2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT) | 978-1-6654-3559-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/RTEICT52294.2021.9573726

Optimal metro route identification using Q-Learning

Parth Miglani
Department of Virtualization
University of Petroleum and Energy Studies
Dehradun, India
parth.miglani2@gmail.com

Avita Katal
Department of Virtualization
University of Petroleum and Energy Studies
Dehradun, India
avita207@gmail.com

Harsh Upadhyay
Department of Virtualization
University of Petroleum and Energy Studies
Dehradun, India
hupadhyay1005@gmail.com

Jayant Kochar
Department of Virtualization
University of Petroleum and Energy Studies
Dehradun, India
jayantkochar20@gmail.com

***Abstract*— Public transport plays quite an essential role in the life of a common man. In metropolitan cities, people try to avoid road-based public transport due to reasons like increasing road congestion, air pollution, energy, and oil consumption, and most importantly to save time. Metro is the most common means of transportation which connects the whole city and the nearby areas via a train that runs on various routes connecting every station in the city. The aim of this paper is to find an optimal metro route for a passenger. The proposed system finds all the stations in order the passenger needs to visit or pass by to reach his destination station by the shortest route. The system considers the whole Metro Map as a graph where the stations act as nodes and are connected using edges. The system uses the Q-learning algorithm to find the optimal path. Q-learning is a reinforcement learning algorithm that teaches an entity which actions to perform under which conditions. It is a reward-based algorithm, stateless and off-policy and hence learns how much long-term reward it will get for every possible action taken on or from every state. Q-learning is also a feedback-based algorithm, it carries forward the output of the previous step as input to the next step. The proposed technique depicts the optimal metro route found by the algorithm. The proposed system aims to reduce the idle/stationary time by reducing the number of stations traversed while reaching the destination station which in turn reduces the total travel time.**

***Keywords*— Q-Learning, Reinforcement Learning, Agent, Reward, Environment, Shortest Distance**

I. INTRODUCTION

Q-learning is based on model-free Reinforcement Learning (RL). There are two types of Reinforcement Learnings, model-based and model-free. In model-based learning, the agent gets to know about the future reward before taking each step or about what's the next state. If the agent is not able to get the future reward or the next state, then it is model-free. The model-based algorithm uses transition functions unlike model-free. Reinforcement learning is the technique of making optimal decisions using past experiences. Machine Learning models are trained using Reinforcement Learning enabling these models to create a sequence of decisions.

Q-learning is an off-policy algorithm as it learns from actions that are outside the policy. The Q-learning agent doesn't learn in a single step i.e., it learns step-by-step. The agent gets a reward for every action it takes and then it initiates the next action based on the previous action, where it gets the maximum reward. The model gets trained by taking these actions on the basis of the rewards earned. Once the model is fully trained, it can produce results infinitely without training the model repeatedly.

Reinforcement learning model consists of the following components as shown in Fig 1:

1. *Agent* learns about what it has to do in the environment.
2. *Environment* is the programmatic definition of the physical world in which the agent operates and learns about what it has to do.
3. *Reward* is given to the agent by the environment on selecting a particular action. The reward is not always positive, it can also be negative in value that is the penalty given for choosing an undesirable action.
4. *Action* is the collection of actions the agent can perform within the environment.

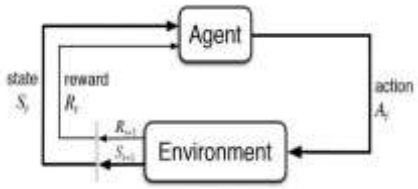

Fig 1: Reinforcement Learning (RL) model

The proposed algorithm in this paper has an edge over the traditional path finding algorithms such as Bellman-Ford, Dijkstra and Floyd-Warshall algorithm. It executes only a single time and produces a fully trained Quality Matrix that will execute infinitely without executing the time-consuming matrix training part repeatedly whereas in traditional path finding algorithms, the whole algorithm executes each time a request is made [1]. Once the matrix is trained, the calculations are done for all the possible paths whereas in the primitive path finding algorithms, calculations are done for the path which is most likely to be taken, rather than for all. This makes the proposed algorithm faster as it does not need to do any other calculations for any

978-1-6654-3559-8/21/$31.00 ©2021 IEEE

other path. Comparing the traditional path finding algorithms and Q-learning algorithms on the basis of time complexity, it is found that Q-learning takes time only for the first iteration after which it takes almost a fraction of the traditional algorithms.

