Reinforcement Learning for Vehicle Route Optimization in SUMO

Koh Song Sang
Department of Computer Science,
Liverpool John Moores University,
Liverpool, L3 3AF, UK
S.S.Koh@2014.ljmu.ac.uk

Bo Zhou
Department of Computer Science,
Liverpool John Moores University,
Liverpool, L3 3AF, UK
B.Zhou@ljmu.ac.uk

Po Yang
Department of Computer Science,
Liverpool John Moores University,
Liverpool, L3 3AF, UK
B.Yang@ljmu.ac.uk

Zaili Yang
Department of Computer Science,
Liverpool John Moores University,
Liverpool, L3 3AF, UK
Z.Yang@ljmu.ac.uk

Hui Fang
Department of Computer Science,
Liverpool John Moores University,
Liverpool, L3 3AF, UK
H.Fang@ljmu.ac.uk

Jianxin Feng
Department of Computer Science,
Liverpool John Moores University,
Liverpool, L3 3AF, UK
J.Feng@ljmu.ac.uk

***Abstract* – Urban traffic control becomes a major topic for urban development lately as the growing number of vehicles in the transportation network. Recent advances in reinforcement learning methodologies have shown highly potential results in solving complex traffic control problem with multi-dimensional states and actions. It offers an opportunity to build a sustainable and resilient urban transport network for a variety of objects, such as minimizing the fuel consumption or improving the safety of roadway. Inspired by this promising idea, this paper presents an experience how to apply reinforcement learning method to optimize the route of a single vehicle in a network. This experience uses an open-source simulator SUMO to simulate the traffic. It shows promising result in finding the best route and avoiding the congestion path.**

***Index Terms* — Reinforcement Learning, Traffic Control, Transportation Network, Route Optimization**

I. INTRODUCTION

As the population of urban and vehicle number keep increasing rapidly, the demand of the urban transportation continues to grow. In this circumstance, traffic congestion becomes one of the major problem in urban development. The modern solution for traffic congestion aims to achieve an autonomous, environmental and sustainable traffic control, optimize each vehicle's route in transportation network by reducing their travel time to destinations.

Nowadays wireless sensor networks are common and reliable which enable the Vehicle to Vehicle (V2V), Vehicle to Roadside (V2R) or Vehicle to Infrastructure (V2I) communication. With the sharing data through the network, observing and monitoring the state of the transportation would be achievable. By accessing the real-time traffic data, the vehicle traffic management system controller could response instantly to control the urban traffic flow [1].

Route optimization is a general solution for Vehicle Routing Problem (VRP) which is described as the problem of designing optimal delivery or collection routes from one or several depots to several geographically scattered cities or customers, subject to side constraints [2].

As the complex nature of the urban transportation, the vehicle routing problem is a complex problem which involves many features that need to be considered. This paper used reinforcement learning method and showed how it could be applied to solve the vehicle routing problem. The framework of reinforcement learning is to learn from model with optimal policy based on its observation. Each action that the agent would take will lead to a reward or punishment with the new observation of the state. Through its learning progress, the agent will learn a distributed routing policy that could maximize the capacity of urban. This process could be treated as a Markov Decision Process (MDP) [3].

This research uses SUMO (Simulation of Urban Mobility) as the traffic simulator. SUMO is an open source, microscopic, multimodal traffic simulator. It allows the user to simulate how a specified traffic demand performs on a given road network. SUMO is an excellent choice for urban traffic simulation as it is able to perform an optimised traffic distribution method based on vehicle type or driver behaviour. It is also able to update the vehicle's route if there is any congestion or if an accident occurs, finding the best routing option to enhance urban sustainability and resilience. It also supports Traffic Control Interface (TraCI for short), which is a Python API that treats the SUMO simulation as a server and allows the user to gather data from a traffic simulation or modify the simulation.

This makes TraCI a perfect tool for performing Reinforcement Learning in the SUMO simulator.

The remaining of the report is as follows: section 2 provides a systematic review on the previous research on route optimization. Section 3 shows how reinforcement learning could be applied in SUMO. Section 4 presents an example of reinforcement learning approach in SUMO, section 5 discusses the future work and conclusion of this research.

