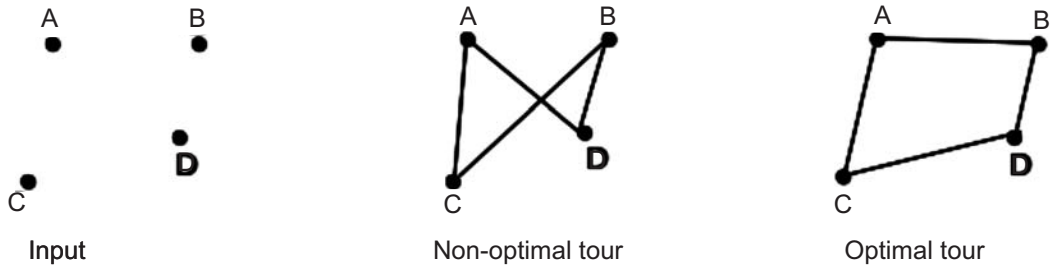

Figure 1: Example of TSP

TSP is solvable, that is, an algorithm can be designed that can find correct answer every time. However, such an algorithm looks at all the possible $n!$ ways a circuit can be constructed with all the nodes of the graph. As a result, such algorithm essentially takes $O(n!)$ time and therefore requires large amount of time for datasets of large size [4, 5].

There are two types of methods available to solve TSP, one which gives exact solutions such as cutting plane method and held-karp algorithm, and the other which use heuristics to approximate the solution such as nearest neighbor and many of the machine learning techniques use these approximations [1]. Finding exact solutions is time consuming and for very large problem set may take too long which makes the solution unproductive. On the other hand, heuristic solutions give a good approximate to the problem in reasonable amount of time. Of course, the solution may not be the best but the solution is obtained in less time. In real life, it is desirable to obtain a near optimal solution in less time than to obtain an optimal solution in near infinite time.

In this paper we have investigated different techniques to solve TSP. In literature, Machine Learning techniques such as Q learning and Hopfield neural networks have been extensively used. We have performed a comparative analysis of these techniques used for solving TSP.

The rest of the paper is organized as follows: Section II presents brief introduction of machine learning and its two techniques namely Q learning and Hopfield Neural networks. Section III presents Q learning methods for solving TSP and a comparison is done. Section IV presents Hopfield Neural Networks techniques to solve TSP and section V concludes the paper and presents scope for future work.

2. MACHINE LEARNING

Machine learning gives computers the ability to learn without being explicitly programmed. It evolves from the study of recognizing patterns and computational learning [6]. It involves constructing agents (algorithms) that learn and generate predictions, by synthesizing a model from sample inputs. It is useful to provide a solution to tasks where designing and programming exact algorithms is impractical or in cases where a suboptimal but efficient solutions may be desirable, as is the case for solving the Traveling Salesman Problem. Machine learning techniques that can be used to solve TSP include:

1. Q Learning
2. Hopfield Networks
3. Ant Colony System
4. Genetic Algorithms

In this paper, we have examined two Machine Learning approaches, Q-learning and Hopfield Neural Networks that solve the TSP. These approaches aim to create intelligent agents that give the best way of action, or a good approximate tour after learning the path it understands as optimal. Next, we provide brief overview of both the techniques.

2.1. Q Learning

Agent learns how to deal with a problem it has encountered with the dynamic environment through the trial-and-error interactions in reinforcement learning. For a particular action taken from a given state, the agent gets a numerical reward value for that action and changes its state. Q-learning can be viewed as an asynchronous dynamic programming method [6, 7]. It is a form of model-free reinforcement learning. It is model free as it does not require model the environment an agent will be placed in. Fig. 2 illustrates block diagram of Q learning.