Some popularly known learnings are Deep learning and Reinforcement learning. In deep learning the machine learns its best from that data set and then applies that learning to a new data set created by it [2] whereas in Reinforcement learning, it mainly learns from the feedback or the output of the previous step or state so that with the help of this it can maximize its rewards. Q-learning algorithms are based on Reinforcement Learning [3] making it an off-policy model-free Reinforcement Learning Algorithm.

Reinforcement learning (RL) can be used to solve various complex problems. It can also correct the errors itself that mainly occur in the code or during the runtime [4]. It is used in Machine Learning when there is only one solution or only one way to collect the information for performing the task. It also has some disadvantages as it can affect the results due to the overloading of states in the paths.

II. Related Work

The authors in [5] suggested the Adaptive and Random Exploration (ARE) technique to deal with the aforementioned difficulty in the job of UAV path planning. The main idea behind ARE is to allow the UAV to investigate its surroundings and take actions based on the current evaluation. The suggested technique balances the adaptive mechanisms of self-learning with convergent random search such that the subjective UAV may identify broad routes while also evading the problem caused by learning mistakes.

The authors in [6] provide a method for finding numerous viable routes within obstacle settings using the Q-learning Algorithm. First, the Q-learning algorithm was pre-trained to be suited for path planning. The state action-value function was then used to create an obstacle environment map and a path planning program. To ensure the efficacy of the final routes, all of the function's arguments are made to be constant.

The authors in [7] investigated how to discover the best paths for pedestrians in cities. The objective is to develop optimum rules that provide pedestrians with the quickest paths on streets with crossings and traffic lights. They outlined the problem and demonstrated how model-free Q-learning algorithms may be utilized to solve it. They discovered that, unlike normal reinforcement learning issues, pedestrian navigation is agent-specific, meaning that the optimum policies are determined not just by stochastic processes such as traffic signals, road conditions, and so on, but also by the agents' behaviour.

The authors of [8] developed a parametric Q-learning technique for generating an estimated strategy with a sample size proportional to the feature dimension K and invariant in terms of state space size. They used the Bellman operator's monotonicity property and inherent noise structure.

The authors in [9] have performed the energy transmission route planning using a Geographic Information System (GIS) Multiagent Systems (MAS), a subdomain of Distributed Artificial Intelligence, were used to investigate the multi-criteria influencing energy transmission lines (ETL) many times.

The authors in [10] offer a reinforcement learning (RL) methodology to simulate optimum exercise methods for option-type goods. They used RL to discover the best action-value function for the underlying halting problem. In addition to obtaining the best Q-function at every time step, the contract can be priced from the outset.

The authors in [10] offer a reinforcement learning (RL) methodology to simulate optimum exercise methods for option-type goods. They used RL to discover the best action-value function for the underlying halting problem. In addition to obtaining the best Q-function at every time step, the contract can be priced from the outset.

While travelling in a metro, the distance covered is not the only factor that contributes to the total travel time between two stations. One of the most important factors while travelling in the metro is the stationary/idle time spent by a train when it stops at the intermediate station to board passengers.

The novelty of the proposed system is that it takes into consideration the stationary/idle time and gives the most optimal route between the two stations as the output having the least travel time possible.

III. Problem Statement

Metro is the most common means of transport across many cities in the world. It is quite essential to locate an optimal path for travel to save time. The proposed system eases travel in the metro. It finds the optimal metro route that the passenger needs to follow to reach from one station to another in the minimum time. The system finds the minimum approximate time of travel from one station to another. To find the optimal path the proposed system uses the Q-learning algorithm. Q-learning algorithm is a reward-based algorithm that learns how much reward it will get for each action performed [11] and then choses next action based upon the feedback received from previous action.