II. Related Work

As mentioned in previous session, Vehicle routing problem is a major challenge in urban transportation network. Finding an optimal route is a general method for resolving Vehicle Routing Problem (VRP). In this session, we present a brief review of related studies to search for the optimal route for vehicle in transportation network.

In 1959, Dijkstra proposed a static algorithm to find the shortest path in two nodes [4]. This approach is not considering any extra factor such as congestion, vehicle amount etc. With the nature of urban transportation network, the optimal route is not always the shortest path between two nodes. Therefore, ideal route optimization method should be considering the latest state of the transportation network and make adaption in real time.

Dorigo et al. [5] proposed Ant-colony algorithm which was inspired by the natural behavior performed by ants in finding food resources. In this natural phenomenon, ants leave pheromone trails to allow other ants to track the path and eventually find the food resources. Previous experiments have proven that ants can find the shortest route between two individual sections. Therefore, ant-colony algorithm is widely applied to solve vehicle routing problem, although some modifications have been applied depending on different circumstances.

Kammoun et al. [6] however proposed an adaptive vehicle guidance system which can search for the best path in a smarter way by using real-time changes in the network. To achieve dynamic traffic control and improve driver's request management, this method used three types of agents, which are:

- Intelligent vehicle-ant agent (IVAA): standing for a vehicle and encapsulates a deliberative module for the selection of the best itinerary alternative.
- Road Supervisor Agent (RSA): Representing a software agent implanted in the server.
- City Agent (CA): Representing a software agent to manage the road network in the city to obtain a better exploitation of the network.

This method is focused on individual drivers and lack of consideration for the bigger picture of urban transportation network. It is difficult to manage a large, massive and complex urban transportation network.

Cong et al. [7] developed a model to optimise dynamic traffic routing by using a two-steps approach: network pruning and network flow optimisation. In the network-pruning phase, ant pheromone is removed after the best route is found by the agents to increase exploration rate. In the flow optimisation phase, which is based on Ant-Colony Optimization with the stench pheromone and colored pheromone, the agents correspond to the links selected in the network-pruning phase only. Moreover, this two-steps approach effectively reduces the computing burden when addressing complex and dynamic traffic control problems.

With SUMO is a new growing tool for traffic simulation, there are couple attempts to apply reinforcement learning in SUMO in previous studies. Although as far as we concerned, there are no recent study is applying reinforcement method for route optimization in SUMO, the idea of how these studies design and observe the state are very helpful.

David Isele [8] analyzed how the knowledge to autonomously handle one type of intersection, represented as a Deep Q-Network (DQN), translates to other types of intersections. The DQN takes each step of simulation in SUMO as a graphic input and analyzed how to handle the different type of interception. Although this paper focus on analyzing different type of interception rather than route optimization, it provides very good idea to explain how to apply reinforcement learning in SUMO.

Seyed Sajad Mousavi [9] proposed two reinforcement learning algorithms (deep policy-gradient and value-function based agents) which can predict the best possible tra□c signal for a tra□c intersection. The policy-gradient based agent maps its observation based on the snapshot of the current state of a graphical tra□c simulator directly to the control signal, however value-function based agent first estimates values for all legal control signals. Then the agent could make selection depend on the highest value.

Wade Genders [10] proposed a deep neural network as a traffic signal control agent (TSCA) that trained by using reinforcement learning method to solve the traffic signal control problem by developing an optimal control policy. He used a new state space definition which named the discrete traffic state encoding (DTSE), that contains more relevant information compared to previous research's state space definitions.

Cathy Wu [11] however proposed a method to do benchmarking for the performance of classical (including hand-designed) controllers with learned policies (control laws) by using reinforcement learning approach.

Giorgio Stampa [12] designed Deep-Reinforcement Learning agent which is able to adapt automatically to current traffic conditions and proposes tailored configurations that attempt to minimize the network delay. However, it approaches only focus on one vehicle rather than the whole urban transportation network.

III. Reinforcement Learning in SUMO

Simulation of Urban Mobility (or SUMO for short) is a powerful simulator designed to handle a large load network and specified traffic demand, including a vehicle route and car following model. It also provides a lot of useful information such as vehicle speed, model, and position.

