

6-2021

A REINFORCEMENT LEARNING APPROACH TO VEHICLE PATH OPTIMIZATION IN URBAN ENVIRONMENTS

Shamsa Abdulla Al Hassani

Follow this and additional works at: https://scholarworks.uaeu.ac.ae/all_theses



Part of the [Software Engineering Commons](#)

Recommended Citation

Al Hassani, Shamsa Abdulla, "A REINFORCEMENT LEARNING APPROACH TO VEHICLE PATH OPTIMIZATION IN URBAN ENVIRONMENTS" (2021). *Theses*. 810.
https://scholarworks.uaeu.ac.ae/all_theses/810

This Thesis is brought to you for free and open access by the Electronic Theses and Dissertations at Scholarworks@UAEU. It has been accepted for inclusion in Theses by an authorized administrator of Scholarworks@UAEU. For more information, please contact mariam_aljaberi@uaeu.ac.ae.

United Arab Emirates University

College of Information Technology

Department of Computer Science and Software Engineering

A REINFORCEMENT LEARNING APPROACH TO VEHICLE PATH
OPTIMIZATION IN URBAN ENVIRONMENTS

Shamsa Abdulla Al Hassani

This thesis is submitted in partial fulfilment of the requirements for the degree of
Master of Science in Software Engineering

Under the Supervision of Professor Abderrahmane Lakas

June 2021

Declaration of Original Work

I, Shamsa Abdulla Al Hassani, the undersigned, a graduate student at the United Arab Emirates University (UAEU), and the author of this thesis entitled “*A Reinforcement Learning Approach to Vehicle Path Optimization in Urban Environments*”, hereby, solemnly declare that this thesis is my own original research work that has been done and prepared by me under the supervision of Professor Abderrahmane Lakas, in the College of Information Technology at UAEU. This work has not previously formed the basis for the award of any academic degree, diploma or a similar title at this or any other university. Any materials borrowed from other sources (whether published or unpublished) and relied upon or included in my thesis have been properly cited and acknowledged in accordance with appropriate academic conventions. I further declare that there is no potential conflict of interest with respect to the research, data collection, authorship, presentation and/or publication of this thesis.

Student's Signature: Shamsa Date: 24-06-2021

Copyright © 2021 Shamsa Abdulla Al Hassani
All Rights Reserved

Approval of the Master Thesis

This Master Thesis is approved by the following Examining Committee Members:

- 1) Advisor (Committee Chair): Prof. Abderrahmane Lakas

Title: Professor

Department of Computer and Network Engineering

College of Information Technology


Signature  _____ Date 13/06/2021

- 2) Member: Dr. Amir Ahmad

Title: Assistant Professor

Department of Information Systems and Security

College of Information Technology

Signature  _____ Date 13/06/2021

- 3) Member: Dr. Adel Khelifi

Title: Associate Professor

Department of Computer Science and Information Technology

Institution: Abu Dhabi University, UAE

Signature  _____ Date 13/06/2021

This Master Thesis is accepted by:

Dean of the College of Information Technology: Professor Taieb Znati

Signature Taieb Znati Date 30/07/2021

Dean of the College of Graduate Studies: Professor Ali Al-Marzouqi

Signature Ali Hassan Date 30/07/2021

Copy ____ of ____

Abstract

Road traffic management in metropolitan cities and urban areas in general is an important component of Intelligent Transportation Systems (ITS). With the increasing number of world population and vehicles, a dramatic increase in the road traffic is expected putting pressure on the transportation infrastructure. Therefore, there is a pressing need to devise new ways to optimize the traffic flow in order to accommodate the growing needs of transportation systems. This work proposes to use an Artificial Intelligent (AI) method based on reinforcement learning techniques for computing near-optimal vehicle itineraries applied to Vehicular Ad-hoc Networks (VANETs). These itineraries are optimized based on the vehicle's travel distance, travel time, and traffic road congestion. The problem of traffic density formulated as a Markov Decision Process (MDP). In particular, this work introduce a new reward function that takes into account the traffic congestion when learning about the vehicle's best action (best turn) to take in different situations. To learn the effect of this approach, the work investigated different learning algorithms such as Q-Learning and SARSA in conjunction with two exploration strategies: (a) e-greedy, and (b) Softmax. A comparative performance study of these methods is presented to determine the most effective solution that enables the vehicles to find a fast and reliable path. Simulation experiments illustrate the effectiveness of proposed methods in computing optimal itineraries allowing vehicles to avoid traffic congestion while maintaining reasonable travel times and distances.

Keywords: VANET, reinforcement learning, markov decision process, road traffic congestion.

