University of the Witwatersrand

Machine Learning

The Traveling Salesman

Semester Project Report

Lecturer:
Prof. T. Celik

Author:
Amine Boukrout

Student No. 886515

June 6, 2018

WITS
UNIVERSITY

Contents

1 Introduction

The Traveling Salesman Problem (TSP) is a well-known problem in the Computer Science and Mathematics disciplines. Since the problem was defined in the 1800s (according to Wikipedia) multiple solutions to the TSP has been presented by many researchers over the years, but an exact fixed solution to the problem hasn't been formally presented.

The TSP can be defined by considering a salesman that has to travel N cities, where the order of visiting each city does not matter, as long as each city is visited only once, and ends off with the city that was initially started w ith. An important factor to consider is that all the cities are connected implying that the salesman can get to any city from any city he is situated at the current point in time, however there is a cost associated with traversing or traveling from one city to another. The main objective of the TSP is to find an ordering of the cities that will minimize the cost of traveling to each city.

The TSP is best represented as a graph with nodes representing the cities and the edges between the cities represent the transportation between two given cities. Initially there will be an edge between each city and once the TSP has been solved there will be only one edge between two cities in a directed manner. This implies that the final output will be a non-cyclic directed graph and the reason that the graph is non-cyclic is that the condition that a city can only be visited once.

There are many solutions to the TSP such as the brute force approach and the dynamic programming approach. However, in this project, a machine learning approach known as reinforcement learning will be used. The Q-Learning algorithm will be used to solve the TSP.

2 The Q-Learning Algorithm

The Q-Learning Algorithm used for solving the TSP takes a set of fixed parameters which are stated in the article, [1.], used for this project. The parameters are as follows:

- The discount factor γ which determines the values of the future rewards by using $V(s) = r + \gamma V(s')$.
- The learning rate α which estimates the value of each state.
- The number of episodes which states the number times the algorithm runs.

The Q-Learning algorithm used also has some other parameters such as the number of iterations it should run for, number of samples of examining the best path found. The basic steps of the Q-Learning algorithm used for the TSP is outlined as follows:

1. Define the Q table setting all the values to zero.
2. Define an initial state s_0
3. Define the possible actions from s_0 represented as A_0
4. By making use of the $\epsilon-$greedy policy choose an action, see below for details.
5. The next state is defined a_{n+1}.
6. Possible actions from the next state are determined.
7. An update to the Q table is made as follows: $Q(s, a) \leftarrow \gamma \max_{a'} Q(s', a')$.
8. Set the current state to the next state

The implemented Q-learning algorithm also incorporates a reward and punishment function. The agent while exploring the actions that can be taken in the next state gets reward if the accumulated cost of taking action a_i is less than taking action a_j, otherwise it receives a punishment. The reward and punishment values are set to be fixed integers, for example a punishment would result of -50 and +50 as a reward.

3 Learning the Optimal Policy

The TSP may seem like a broad problem to solve, however in actuality it is just a computationally, strategic problem to solve. The reason for this is that the TSP must learn an optimal policy once for a given set of nodes (or cities) with weights associated with travelling to node to node. The only instance when the optimal policy can be possibly altered is if there is a change in a node(s), either a node has been added or removed, or if a weight between two nodes has been changed.

At the start of initialization of the algorithm, the agent has no knowledge of what values or consequences does each action result in, thus this is the main reason why reinforcement learning is an appropriate approach to solve the TSP. At the start, it is recommended that a sense of randomness is introduced to allow the agent to explore its environment, and once it has familiarized itself with the environment it can start learning the optimal policy through the rewards and punishments procedure.

To manage and balance between exploitation and exploration, the default learning policy chosen for the implementation of the TSP is the epsilon greedy policy. The main reason that the epsilon greedy approach has been chosen is that it introduces a sense of random action with a probability of ϵ, where $\epsilon \in [0, 1]$.

4 Performance and Results

The main aim of this project was to develop a prototype that could minimize the cost of visiting each city, thus true optimization of the TSP has not been entirely explored due to factors such as time constraints.

In comparison to other TSP solutions such as the greedy approach and dynamic programming, the Q-Learning algorithm has been tested on multiple examples and has been a proficient algorithm for determining the optimal cost solution in an environment consisting of 3 to 5 nodes, to an overall of just over 12 nodes. However, as expected the more nodes in a graph, the higher the complexity resulting in a slower convergence. Nevertheless, the Q-Learning approach to the TSP will find an optimal solution.

In terms of the performance, different discount factors and learning rates did not impact the final cost of the route traversed, or neither on the convergence of the learning agent.

5 Conclusion

In the many literatures reviewed, the use of reinforcement learning has been encouraged over the other common methods. The advantage of reinforcement is that the agent learns by experience, and thus with TSP it will always try to find the optimum solution, which leads to the disadvantage of TSP solved with reinforcement learn that it is often computationally expensive. Overall, TSP with does find the optimum path and can be improved to find the paths much quicker, such as using a genetic algorithm to find the cities or nodes with familiar characteristics.

References

[1] L. M. Gambardella and M. Dorigo, "Ant-q: A reinforcement learning approach to the traveling salesman problem," in *Machine Learning Proceedings 1995*, pp. 252–260, Elsevier, 1995.

[2] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[3] P. Stefán, L. Monostori, and F. Erdélyi, "Reinforcement learning for solving shortest-path and dynamic scheduling problems," in *Proceedings of the 3 rd International Workshop on Emergent Synthesis, IWES*, vol. 1, pp. 83–88, 2001.

[4] D. Wang and D.-m. Lin, "Monte carlo simplification model for traveling salesman problem," *Applied Mathematics & Information Sciences*, vol. 9, no. 2, p. 721, 2015.