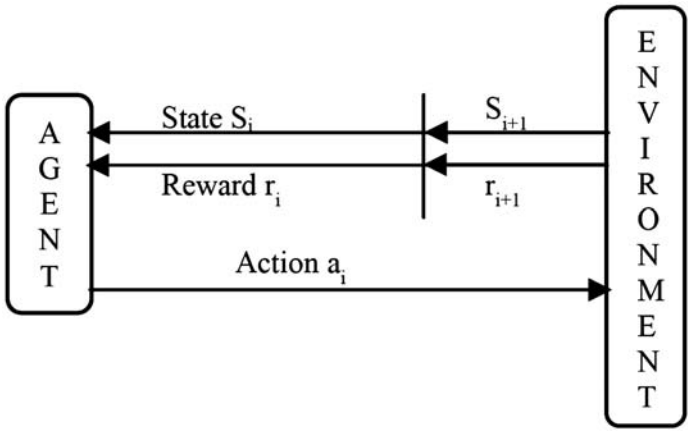

Figure 2:. Block diagram of Q learning

The problem model consists of an agent, a set of states S and a set of actions per state that an agent can take when it is at that particular state. At a particular state, an agent tries an action, and evaluates the immediate reward or penalty it receives. The goal of the agent is to maximize total reward. It can be achieved by learning the optimal action for each state. The agent maintains a Q table. The Q-values Q(*s*, *a*) in Q-table reflect the observations [6]. Each Q-value is initialized at the beginning. The choice of the next state to move from a given state is always based on some predefined policy. The algorithm is given in fig. 3.

Initialize Q(*s*, *a*) arbitrarily
For each episode, repeat the following steps
Initialize *s*
Repeat until final state reached
Choose a from s using an exploratory policy
Take action a, and observe its reward *r*, and new state *s*'
Update Q(*s*, *a*) using below formula
$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_a Q(s', a') - Q(s, a)]$
$s \leftarrow s'$
where α is learning rate and γ is discounted reward rate.

Figure 3: Algorithm for Q learning

2.2. Hopfield neural networks

Neural networks are an approach in computation wherein a large number of states model a biological brain, which is able to provide solutions to problem, using large clusters of neurons. Each such neuron (also referred as state) is connected with many others in the network. Links between nodes can inhibit or strengthen the state values of such connected neurons [8].

The structure of a Hopfield Neural network consists of a finite number of units with each node capable of storing a binary threshold values. For each pair of unit *i* and *j*, a mathematical function $W_{i,j}$ called connectivity weight is defined [9]. Fig. 4 shows an example of Hopfield Network for a set of 4 cities.

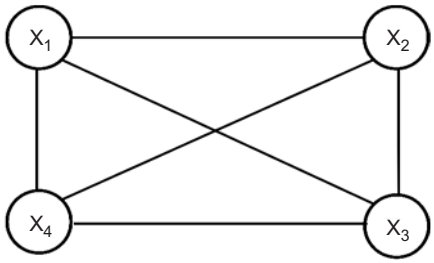

Figure 4: Hopfield network of 4 cities

Apart from the initial value of a node, the same can be updated by using the following:

$$S_i = \begin{cases} 1 & \text{if } \sum_{j \text{ is connected to } i} W_{i,j} * S_i > \text{Threshold} \\ -1 & \text{else} \end{cases}$$

Energy function of the Hopfield neural network is defined such that its value either lowers or remains same during an update step. Such process can be inferred as minimization of the energy function, so as to achieve a stable state. Since such a function can have many local minima, convergence may lead to a suboptimal solution.

For solving optimization problem with Hopfield Neural Network, the following steps are considered:

1. The problem is represented in the structure of the Hopfield network.
2. Energy function for the network is defined.
3. Connectivity weight function for network including the problem constraints is defined.
4. Binary threshold values for all nodes are initialized.
5. Nodes (randomly or sequentially) are updated until stable state is reached.

3. METHODS OF SOLVING TSP USING Q LEARNING

In this section, different approaches of Q- learning methods for solving TSP are discussed.

Erban and Pintea [3] have proposed several heuristics methods such as Nearest Neighbor and Ant colony system for solving the TSP, along with Reinforcement Q-Learning method. The authors have made the following assumptions:

1. The N cities for the tour form the N Q-states, with each state having actions to move to the other N-1 states.
2. A state is considered a final state if it has been reached after agent has visited the other N-1 nodes before it.
3. The reward function for the Q-learning method is defined as the Euclidean distance between the considered cities.

The learned optimal policy, *i.e.* the optimal tour is achieved by using a sufficient number of iterations of algorithm, called episodes. An episode ends when a tour is completed. The algorithm runs in O(N * E) time, where N is the number of cities and E is the number of episodes.