Title and Abstract (in Arabic)

نهج التعلم المعزز لإيجاد المسار شبه الأمثل في البيئات الحضرية

الملخص

في الوقت الحاضر، تعتبر إدارة حركة المرور أحد أهم جوانب المناطق والمدن الحضرية. مع التزايد السريع في عدد السكان والمركبات في جميع أنحاء العالم، من المتوقع أن يزداد الحمل المروري على البنية التحتية للنقل بشكل كبير. وبالتالي، هناك حاجة لتحسين تدفق حركة المرور من أجل تلبية الاحتياجات المتزايدة لأنظمة النقل. في هذا العمل، اقترحنا استخدام تقنية التعلم المعزز مع VANET لتحديد المسار شبه الأمثل في شبكة النقل من حيث أقل مسافة، أقل وقت سفر وازدحام على الطريق. على وجه الخصوص، نقدم وظيفة مكافأة جديدة تأخذ الازدحام المروري في عين الاعتبار لتعليم السيارة أفضل إجراء يمكن اتخاذه في المواقف المختلفة. تم تطبيق هذا الحل باستخدام خوارزميات تعليمية مختلفة، Q-Learning و SARSA جنباً إلى جنب مع استراتيجيتين للاستكشاف: $\epsilon - greedy$ و $softmax$. تم مقارنة أداء هذه الطرق لتحديد الحل الأكثر فعالية الذي يمكن السيارة من العثور على مسار سريع وموثوق. أظهرت التجارب التي تم إجراؤها أن السيارة تختار مسار الرحلة شبه الأمثل مع ازدحام مروري طفيف ووقت سفر أقل مقارنة بالمسارات الأخرى.

مفاهيم البحث الرئيسية: شبكة المركبات المخصصة، التعلم المعزز، عملية اتخاذ القرار ماركوف، الازدحام المروري على الطرق.

Acknowledgements

Throughout the writing of this thesis, I have received a great deal of support and assistance.

I would like to thank my supervisor, Professor Abderrahmane Lakas, for providing invaluable guidance throughout this research. His constructive feedback pushed me to sharpen my thinking and brought my work to a higher level. It was a great privilege and honor to work and study under his guidance.

My thanks go to the examining committee members Dr. Adel Khelifi and Dr. Amir Ahmad for their valuable comments that developed this thesis.

I would like to acknowledge all College of Information Technology members at the United Arab Emirates University for the invaluable assistance during my study, especially Dr. Salah Bouktif.

Nobody has been more important to me in the pursuit of this research than my family members. I would like to thank my parents for their wise counsel and sympathetic ear. A special thanks to my sister Amal and brother Khaled for the persistent help all the time. They are always there for me. Finally, I would like to thank my friends, Meera and Nouf, who provided stimulating discussions and happy distractions to rest my mind outside of my research.

Dedication

To my beloved parents and family

Table of Contents

Title	i
Declaration of Original Work	ii
Copyright	iii
Approval of the Master Thesis	iv
Abstract	vi
Title and Abstract (in Arabic)	vii
Acknowledgements	viii
Dedication	ix
Table of Contents	x
List of Tables	xii
List of Figures	xiii
List of Abbreviations	xiv
Chapter 1: Introduction	1
1.1 Statement of the Problem	3
1.2 Research Questions	4
1.3 Methodology	4
1.4 Structure of the Thesis	5
Chapter 2: Literature Review	6
2.1 Reinforcement Learning	6
2.2 Road Traffic Congestion Systems.....	7
2.3 Route Planning Algorithms.....	10
Chapter 3: Reinforcement Learning.....	14
3.1 Machine Learning	14
3.2 Markov Decision Process.....	16
3.3 Policies and Value Functions.....	17
3.4 Optimal Policy	18
3.5 Q-learning and Sarsa Algorithms.....	19
3.6 Exploration-Exploitation Trade-off	21
Chapter 4: System Design.....	24
4.1 Road Traffic Model.....	24
4.2 Reinforcement Learning	26
4.2.1 State Space.....	27
4.2.2 Action Space.....	28
4.2.3 Reward Function	29
4.2.4 RL Algorithms.....	31

Chapter 5: Evaluation and Performance Analysis	34
5.1 Experimental Environment	34
5.2 Tuning Learning Parameters	38
5.3 Comparisons.....	43
5.3.1 Comparing the Length of the Path and Traffic Load	43
5.3.2 Comparing the Average Cumulative Rewards	45
5.3.3 Comparing the Average Training Times	48
5.3.4 Comparing the Average Visited State	49
5.4 Discussion	50
Chapter 6: Conclusion.....	52
References	53