One of the major features of SUMO is the Traffic Control Interface (or TraCI for short), which is a Python API that treats the SUMO simulation as a server and allows users to gain information from a traffic simulation or modify the simulation.

TraCI enables an interface to allow third party systems (or libraries) to integrate with the SUMO traffic simulation. This session explains how a reinforcement learning method could work with SUMO by using TraCl, and how this could benefit urban traffic management.

Reinforcement learning as a machine learning technique has led to very promising results as a solution for complex systems. A reinforcement learning method is able to gain knowledge or improve the performance by interacting with the environment itself. The theory of reinforcement learning is inspired by behavioural psychology, it gains reward after taking certain actions under a policy in an environment. The goal of reinforcement learning is to learn an optimal policy based on the reward obtained by repeating the interaction with the environment. This consistently optimises the policy and eventually creates a solution.

A good example of the application of reinforcement learning in a real-world situation is in urban traffic management. As the number of vehicles within our urban transport network increases, designing a smart traffic management system to perform intelligent routing for vehicles is in high demand. However, the complexity of the urban transportation network causes a lot of challenges in traffic management, such as high-speed changes in traffic systems and the wide distribution of vehicles on the roadway. To deal with these challenges, a reinforcement learning approach would be useful as it has already successfully proven that it is able to handle complex optimisation problems.

As mentioned in the beginning of this session, the SUMO simulator can enable third party systems to approach reinforcement learning. In this case, TraCI will play the role of the "convertor" between SUMO and the reinforcement learning approach to establish this interaction. TraCI is able to retrieve every piece of information in the simulation, including the vehicle and network. This provides useful features for the reinforcement learning agent to justify the states of the environment. Based on the observation of the states, we could set and assign the rewards accordingly and let reinforcement learning optimise the policy based on the reward. After that, the reinforcement learning agent will assign a new action to SUMO through TraCI and continuously observe the environmental state.

TraCI could be accessed using multiple programming languages, with the most common language being Python. The package tools/TraCl in the SUMO simulator allows users to interact with SUMO using Python. This is advantageous as Python is already a well-established script language for machine learning, providing useful libraries (such as Numpy and Pandas) whilst implementing a machine learning algorithm.

The interaction between the reinforcement learning agent and the environment via TraCI would be continued until it reaches a terminal state, or the agent meets a termination condition. Essentially, the reinforcement learning techniques apply the Markov decision processes (MDPs). A MDP is defined as a five-tuple $< S, A, T, R, \gamma >$, where S is the collection of states, A is the set of actions which could change the status, T is the transition function, which is the probability of the state change under the certain action, R is the reward function, and γ is known as the discount factor, which models the importance of the future and immediate rewards.

Reinforcement learning optimises its policy by repeating the following step: at each time step t, the reinforcement learning agent perceives the state from state collection S and, based on its observation, selects an action and executes it to lead the state of the environment transition to the next state. Then, the agent receives immediate reward R, observes the new state, and updates the policy with the equation above including the discounted reward γ.

In order to approach reinforcement learning in the SUMO simulator, we need corresponding elements for a Markov decision process in SUMO. The SUMO simulator uses default routing method "DuaRouter" to generate route files for every vehicle in the simulation. "DuaRouter" performs dynamic user assignment (DUA) based on shortest path computation. The reinforcement learning agent will replace the default routing method with the optimal policy that it has learnt.

IV. EXAMPLE

This post will demonstrate a simple simulation that applies reinforcement learning approach in SUMO simulator by using TraCI. This simulation is to show a vehicle learns how to arrive its destination by selecting the right path to avoid traffic congestion. It also shows how the vehicle optimise his path selection policy based on the reward it receives. Below shows the general process of this example:

s = state, a = action, r = reward, γ = discount factor
Initialize Q(s, a) arbitrary
Repeat (for each episode):
    Initialize s
    Repeat (for each step of episode):
        Choose a from s using policy derived from Q
        Take action a, observe r, s'
        $Q(s, a) \leftarrow [r + \gamma \max Q(s', a')]$
        $s \leftarrow s'$
    until s is terminal

First thing first, we need to prepare a network file which contains the roadway information, and a route file which contains the vehicle route information for building the environment for simulation.