IV. Methodology

Q-learning steps outside the current policy, and sometimes takes steps based on random actions. Q-learning considers a single state-ahead and calculates Q-value for every state-action pair [12]. The Q value is directly proportional to the future reward of taking that action.

Q is a state-action value that defines how valuable an action is for the system in a state, looking one step ahead only. The Q values are updated as per the below mentioned equation:

$$Q_{t+1}(s,a) = Q_t(s,a) + \underline{\alpha}(r + \boldsymbol{\gamma} \max Q_t(s', a') - Q_t(s,a)) \quad (1)$$

Where,

Q(s', a') denotes the action-state value of the next possible state, choosing the optimal action a'

r denotes the immediate reinforcement

α denotes the learning rate

γ denotes the discount rate

The following are the prerequisites to find the minimum distance between two stations using Reinforcement Learning:

- values of r, α, γ
- n denotes the number of metro stations
- Two adjacency matrices q[n][n] and r[n][n] where the Q matrix denotes the Q value for all the state-action pairs and r[n][n] denotes the reward for all state-action pairs.

First, the source and the destination node will be defined where nodes will be representing the metro stations. Initially, all the values in the Q matrix are initialized as 0.0 and the reward matrix as -1.0. The reward matrix is defined based on the source and destination nodes.

Q-matrix is trained by providing 500-1000 iterations. In the next step the best possible Q value for every state-action pair is defined. In these iterations, a random row value is selected and with reference to the initial row value, all the column indices whose value is greater than equal to 0 are collected in an array and a random index is chosen from the same. This random index value and chosen row value is taken as a parameter by the final update function, which is actually using equation 1 and updating the Q-matrix.

Algorithm 1: Training the Q-matrix

Input: double qMatrix[11][11], rMatrix[11][11]

Output: Fully trained qMarix[11][11]

Initialization: int available_acts[8], int current_state, double scores[500]

```
1: for i: 0 to n do
2:      current_state= returnRandom()
3:      size_av_actions=
available_actions(current_state,available_acts,rMatrix)
4:      action= sample_next_action(size_av_actions,available_acts)
5:      score= update(current_state, action, rMatrix, qMatrix)
6:      scores[i]= score
7: end for
```

Algorithm 2: Updation of Q-matrix

Input: current_state, action, rMatrix[11][11], qMatrix[11][11]

Output: Return the total of Q-matrix (sumA)

Initialization: int i = 0, j = 0, k = 0, index_of_max, int max_index[11]

double temp_max = 0.0, max_value = 0.0, sumA = 0.0, int a

```
1:  for i=0 to 10
2:          max_index[i] = 0
3:          if (temp_max == qMatrix[action][i]) then
4:                  max_index[j] = i;
5:                  j++;
6:          else if (temp_max <  qMatrix[action][i])
7:                  j = 0;
8:                  temp_max = qMatrix[action][i]
9:                  max_index[j] = i;
10:                 j++;
11:         end if
12: end for
13: a = returnRandom() % j
14: index_of_max = max_index[a]
15: max_value = qMatrix[action][index_of_max]
16: qMatrix[current_state][action] = rMatrix[current_state][action]
+ (gammaLR * max_value)
17: temp_max = 0.0
18: for i=0 to 11
19:         for j=0 to 11
20:                 if (qMatrix[i][j] > temp_max) then
21:                         temp_max = qMatrix[i][j]
22:                 end if
23:         end for
24: end for
25: if (temp_max > 0 ) then
26:         for i=0 to 11
27:                 for j=0 to 11
28:                         sumA =sumA + (qMatrix[i][j] /
temp_max)
29:                 end for
30:         end for
31:         sumA = sumA * 100
32:         return sumA
33: else
34:         return 0.0
35: end if
```

Once the matrix has been trained, the maximum value from the Q-Matrix is obtained so that the trained Q-Matrix shows normalized values. The normalized values means that there should not be a major difference in the range of Q-values.