List of Tables

Table 1: Example of Q-Table.....	20
Table 2: Simulation and learning parameters.....	33
Table 3: Simulation parameters of 6×6 , 10×10 , and 20×20 , respectively	34
Table 4: Optimal parameters for each type of experiment.....	43
Table 5: Comparison of the path length and load on three different experimental setups	44
Table 6: Different performances obtains from different weight factors in 20×20 map	45
Table 7: Comparison of average cumulative rewards.....	47
Table 8: Comparison of average training times	48
Table 9: Comparison of average visited states.....	50

List of Figures

Figure 1: Machine Learning Types	14
Figure 2: Reinforcement learning schema	15
Figure 3: Example of road network environment	24
Figure 4: A virtual environment divided into cells	27
Figure 5: Four possible actions	28
Figure 6: 6×6 map	35
Figure 7: 10×10 map	36
Figure 8: 20×20 map	37
Figure 9: Comparison of parameters for Q-learning- e-greedy in term of cumulative reward with the three different experimental setups	39
Figure 10: Comparison of parameters for Sarsa- e-greedy in term of cumulative reward with the three different experimental setups	40
Figure 11: Comparison of parameters for Q-learning-softmax in term of cumulative reward with the three different experimental setups	41
Figure 12: Comparison of parameters for Sarsa- softmax in term of cumulative reward with the three different experimental setups	42
Figure 13: Learning curve of the proposed algorithms tested in 6×6 map.....	46
Figure 14: Learning curve of the proposed algorithms tested in 10×10 map.....	46
Figure 15: Learning curve of the proposed algorithms tested in 20×20 map.....	47
Figure 16: Average training time of the proposed algorithms tested in 6×6, 10×10 and 20×20 map	49

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
MDP	Markov Decision Process
RL	Reinforcement Learning
SARSA	State–Action–Reward–State–Action
VANET	Vehicular Ad Hoc Network

Chapter 1: Introduction

In recent decades, the majority of the world's population has been heading to the urban environment, which has directly impacted every aspect of life. The rate of automobile growth is outpacing the expansion of the road network infrastructure in urban areas due to space and budget limitations. This situation causes severe traffic congestion on the road and increases the vehicle's travel time. As a result, excessive carbon emissions pollute cities and degrade the quality of human life. Intelligent Transportation Systems (ITS) have emerged as a potential solution to improve highway efficiency. It uses several communication channels and networks, such as Vehicular Ad-hoc Network (VANET), to monitor and regulate vehicular traffic in an intelligent manner. VANET is a special class of Mobile ad-hoc Networks (MANET) in which moving vehicles act as either a node or a router to exchange data between them to create an extremely large scale mobile network. It is aimed to support both vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications (Sheikh & Liang, 2019).

Another technique that used recently to optimize the traffic management is Reinforcement learning. RL is a subfield of machine learning in which an agent (decision maker) learns to make sequential decisions by interacting with an environment (Gottesman et al., 2018). The learning strategy of RL follows the method of “trial and error” to learn an optimal policy by perceiving states from the environment, taking an action based on the current states, and receiving penalty or rewards from the environment. The policy that selects the best action at each state to maximize the expected long-term cumulative reward is considered as the optimal one. RL algorithms can be found implemented in robot control (Kober et al., 2013) and

board games like Tic-tac-toe, and chess. In these kinds of problems, agent is modeled to learn through frequent interactions with their environment and the returns signal of these interactions; it learns from past experience. These tasks deal with one learning agent only (single-agent). However, various real- world decision problems such as swarm robot and traffic are inherently composed of several tasks which demand models with multiple agents. Multi-agent models can simplify the complex problem by dividing knowledge among the agents.

The family of RL has different algorithms such as Q-Learning (Ho et al., 2006), State Action Reward State Action (SARSA) (Chen & Wei, 2008). The most important feature of these algorithms is that they do not require knowledge of the environment with which they interact. In 2013, Google Deepmind team have proposed the first successful Deep Q-network (DQN) framework that combines deep learning with reinforcement learning. The authors used a Deep Q-network (DQN) to estimate the Q-function for Q-learning. The combination of neural networks and reinforcement learning is capable of solving more complex tasks as all have been witnessed in many applications ranging from Google¹, Uber², and Tesla³ autonomous car to Google's DeepMind AlphaGo⁴ algorithm that defeated the World Champion in the game of Go.