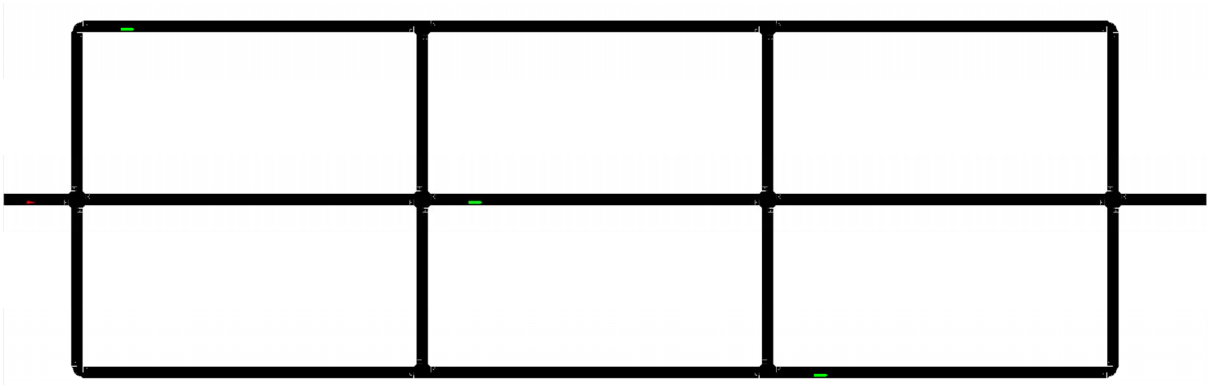

Figure 1

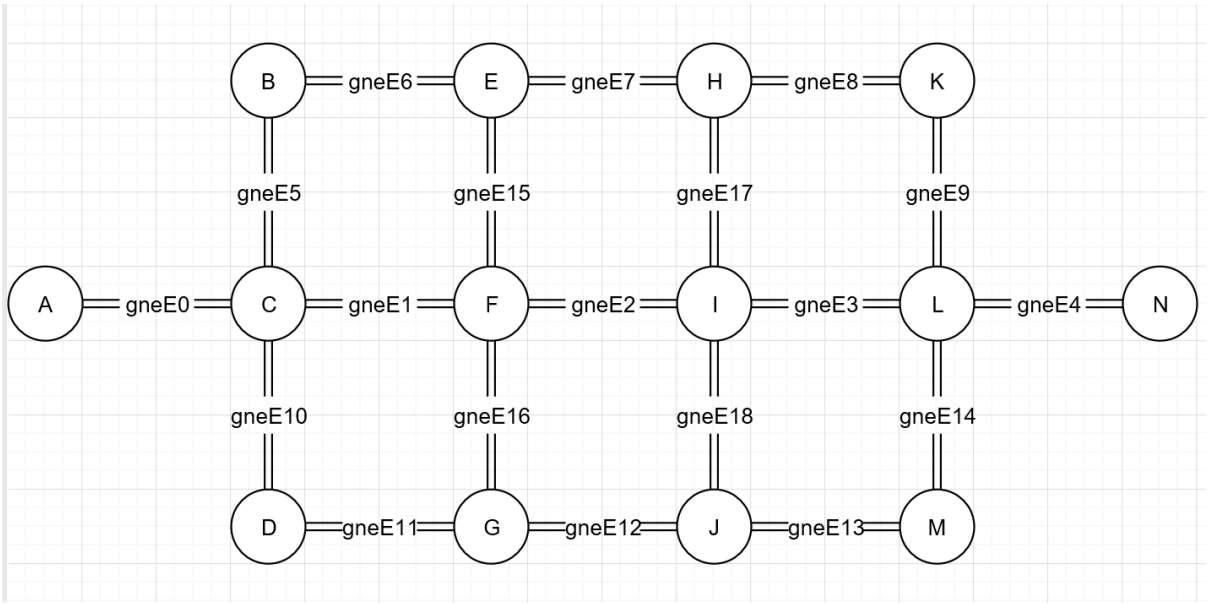

Figure 2

The figure 1 shows the roadway network of the simulation which is generated by the .net file and .rou file. The 3 little green rectangle shapes on the roadway are the trucks that block the road and cause the congestion. The little triangle shape on the left is the vehicle that like to arrive its destination, which is on the right of the network.