By applying the prescribed algorithm to several examples of TSP in Cartesian plane, the authors have found that the algorithm, in average, gives better results than Neural Network approach.

Uslan and Bucak [4] have compared various machine learning techniques to solve TSP and proposed a Q learning method which combines basic Q learning technique with 2-opt localization as the last step. The learning agent manages two tables one Q table which reflects observations for finding shortest path and other a reward table which is modified whenever agent finds the shortest path on repetition and other a reward table.

Each $r(c_i, c_j)$ pair is modified when the agent finds the shortest path during the repetition. This is where the reinforcement learning takes place and better solutions get a higher reinforcement. All reward values are modified by a discount rate except the pair values of the recent found shortest path. Next, all $Q(c_i, c_j)$ pairs are initialized and pairs of the best tour are given a numerical reward to reflect a fresh observation. Then, instant Q-values are calculated with those recently modified $r(c_i, c_j)$ pairs. After some predefined criterion is reached the 2-opt localization is applied to the solution produce.

$$\text{Q function used was } Q^*(s, a) = r + \gamma.\max_{a'} Q(s', a')$$

where, (s, a) is state action pair

r is the immediate reward received

γ is learning rate

The idea is to take a route that crosses over itself and reorder it in such a way that it does not.

2-opt improvements were implemented by removing two edges each time and reconnecting them such that $d(c1, c2) + d(c3, c4) > d(c1, c3) + d(c2, c4)$. Fig. 5 illustrates 2-opt localization.

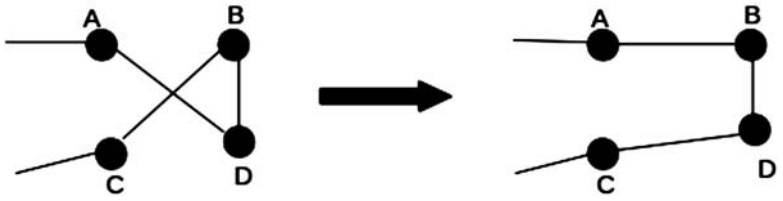

Figure 5: Example of 2-opt localization

The algorithm for the above approach is given in fig. 6.

1. Initialize Q-table and reward table for each pair of cities (c_i, c_j) of distinct cities.
2. Use greedy ε-greedy approach for choosing next city to move.
3. Receive an immediate reward $r(c_i, c_j)$ and adjust Q-table.
4. If shortest path so far is found update rewards and initialize the Q-table by giving numerical reward to each $r(c_i, c_j)$ belonging to shortest route.
5. Repeat steps 2 to 4 until stopping condition is not satisfied.
6. Apply 2-opt localization. Using Q values select global minimum tour.

Figure 6: Algorithm for Q learning [4]

The intent was to decrease the number of intersections as much as possible, 2-opt localization has been performed at the end of the constructed tour from the Q-values. Depending on the solution obtained, the number of replacements varies. It was observed that Q- learning method slowly converged to optimal or near optimal value. As the city size grew, the solution obtained started to diverge from the optimal one. The performance and speed was sufficient for the problems less than 200 cities and could find near optimal solutions. For problems size greater than 200 cities the author have suggested 3-opt localization.

Dorigo and Gambardella [10, 11] have proposed Ant-Q algorithm which is combination of Q learning algorithm and ant colonies behavior. Ant Q algorithm is divided into three phases which includes initialization phase, building the tour and finally ants update AQ values globally and ultimately find the optimal solution. In the initialization phase starting city for ants is chosen. In the second phase ants build tours by applying state transition rules and locally updating AQ values. In this phase ants choose the city where they will go. In the final phase ants update the edges belonging to the best tour done and finally updating the AQ values globally.

Ant Q is a set of simple agents called ants and Ant Q algorithm belongs to ant colony methods. These ants cooperate to find optimal solution. In Q learning only one agent explores the state space whereas in Ant Q cooperating agents are used to explore the state space. AQ values between the agents are exchanged. When compared to other algorithms for solving traveling salesman problem, Ant-Q algorithm was found to produce better results in most cases. Ant-Q was almost always best performing algorithm.