¹ <https://www.google.com/selfdrivingcar/>

² <https://www.uber.com/en-BE/>

³ <https://www.tesla.com/autopilot>

⁴ <https://deepmind.com/research/case-studies/alphago-the-story-so-far>

1.1 Statement of the Problem

Transportation and traffic systems are the backbones of any city. It is regarded as an essential component of the town's growth, development and fulfills users presumed social and economic needs. However, as the population and automobiles grow, the traffic demand on transportation infrastructure grows, making it difficult for the transportation system to serve the public interest. Traffic congestion is the term for this problem, and it consists of incremental delay in travel time, vehicle operating costs such as fuel consumption, pollution emissions due to CO₂ emissions (Peters et al., 2004). Furthermore, it causes more stress and inconvenience to drivers for additional time spent and delaying their work and interests. In this circumstance, traffic congestion becomes an ever-increasing problem in urban development.

According to the Ohio Department of Transportation (Azimian, 2011), traffic congestion stops Honda's employees from arriving on schedule, threatening Honda's low-inventory strategy in Ohio. There are always concerns that traffic load could cause emergency services to be delayed at crucial times when they need to arrive as soon as possible. In 2018, recent research stated that the drivers spent an average of 50 during peak traffic in Abu Dhabi. Simultaneously, the congestion increased in Dubai as the time spent reached an average of 80 hours stuck in traffic jams (Cleofe, 2019) .

Designing efficient real-time path planning can efficiently relieve traffic congestion in urban scenarios. Thus, this thesis aims to investigate the use of Reinforcement Learning techniques in the computation of the best vehicle trajectories in Vehicular Ad hoc Networks (VANET) for the purpose of avoiding and dissipating road traffic congestion. Since the reinforcement learning not always provide the optimal paths in the network, this work focuses on computing the near-optimal

trajectories which are close to the optimum solution. These itineraries are optimized based on the vehicle's travel distance, travel time, and traffic road congestion. The congestion state on the road is assumed to be collected and exchanged using VANET. This information will then be used by reinforcement learning for path planning based on the road traffic congestion. Based on that, the vehicles will be distributed proportionally to the road's capacity in the network environment. Therefore, the driver will achieve reasonable travel times from his current location to his destination.

1.2 Research Questions

The research questions that will guide this thesis are as follows:

1. How to model a road environment, road traffic and determine the state and action space that characterize the environment?
2. What are the optimal learning parameters that compute efficient vehicle trajectories?
3. How to design an efficient reward function that encompasses different road and congestion metrics in calculating the near-optimal paths?
4. How does the type of learning algorithms and exploration strategies affect learning performance?

1.3 Methodology

1. Build an efficient reward function which captures the driving environment and accelerates the learning speed.
2. Evaluate and compare the performance of Q-learning and Sarsa in conjunction with two exploration strategies: (a) e-greedy, and (b) Softmax.

3. Study the impact of the learning rate and the discount factor on the quality of the computed solutions.

1.4 Structure of the Thesis

Following this introductory chapter, this thesis is structured in 6 main chapters that briefly describe now:

- Chapter 2 provides a systematic review of the previous research on which the work is based.
- Chapter 3 introduces background on basic mathematical formalism for Reinforcement Learning, which is the Markov Decision Processes. The chapter also discusses RL methods and exploration strategies.
- Chapter 4, the system design and methods used during the testing, is presented in this chapter.
- Chapter 5 discusses the results obtained from the experiments as well as the comparative evaluation of proposed methods.
- Chapter 6 summarize the conclusions and present ideas for future work.

Chapter 2: Literature Review

2.1 Reinforcement Learning

Reinforcement learning is a general-purpose learning framework that can address many important aspects of Artificial Intelligence (AI). The Tamilselvi et al. (2011) work has implemented Reinforcement learning, Q-Learning algorithm for mobile robot navigation in an indoor environment. The robot was operated in grid (10×10) environment with different positions in the environment to find the optimum path between source and destination.

Sichkar (2019) deployed and evaluated the performance of Q-learning and SARSA algorithms for guiding the mobile robot to the desired goal while avoiding obstacles. Experiments were performed in the 2-dimensional virtual environment. The obtained results showed differences between the two Reinforcement Learning algorithms in learning time and the methods of building a path to avoid obstacles until reach a destination point.

Path and motion planning for a robot in the real world was presented in Babu et al. (2016). The main objective of this work is to develop an autonomous robot that uses Q-learning for navigation in an unknown environment. These were achieved by calculating the shortest path from the current state to the goal state through analyzing the captured images of the environment.