However, figure 2 is the same network design as figure 1 but shows the id of the edges that connects each node, which also indicate the path that the vehicle would take. if a vehicle move from left to right node or up to down node, we record the vehicle takes the path with the positive edge ID, otherwise (from right to left or down to up) we record the vehicle takes the path with the negative edge ID. For instance, if a vehicle moves from node A -> C -> F -> E -> B -> C and back to node A, the taken path would be recorded as [“gneE0”, “gneE1”, “-gneE15”, “-gneE6”, “gneE5”, “gneE0”].

The next step we control the SUMO simulator by using TraCI library. Meanwhile, we can also connect to library for reinforcement learning approach. In this case, numpy and pandas libraries would be used. Numpy library is for supporting multi-dimensional arrays and metrices along with a large set of mathematical functions however pandas library is for data manipulation and analysis. Both libraries are very widely used by machine learning.

Before we start the simulation for learning, we initialised the q-table. A q-table is an updatable table for path selection policy. The value of the table would be updated every time when an action is taken in order to optimise the policy. The table will record every state of the environment and the probability of taking which actions. The format of the q-table for this case is shown below.

States	First Road	Second Road	Third Road
[current edge]	[score]	[score]	[score]

From table above we could see we set the edge ID as the stage with the score of their 3 available moves. If vehicle enters to a new edge that not covered in the q-table, we will record the edge ID as a new record and assign it moves randomly and update the score depend on next edge the vehicle enter.

We set the learning episode as 30, which mean the we will run the simulation 30 times, each simulation will be end until the vehicle on the left finds his way to destination or stuck in the congestion. The vehicle starts from edge "gneE0", aim to arrive edge "gneE4" and try to avoid the congestion edges, which are "gneE2", "gneE6" and "gneE13". Therefore, we give reward when the vehicle arrive edge "gneE4" and give punishment if the vehicle enters "gneE2", "gneE6" and "gneE13".

Reinforcement learning make action move that based on q-table. After that it updates the q-table regarding the reward or punishment it receives in the next stage by making that action. Therefore, next time the it will act greedier by looking for the action that contains higher score. In this case, we update the q-table score with the algorithm below:

q_table.[edge_id, action] += learning_ rate * (q_target - q_predict) where discount factor is 0.1, q_target is the reward or punishment that we receive in the next stage, and q_predict is the current score for the particular action (initially 0).

After running the simulation 30 times, we can see the vehicle is getting smarter to reach its destination by avoiding the congestion path. Below shows the total path that be taken by the vehicle in each simulation.

```
Episode 1
Route taken: ['gneE5', 'gneE6']
Episode 2
Route taken: ['gneE5', 'gneE6']
Episode 3
Route taken: ['gneE10', 'gneE11', '-gneE16', '-gneE15', 'gneE7', 'gneE8', 'gneE9', 'gneE4']
Episode 4
Route taken: ['gneE10', 'gneE11', '-gneE16', 'gneE2']
Episode 5
Route taken: ['gneE10', 'gneE11', '-gneE16', '-gneE15', 'gneE7', 'gneE8', 'gneE9', 'gneE4']
Episode 6
Route taken: ['gneE10', 'gneE11', '-gneE16', '-gneE15', '-gneE6', '-gneE5', 'gneE1', 'gneE2']
Episode 7
Route taken: ['gneE10', 'gneE11', '-gneE16', '-gneE15', '-gneE6', '-gneE5', 'gneE10', 'gneE11', '-gneE16', '-gneE15', 'gneE7', 'gneE17', 'gneE3', '-gneE9', '-gneE8', '-gneE7',
'gneE15', '-gneE1', 'gneE10', 'gneE11', 'gneE12', 'gneE13']
Episode 8
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', '-gneE17', '-gneE7', 'gneE15', 'gneE16', 'gneE12', '-gneE18', '-gneE2', '-gneE1', 'gneE10', 'gneE11', 'gneE12', '-gneE18',
'-gneE17', 'gneE8', 'gneE9', 'gneE4']
Episode 9
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 10
Route taken: ['gneE10', 'gneE11', '-gneE16', '-gneE15', 'gneE7', 'gneE8', 'gneE9', 'gneE4']
Episode 11
Route taken: ['gneE10', 'gneE11', '-gneE16', '-gneE15', '-gneE6', '-gneE5', 'gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 12
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 13
Route taken: ['gneE10', 'gneE11', '-gneE16', '-gneE15', '-gneE6', '-gneE5', 'gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 14
Route taken: ['gneE1', 'gneE16', '-gneE11', '-gneE10', 'gneE5', 'gneE6']
Episode 15
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 16
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 17
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 18
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 19
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 20
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', '-gneE17', 'gneE8', 'gneE9', 'gneE4']
Episode 21
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 22
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 23
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 24
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 25
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 26
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 27
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 28
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 29
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
Episode 30
Route taken: ['gneE10', 'gneE11', 'gneE12', '-gneE18', 'gneE3', 'gneE4']
```