Ant Q outperforms the other algorithm of same category such as heuristic algorithms, simulated annealing, elastic net, self-organizing map, farthest insertion etc. The solutions were also locally optimal with respect to 2 opt and 3 opt heuristics. However, the complexity of Ant Q iteration ($O(m.n^2)$) where m is the number of ant agents and n is the number of cities. Due to this, its application becomes infeasible to big TSP problems. Table 1 shows the comparison among Q learning approaches.

Table 1
Comparison of Q learning approaches

Property	*First method [3]*	*2-opt method [4]*	*Ant-Q method [10, 11]*
Graph	Complete	Complete	Complete
Agent System	Single agent	Single agent	Multi agent
Reward Mechanism	Static	Dynamic	Static
Action Selection	Epsilon Greedy	Epsilon Greedy	Pseudo random
Optimization	None	2-opt	None
Time complexity	O(N E)	O(N^2 E)	O(M N^2 E)
Performance	Good	Better	Best

On the basis of comparison of Q learning approaches to solve TSP, it can be concluded that all three approaches work on complete graphs, and have randomness to decision making procedure in the algorithms. They differ in reward mechanism and optimization used in the process. A straightforward pattern of trade-off between performance and efficiency is observed among the three methods. Whereas the first approach is good for large inputs, the best performance is achieved by the multi agent Ant-Q method for comparatively smaller inputs.

4. TSP USING HOPFIELD NEURAL NETWORK

Various methods to solve TSP using neural network are discussed here.

Moetty [5] has compared shortest path nearest neighbor and Ant Colony System along with neural networks system for solving TSP.

For using Hopfield neural networks for solving TSP, the energy function should have some necessary properties as given below:

1. It should lead to a stable combination matrix.
2. It should lead to the shortest traveling path.

The energy function used by the author was :

$$E = A_1 \Sigma_x \Sigma_k \Sigma_{j \neq k} V_{x,k} V_{x,j} + A_2 \Sigma_i \Sigma_x \Sigma_{j \neq x} V_{x,i} V_{j,i} + A_3 [(\Sigma_x \Sigma_i V_{x,i}) - N]^2 + A_4 \Sigma_x \Sigma_{j \neq x} \Sigma_i d_{x,j} V_{x,i} (V_{j,i+1} + V_{j,i-1})$$

where A_1, A_2, A_3 and A_4 are positive integer constants, and for the performance of Hopfield network, the setting of these constants are very critical. N is the number of cities to be visited in the tour. Let V_{xi} denote the output of the i^{th} neuron in the array of neurons corresponding to the x^{th} city. The variable V_{xi} denotes the fact that city x is the i^{th}- city visited in a tour. There are N^2 such variable and at the end, their value will be 0 or 1 or very close to 0 or 1.

Using energy function the weight matrix was set up as follows:

$$Wxi, yj = -A_1 \delta xy(1 - \delta ij) - A_2 \delta ij(1 - \delta xy) - A_3 - A_4 dxy(\delta j, i+1 + \delta j, i-1)$$

[$\delta ij = 1$ if $i = j$ and is 0 otherwise]

The algorithm for finding a solution to TSP using Neural Networks is given in fig. 7.

Input the network parameters x, y co-ordinates for each city and compute the distance between all pairs of cities.

/* initialize the network input according to the following equation */

$$u_{xi}(0) = u_{00} + (2 * \text{rand} - 1) / (10 * u_0)$$

/* this is the phase in which neurons are updated using following equation */

$$u_{xi}(t+1) = u_{xi}(t) + dt\ (du_{xi}/dt)$$

Running mode /* by using equation*/

/* in this phase calculate neuron output according to equation*/

$$V_{xi} = (1 + \tanh(u_{xi} / u_0)) / 2$$

Compute the tour length

If max_iterations is reached and tour length unchanged for certain number of loop

Then

Print tour length

Else, perform next iteration

Figure 7: Hopfield network of 3 cities [5]

The Experimental results found that neural networks were fast as compared to standard techniques. Also it gave less or same values for shortest path. On changing the starting city, the solution produced also changed as reported by [12]. Also [12] observed that on increasing number of cities in the problem, the number of iterations required for finding the solution increased non linearly. Also in some cases, the network failed to converge. Hopfield neural networks tend to get captured in local minima.