The work in Gao et al. (2019) utilized a new global planning algorithm combined with Q-Learning to find the global path for robots. The experiments were conducted in both physical and simulation environments with various scenarios. To evaluate the effectiveness of the proposed algorithm, authors compared their algorithm with the Best First Search (BFS) and Rapidly-exploring Random Trees (RRT)

algorithm. The analyzed results show the shorter and smoother paths obtained by the proposed algorithm compared to the BFS algorithm and RRT algorithm.

A novel end-to-end mobile robot path planning using deep reinforcement learning is proposed in Xin et al. (2017). Using the original visual perception without any hand-crafted features and feature matching, the suggested planning approach can decide the optimal action to make the mobile robot reach the target point while avoiding obstacles.

The work presented in Luo et al. (2018) proposed the Deep-Sarsa approach for autonomous path planning as well as avoiding obstacles for Unmanned Aerial Vehicles (UAVs). The model is trained in a grid environment before being deployed in an environment in ROS-Gazebo for UAVs. Results of the experiments show the success of the trained Deep-Sarsa model in guiding the UAVs to the target without any collisions.

2.2 Road Traffic Congestion Systems

Researchers have paid considerable attention to the issue of traffic congestion in recent years. Many road traffic congestion systems have been introduced using different techniques to manage the traffic challenge in cities and overcome the limitation of the traditional systems. Cooperative Intelligent Transport Systems or C-ITS (Festag, 2014; Sjoberg et al., 2017) is a new transportation system that allows vehicles to communicate with other vehicles (V2V) and infrastructure (V2X) such as traffic signals and roadside, that are fitted with the same system at a carrier frequency of 5.9 GHz. It provides intelligent solutions for a variety of road traffic problems by applying advanced technologies and service levels via transmit real-time traffic information using wireless technology. Drivers then receive alerts about upcoming

hazards and act accordingly in order to increase traffic safety and efficiency in road transport.

The work in Rahman et al. (2014) presented a traffic management system based on Wireless Sensor Networks (WSN) with a dynamic mathematical model for the management of road traffic at important city intersections. This system detects the road congestion and broadcasts the information to drivers so that they can take a detour to avoid the traffic.

In Jayapal and Roy (2016) authors proposed a mobile-enabled VANET technology to reduce traffic congestion and divert vehicles. The system is a distributed, collaborative traffic congestion detection and dissemination system. It uses smart phones of drivers that equipped with a Traffic App to detect location through Geographic Position based System (GPS) to be sent to a remote server that predicts traffic congestion. Once congestion is confirmed, it is passed on to the end user's phone through RSUs.

In Akhtar et al. (2020), the authors proposed a congestion level-based dynamic traffic management system using IoT. The system regulates the duration of traffic lights according to the real-time congestion level measured at the road crossings by using ultrasonic sensors. Similarly, Javaid et al. (2018) has provided a solution to optimize traffic flow on roads by exploiting the concepts of IoT and Artificial Intelligence together.

The work in Walraven et al. (2016), proposed a new method to address the issue of traffic congestion by using reinforcement learning. It formulates the traffic flow optimization problem as a Markov Decision Process and uses Q-learning to find policies to assign speed limits of the vehicles that are allowed on a highway, such that traffic congestion is reduced. This can be estimated according to the attributes of the

highway as well as demand volumes filling the highway and predictions regarding future traffic conditions.

Deep Reinforcement Learning has been also studied to address one of the most pressing problems in road traffic management, namely that of Traffic Light Optimization (TLO). The TLO problem aims to improve traffic light timings in order to optimize the overall travel time of the vehicles that traverse the road network and reduce fuel consumption. In Coskun et al. (2018) authors introduce a new reward function that takes the traffic flow and traffic delay into account to provide a solution to traffic light optimization which in turn decreases travel time. They use both Deep Q-Learning and Policy Gradient approaches to solve the resulting reinforcement learning problem.

In Liang et al. (2019), a deep reinforcement learning, in particular, Double Dueling Deep Q Network (3DQN) was proposed to decide the duration of the traffic signals based on the collected data from different sensors and vehicular networks. In the model, the states are two-dimension values with the position of vehicles and speed information. The actions are modeled as a Markov decision process and the rewards are the cumulative waiting time difference between two cycles.

Van der Pol and Oliehoek (2016) presented the learning control policies for traffic lights by the use of the DQN algorithm with transfer planning as a promising and scalable multi-agent approach to deep reinforcement learning. The combination between DQN and the transfer planning approach allows for faster and more scalable learning. The obtained results show how the proposed approach reduces the travel times of vehicles compared to earlier work on reinforcement learning methods for traffic light control.