Algorithm 3: Normalization Q-matrix

Input: qMatrix[11][11]

Output: Normalized qMatrix[11][11]

Initialization:

```
1: for i=0 to 11
2:          for j=0 to 11
3:                  if (final_max < qMatrix[i][j] )
4:                          final_max = qMatrix[i][j]
5:                  end if
6:          end for
7: end for
```

The initial row value will be the initial state. In order to find the optimal path, a visited array will be initialized to 0 and its value gets updated to 1 for each station after visiting that station. For initial row value, maximum column value is stored in a separate array and taken as a row value and again maximum column value for the same will be searched and stored in an array. The whole cycle iterates over and over until the chosen column value is the destination.

Algorithm 4: Finding optimal path

Input: qMatrix[11][11], final_state

Output: Optimal path

Initialization: int visited[11]={0}, int no_way=0, int row_max=0, int max_ind=0,

```
1: while (visited[final_state]!=1)
2:          print (curr_state)
```

```
3:        row_max=0
4:        max_ind=0
5:        for int i=0 to 11
6:                if (visited[i]==0) then
7:                    if(qMatrix[curr_state][i]>row_max) then
8:                        max_ind=i
9:                        row_max=qMatrix[curr_state][i]
10:                   end if
11:               end if
12:       end for
13:       curr_state=max_ind;
14:           visited[max_ind]=1;
15:        if(row_max==0) then
16:               no_way=1;
17:               break;
18:       end if
19:       if(curr_state==final_state) then
20:               break;
21:       end if
22: end while
23: if(no_way==1) then
24:       print( There's no way after this)
25: else
26:       print (curr_state is the shortest path)
26: end if
```

V. IMPLEMENTATION

Implementation of this algorithm is divided into five parts:
I) Defining the environment of the problem, II) Initializing the constant values, global arrays, matrices and the input nodes, III) Q-matrix training, IV) Normalizing the output values (because of which there is not much deviation in the values), V) Finding the optimal path.

A. Defining the environment

The whole graph will be defined as the sample metro chosen as shown in Fig 2.

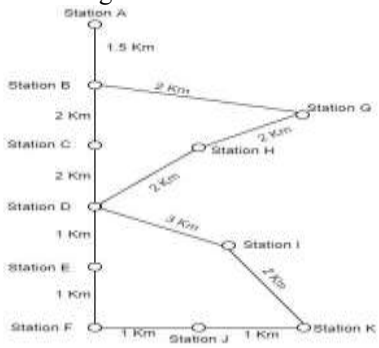

Fig 2: Sample Metro Graph

B. Initialization

Source and destination will be provided. Initially, all the values in Q-Matrix are defined as 0 and in R-matrix, -1 for non-adjacent nodes, 0 for adjacent nodes and 100 for adjacent nodes with the destination node. Fig 3 shows the Points matrix.

Fig 3: Points Matrix

C. Training

The whole idea behind reinforcement learning is to learn from experience. The process will be iterated for finding the optimal path almost 500-1000 times. Fig 4 shows the training flow chart.

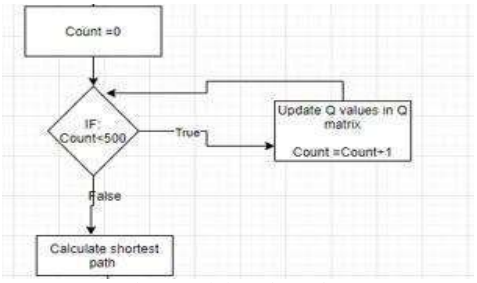

Fig 4: Training Flow Chart

D. Normalization

Once the training part is finished, the maximum value from the Q-Matrix is found so that while displaying the trained Q-Matrix, the normalized values can be displayed. Fig 5 shows the trained Q matrix.

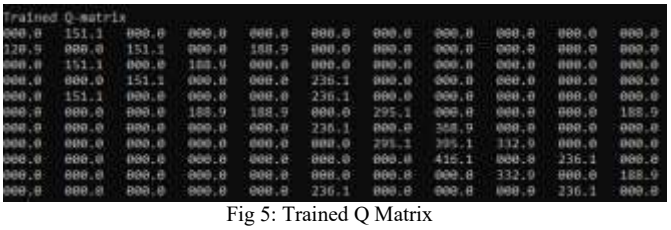

Fig 5: Trained Q Matrix

E. Optimal route calculation

It is known that the rows in Q-matrix define the state and the columns in that row describe the Q-value for transition to that state. The max value for that row will be calculated and the column value for that value will represent the next row to be taken. This whole process ends at the destination node.