Feng and Douligeris [13] proposed an optimization of Hopfield Networks to solve TSP by using stable state analysis technique. Stable state analysis is based on determining mutual relations of parameters in energy function by looking at some special cases when network reaches final stable state. Stable state analysis classifies the problem into Constraint terms and Objective Functions. Hopfield energy function is a weighted sum of constraint terms and objective function. Constraint terms can be divided into 3 categories:-Zero constraint term, non-zero constraint term, hybrid constraint term. If the energy function can be expressed in terms of these terms and sufficient number of other conditions is met, then the problem is 1D constrained problem and otherwise it is 2D constrained problem.

Stable state analysis can be utilized to deal with almost all 1D constrained problem and most of the 2D constrained problems. After obtaining the constraints we use them to set appropriate values to the parameters and finally obtaining valid solutions. This algorithm works better than Aiyer method [14], the mean error of the solutions are compared in the two algorithms and stable state analysis gives better mean error rate as compared to Aiyer method.

Experiments performed on 10 city and 51 city Traveling salesman problem states that on 10-city TSP the method yields results which were comparable to results obtained using Simulated Annealing and the mean error of final solutions to a 51-city TSP is better than the optimal tour.

5. CONCLUSION

In this paper, we have performed comparison of two relatively good machine learning approaches, Q learning and Hopfield Neural Networks, for solving the Traveling Salesman Problem. Q learning methods have been found to give better results than heuristics methods such as Nearest Neighbors method or Ant Colony Systems. Optimization techniques such as 2-opt technique greatly improve their performance. Ant-Q systems turn out to give best results, although being relatively infeasible for larger problem size. Q-learning methods though are difficult in terms of setting and finding the optimal parameters in the Q-learning technique. On the other hand, Hopfield neural network methods have been shown to be excellent for smaller problem sizes, providing a relatively smaller bound on worst deviation from optimal solutions. As a part of future work, other optimization techniques such as 3-opt techniques, improving the learning rate employed in Q learning methods can be applied and combining such approaches in order to obtain more accurate solutions of Traveling Salesman Problem.

REFERENCES

[1] G. Reinelt, *The traveling salesman: computational solutions for TSP applications*: Springer-Verlag, 1994.

[2] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The traveling salesman problem: a computational study*: Princeton university press, 2011.

[3] C.-M. P. G. Erban and C.-M. PINTEA, "Heuristics and Learning Approaches for Solving The Travelling Salesman Problem," ed.

[4] V. Uslan and İ. Ö. Bucak, "A Comparative Study of Machine Learning Heuristic Algorithms to Solve the Traveling Salesman Problem."

[5] S. Abdel-Moetty, "Traveling salesman problem using neural network techniques," in *Informatics and Systems (INFOS), 2010 The 7th International Conference on*, 2010, pp. 1-6.

[6] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning,* vol. 8, pp. 279-292, 1992.

[7] J. Peng and R. J. Williams, "Incremental multi-step Q-learning," *Machine learning,* vol. 22, pp. 283-290, 1996.

[8] K. Y. Lee, A. Sode-Yome, and J. H. Park, "Adaptive Hopfield neural networks for economic load dispatch," *IEEE Transactions on Power Systems,* vol. 13, pp. 519-526, 1998.

[9] B. N. K. L. Ding, "Neural network fundamentals with graphs, algorithms and applications," *Mac Graw-Hill,* 1996.

[10] M. Dorigo and L. Gambardella, "Ant-Q: A reinforcement learning approach to the traveling salesman problem," in *Proceedings of ML-95, Twelfth Intern. Conf. on Machine Learning*, 2016, pp. 252-260.

[11] M. Dorigo and L. M. Gambardella, "A study of some properties of Ant-Q," in *International Conference on Parallel Problem Solving from Nature*, 1996, pp. 656-665.

[12] D. Graupe and R. Gandhi, "Implementation of Traveling Salesman's Problem using Neural Network," *Final Project Report, ECE,* vol. 559, 2001.

[13] G. Feng and C. Douligeris, "Using Hopfield networks to solve traveling salesman problems based on stable state analysis technique," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, 2000, pp. 521-526.

[14] S. V. Aiyer, M. Niranjan, and F. Fallside, "A theoretical investigation into the performance of the Hopfield model," *IEEE Transactions on Neural Networks,* vol. 1, pp. 204-215, 1990.