2.3 Route Planning Algorithms

Dijkstra (1959), proposed a static algorithm to find the path with the lowest cost (i.e., usually refers to the shortest path) from the source node to all other nodes without considering external parameters such as congestion, vehicle amount, etc. In Zhan and Noon (1998), authors state that it is worthwhile to consider the Dijkstra algorithm to find the shortest path from the one-to-one shortest path problem since this algorithm is terminated as soon as the destination node is permanently labeled which implies that the shortest path is found. However, the optimal route is not always the shortest path between two nodes due to the continuous changes in the road traffic network. Thus, vehicle routing optimization should take into account the latest state of the transportation network and make real-time adjustments in order to arrive at their destination in the shortest time possible.

The A* route planning algorithm employs a heuristic function instead of the optimized search mechanism used by the Dijkstra algorithm. Dere and Durdu (2018) proposed the use of the A-Star algorithm for finding the shortest path between a starting-point and ending-point on the Google Map that segmented as grid-cells. In addition, the traffic intensity of various roads was constructed on the map so that the algorithm takes the traffic density into consideration when it finds the shortest route.

A Vehicular Ad-hoc Network (VANET) based A* (VBA*) for enhanced route planning is designed in Chang et al. (2013). The proposed solution aims to dynamically calculate the optimum route that meets the shortest travel time or the least fuel consumption using information from Google Map.

Nafi et al. (2014) proposed a predictive road traffic management system named PRTMS based on the Vehicular Ad-hoc Network (VANET) architecture. The PRTMS

uses a modified linear prediction algorithm to estimate the future traffic intensities at intersections point on road. The vehicles are re-routing based on this prediction to reduce the congestion level and minimize the traveling time of the individual.

In Touluni et al. (2014), a new approach based on VANETs has been proposed to addresses the problem of the optimal path in road networks in order to reduce travel time and fuel consumption. More specifically, the authors applied Dijkstra's algorithm to determine the optimal route from the current vehicle position to the destination point based on the analyzed collected traffic data in real-time. Having this data will not only reduce the travel time but also avoid congestion queues in more efficient and optimal use of existing road infrastructure. The experiment has been conducted by using SUMO as a platform to provide dynamic simulation Traffic Control Interface (TraCI) to allows the change of scenario when running.

Machine learning techniques are used in Chhatpar et al. (2018) to predicts the traffic densities in a given area. In particular, the authors used Supervised Learning techniques such as Back Propagation Neural Network (BPN) via an android application which makes use of real-time traffic data and provides a predictive analysis of traffic in an offline mode. Based on this information, the best route from source to destination is provided in order to reduce the congestion on roads.

A group routing suggestion algorithm is proposed in Sang et al. (2017) based on Markov Decision Process (MDP) (Smelser & Baltes, 2001). Instead of optimizing the routing path for individual vehicles, a routing group of vehicles will be suggested based on vehicles' or drivers' similarities in a specific urban's transportation environment. The authors discussed the design of the general flow of group routing method and studied how it is going to work with their proposed prototype.

The authors in Mejdoubi et al. (2020), applied a reinforcement learning approach based on VANET to enable efficient flow management by providing optimal paths suggestion and minimizing the total traveling time for drivers. In particular, they employed Q-learning to learn the best action to take in various traffic situations. They also highlight vehicle-to-vehicle and vehicle-to-roadside unit communications in order to collect and exchange the real-time traffic status.

Koh et al. (2018) conducted an experience to perform a reinforcement learning approach to optimize the route of a single vehicle in a network. The proposed experience uses an open-source simulator called Simulation of Urban Mobility (or SUMO for short). It offers promising results in finding the optimal route to reach the destination and avoiding the congestion path.

In Koh et al. (2020), a novel Deep Reinforcement Learning (DRL) based vehicle routing optimization method was proposed to re-route vehicles to their goals in complex urban transportation networks. A nine realistic traffic scenarios are simulated using the SUMO simulator to test the proposed navigation method.

The work of Lee et al. (2020), proposed a framework for an Electric Vehicle Charging Navigation System (EVCNS) based on model-free Deep Reinforcement Learning (DRL). This framework aims to reduce the total travel time of Electric Vehicles (EV) charging requests from a start point to the end point by selecting the optimal route and charging station taking into account the continuous changing of traffic conditions and unknown future requests.

Authors in Geng et al. (2020) applied a route planning algorithm based on Deep Reinforcement Learning (DRL) for pedestrians. They plan the route by predicting pedestrian flow in the road network and the travel time consumption was used as the metric. This experiment was conducted using an intelligent robot on a virtual map

where the robot acts as a pedestrian and assuming that it does not require any prior knowledge of road networks.