VI. ANALYSIS

The results obtained from the proposed system need to be compared with the results of the pre-existing optimal path finding algorithm i.e. Dijkstra's algorithm.
System configuration chosen to test both algorithms on the same environment involves:

- Processor-Core i5
- RAM-12 GB
- Clock Speed- 1.7 GHz
- OS- Windows 10 Pro 64-bit

Both algorithms are compared on the basis of execution time and stationary/idle time. Stationary time is the time spent when a train stops at an intermediate station.

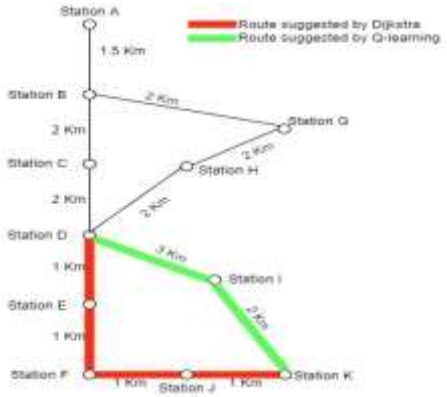

Fig 6: Optimal route selected by Dijkstra and Q learning.

For every path chosen, as shown in Fig 6, Dijkstra chooses the path having less distance to travel but it doesn't consider the idle time which indirectly increases the travel time, whereas the proposed algorithm takes into account waiting/idle time. The observations also show that, once a trained q-matrix is obtained, the proposed algorithm takes around 7-9 times less time as compared to Dijkstra's algorithm shown in Fig 8 and Fig 9.

VII. RESULTS

For input and output, a command line interface is developed which first displays the mapping of Station ID and Station Name as an informative description. With reference to the description, station IDs of origin and destination are taken as input, as shown in Fig 7. According to the input, the output window displays the best optimal path from origin to destination with intermediate stations as well, along with travel time and execution time for the process. Fig 8 and Fig 9 shows the output for Q learning and Dijkstra respectively.

```
Station ID      Station Name
-----------------------------------------------
0               Station A
1               Station B
2               Station C
3               Station D
4               Station E
5               Station F
6               Station G
7               Station H
8               Station I
9               Station J
10              Station K
enter the source and destination: 3 9
```
Fig 7: Input with station Ids and names

```
Path:
Station D-> Station I-> Station K is the shortest path
You will travel through 1 intermediate station(s) and you will experience
approximately 1 minutes of stationary/idle time

 Execution time: 0.001000 s
```
Fig 8: Q-Learning Output

```
Distance= 3.0 Km
Optimal path is -----  Station D -> Station E -> Station F -> Station J
You will travel through 3 intermediate station(s) and you will experience
approximately 3 minutes of stationery/idle time
Algorithm execution time = 0.009000 s
```
Fig 9: Dijkstra Output

CONCLUSION AND FUTURE SCOPE

Metro is the most common means of transport across many cities in the world, it is quite essential to locate an optimal path for travel to save time. There can be multiple ways to travel from one station to another, this technique finds an optimal route that traverses through the minimum number of stations and it also calculates the approximate time and distance for the journey. For future work switching between different metro lines, a bigger map, a better GUI is to be integrated with this technique.

REFERENCES

[1]. S. W. G. AbuSalim, R. Ibrahim, M. Zainuri Saringat, S. Jamel, and J. Abdul Wahab, "Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization," IOP Conference Series: Materials Science and Engineering, vol. 917, p. 012077, 2020.

[2]. M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. Awwal, and V. K. Asari, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," Electronics, vol. 8, no. 3, p. 292, 2019.

[3]. J. Czech, "Distributed Methods for Reinforcement Learning Survey," Reinforcement Learning Algorithms: Analysis and Applications, pp. 151–161, 2021.