Figure 3

According Figure 3, we can see from the episode 1 to 10, we can see the vehicle enters to the congestion path or taking longer path to arrive destination. From episode 11 to 20, it is getting smarter but still take longer path occasionally. From episode 21 to 30, it becomes very consistent to take shortest path to arrive the destination without entering congestion path.

V. FUTURE WORK AND CONCLUSION

With the real roadway situation contains more properties and countless state to observe, the next stage we will design a neural network to build a Deep Q-Network framework to cope with

more complex transportation network and take consideration for all vehicles in the network.

To summarize this paper, We developed a simple program to perform reinforcement learning in SUMO simulator via TraCI and show that the vehicle is getting more intelligent to reach the destination and avoid the congestion path. This proves that reinforcement learning is applicable in SUMO for route optimization and it performs well in result.

VI. REFERENCES

[1] Kapileswar Nellore and Gerhard P. Hancke. A Survey on Urban Traffic Management System Using Wireless Sensor Networks. January 2016.

[2] Gilbert Laporte. The Vehicle Routing Problem: An overview of exact and approximate algorithms. 1991.

[3] Richard S. Sutton, Andrew G. Barto. Reinforcement Learning: An Introduction. 2012

[4] E. Dijkstra: ' A Note on Two Problems in Connexion with Graphs', Numerische Mathematik, Vol. 1, pp. 269-271. 1959.

[5] Dorigo, M., Maniezzo, V., & Colorni, A. Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics—Part B, 26(1), 29–41. 1996

[6] Kammoun, H.M., Kallel, I., Alimi, A.M. and Casillas, J. 'An adaptive vehicle guidance system instigated from ant colony behavior', IEEE International Conference on Systems Man and Cybernetics (SMC), pp.2948–2955, IEEE.

[7] Zong, X., Xiong, S., Fang, Z. and Li, Q. (2010) 'Multi-ant colony system for evacuation routing problem with mixed traffic flow', in IEEE Congress on Evolutionary Computation (CEC), pp.1–6, IEEE. 2010.

[8] David Isele, Akansel Cosgun and Kikuo Fujimura, Analyzing Knowledge Transfer in Deep Q-Networks for Autonomously Handling Multiple Intersections. 2017

[9] Seyed Sajad Mousavi, Michael Schukat, Enda Howley. Traffic Light Control Using Deep Policy-Gradient and Value-Function Based Reinforcement Learning. 2017.

[10] Wade Genders, Saiedeh Razavi. Using a Deep Reinforcement Learning Agent for Traffic Signal Control. 2016.

[11] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, Alexandre M Bayen. Flow: Architecture and Benchmarking for Reinforcement Learning in Traffic Control. 2017.

[12] Giorgio Stampa, Marta Arias, David Sánchez-Charles, Victor Muntés-Mulero, Albert Cabellos. A Deep-Reinforcement Learning Approach for Software-Defined Networking Routing Optimization. 2017