Chapter 3: Reinforcement Learning

3.1 Machine Learning

Machine learning is a branch of Artificial Intelligence (AI) focused on developing applications with the ability to learn from data and improve automatically through the experience without being explicitly programmed (Ayodele, 2010). The learning algorithms of ML are organized into a taxonomy based on the amount and type of supervision they get during training. Figure 1 shows common algorithms types.

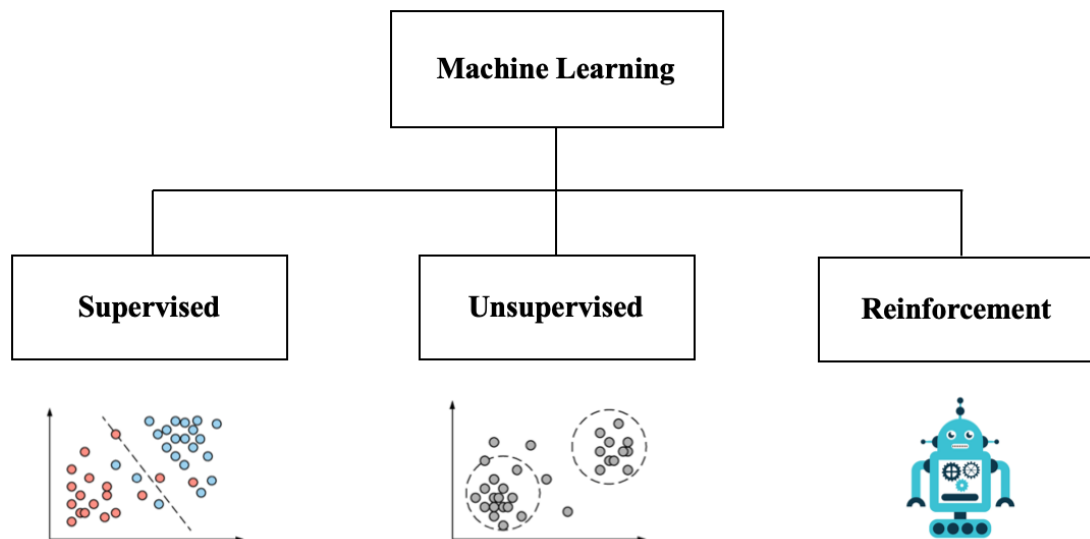


Figure 1: Machine Learning Types

Supervised Learning: is the task of feeding the algorithm with the training data that includes the desired solutions, called labels. Typical supervised learning tasks could be a classification if the output is a class or category of the data such as email spam classification. Another typical task is regression, where the expected result from the model is a numerical value, such as the price of a car.

Unsupervised Learning: is based on the absence of any supervisor or training data. In other words, the training data is unlabeled which means that the system must learn while not receiving any feedback. In this case, an unsupervised learning technique is useful when it's necessary to learn how a set of elements can be grouped based on their similarity (i.e. clustering).

Reinforcement learning: is a learning system, called an agent in this context, evaluates its performance according to the feedback responses and reacts accordingly. More precisely, the agent observes the environment, selects and performs actions, then gets feedback called reward which can be either positive or negative. This learning strategy follows the method of “trial and error” as the agent is not explicitly told which action to take to receive positive rewards. It must then continually interact with the environment and learn by itself the best strategy, called a policy, with regard to the rewards it gets. This is summarized by Figure 2.

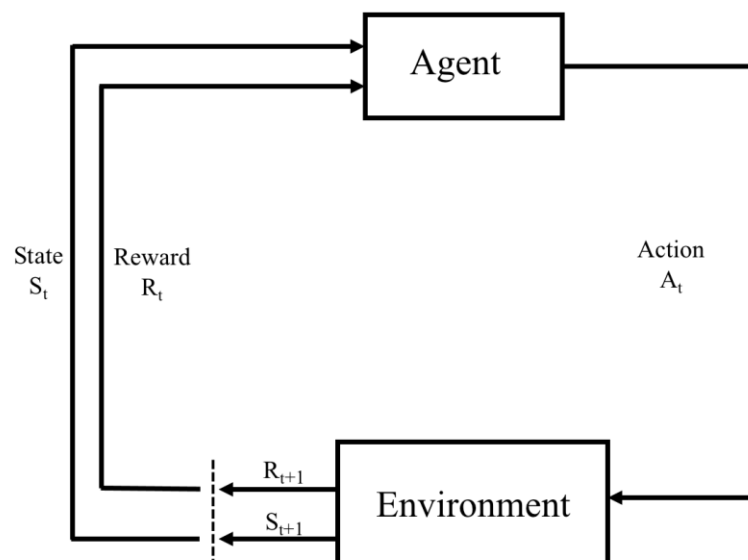


Figure 2: Reinforcement learning schema

3.2 Markov Decision Process

Markov Decision Process (Puterman, 1990), referred to as MDP, offers a standard formalism for describing sequential decision making.