[4]. G. Khekare, P. Verma, U. Dhanre, S. Raut, and S. Sheikh, "The Optimal Path Finding Algorithm Based on Reinforcement Learning," International Journal of Software Science and Computational Intelligence, vol. 12, no. 4, pp. 1–18, 2020.

[5]. Z. Yijing, Z. Zheng, Z. Xiaoyi and L. Yang, "Q learning algorithm based UAV path learning and obstacle avoidence approach," 2017 36th Chinese Control Conference (CCC), 2017, pp. 3397-3402, doi: 10.23919/ChiCC.2017.8027884.

[6]. Y. Hu, L. Yang, and Y. Lou, "Path Planning with Q-Learning," Journal of Physics: Conference Series, vol. 1948, no. 1, p. 012038, 2021.

[7]. L. Miao, "Optimal City Navigation for Pedestrians using Agent-specific Q-learning," 2018 Annual American Control Conference (ACC), 2018, pp. 4123-4128, doi: 10.23919/ACC.2018.8431199.

[8]. L. F. Yang and M. Wang, "Sample-Optimal Parametric Q-Learning Using Linearly Additive Features," ICML, 2019.

[9]. S. Demircan, M. Aydin, and S. S. Durduran, "Finding optimum route of electrical energy transmission line using multi-criteria with Q-learning," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3477–3482, 2011.

[10]. J. Ery and L. Michel, "Solving optimal stopping problems with Deep Q-Learning," arXiv: 2101.09682, 2021.

[11]. S. Manju, & M. Punithavalli, "An analysis of Q-learning algorithms with strategies of reward function", International Journal on Computer Science and Engineering, 3(2), 814-820, 2011.

[12]. [12] I. Szita and A. Lőrincz, "The many faces of optimism," Proceedings of the 25th international conference on Machine learning - ICML '08, 200

[13]. Rajendra Prasad P and Dr. Shivashankar, "Energy Secured Intrusion Detection System and Analysis of Attacks for Mobile Ad-Hoc Networks", Journal of Communications, ISSN: 1796-2021 (Online); 2374-4367 (Print) vol. 15, no. 5, pp. 406-414, DOI: 10.12720/jcm.15.5.406-414, 2020.

[14]. Arjun Kumar G B and Dr. Shivashankar, "Design and control of autonomous hybrid wind solar system with DFIG supplying three-phase four-wire loads", International Journal of Renewable Energy Technology, Inderscience Publication, (Online); Vol. 12, No. 3, pp. 269-299, DOI: 10.15041/15RET.2021.10039067

[15]. Sunil Kumar K N and Dr. Shivashankar, "Bio-signals Compression Using Auto Encoder", Journal of Electrical and Computer Engineering (Q2 Indexed), Institute of Advanced Engineering and Science publishers, ISSN: 2088-8708, Vol.11, No.1, pp. 424-433, http://doi.org/10.11591/ijece.v11i1.pp424-433. 2021.

[16]. Ravi Gatti and Dr. Shivashankar, "Improved Resource Allocation Scheme for Optimizing the Performance of Cell-Edge Users in LTE- A System", Journal of Ambient Intelligence and Humanized Computing, Springer Publisher, ISSN: 1868-5137 (print), ISSN: 1868-5145 (online), pp. 811–819,DOI: https://doi.org/10.1007/s12652-020-02084-x, 2021.

[17]. Rajendra Prasad P and Dr. Shivashankar, "Efficient Performance Analysis of Energy Aware on Demand Routing Protocol in Mobile Ad-Hoc Network", Engineering Reports, Volume 2, Issue 3, e12116 © John Wiley & Sons Ltd, Online ISSN: 2577-8196. doi.org/10.1002/eng2.12116, 2020.

[18]. Arjun Kumar GB, Shivashankar, "Design and control of grid-connected solar-wind integrated conversion system with DFIG supplying three-phase four-wire loads", International Journal of Power Electronics and Drive System, Institute of Advanced Engineering and Science Publisher, Vol.12, No.2, Jun 2021, pp. 1150-1161, DOI: 10.11591/ijpeds.v12.i2.pp1150-1161