Definition 3.2.1: A Markov decision process is a tuple $\langle S, A, T, R \rangle$ (Van Otterlo & Wiering, 2012) in which:

- S is a finite set of states,
- A is a finite set of actions,
- T is a transition function defined as $T: S \times A \times S \rightarrow [0,1]$,
- R is a reward function defined as $R: S \times A \times S \rightarrow \mathbb{R}$

At each time step $t = 0, 1, 2, \dots$ the decision-maker, called an agent receives some representation of the environment's state $s_t \in S$. Based on this state, the agent performs an action $a_t \in A$ which gives the pair of state-action (s_t, a_t) . The time is then incremented to the next time step $t+1$ and the environment changes such that it is in a next state $s_{t+1} \in S$. At this time, the agent gets an immediate numerical reward denoted by r_{t+1} for the action a_t taken from state s_t .

The probability to end up in s_{t+1} is influenced by the chosen action. In math, it is given by the state transition function. Precisely, the state transitions of a Markov decision process satisfy the Markov property: the next state s_{t+1} is dependent only on the current state s and the performed action a . Accordingly, the reward function R can be defined as $R: S \times A \times S \rightarrow \mathbb{R}$ (Van Otterlo & Wiering, 2012).

The goal of an agent in an MDP is to maximize its cumulative rewards. Indeed, there is a way to aggregate and formalize these cumulative rewards, a concept of expected return is introduced to sum all rewards obtained by the agent at a given time step. Mathematically, the return G at time t can be define as (Fragkiadaki, 2018):

$$G_t = R_{t+1} + R_{t+2} + \dots + R_T \quad (3.1)$$

However, in some type of task (i.e. continuing tasks) the agent continues to interact in the environment without limit which makes the final time step $T = \infty$ in Equation 3.1, and therefore the return itself could be infinite. To avoid infinite returns in continuing tasks, the discount factor $0 \leq \gamma < 1$ is used to influence the future rewards, in which the rewards obtained later are discounted more than rewards obtained earlier. This function can be defined as (Fragkiadaki, 2018):

$$G_t = R_t + \gamma R_{t+1} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad (3.2)$$

Where t and γ represent the time step and discount factor, respectively.

3.3 Policies and Value Functions

The selection of actions is modeled as a map called strategy or policy. A policy is an agent's behavior function $\pi: S \rightarrow A$, where it specifies the action that the agent should take based on the current state. In order to determine this action, the agent needs to estimate how good it is for an agent to be in a certain state, or how good it is for the agent to perform a given action in a particular state. The notion of "how good" a state is the value function. The value of a state s under policy π , denoted $V\pi(s)$ is the expected sum of rewards that the agent will receive at any given state s while following a policy π . The value function, $V\pi(s)$ for policy π is given by (Rastogi, 2017):

$$V_{\pi}(s) = E_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s \right\} \quad (3.3)$$

Where E is the expectation, γ is the discounting factor, R_t is the reward at time t and S_t is the state at time t . It can define, in a similar way, the action-value function, also known as the Q-function, as the expected sum of rewards while taking an action a in state s and, thereafter, following policy π . Mathematically, it define $Q_{\pi}(s, a)$ as (Rastogi, 2017):

$$Q_{\pi}(s, a) = E_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a \right\} \quad (3.4)$$

3.4 Optimal Policy

The goal for any given MDP is to find the optimal policy that maximizes the cumulative rewards. Concerning return, a policy π is considered to be better than another policy π' if the expected return of that policy is greater than the expected return of for all states, which implies, $V_{\pi}(s) \geq V_{\pi'}(s)$ for all $s \in S$. Thus, the optimal policy π^* can be computing by defined the optimal value function $V^*(s)$ (Rastogi, 2017):

$$V^*(s) = \max_{\pi} V_{\pi}(s), \quad \forall s \in S. \quad (3.5)$$

Similarly, the optimal action value function, $Q^*(s, a)$ can be defined as (Rastogi, 2017):

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a), \quad \forall s \in S, a \in A. \quad (3.6)$$

One fundamental property of both V^* and Q^* is that they satisfy certain recursive properties. Hence, the expression in Equations 3.7 and 3.8 can recursively defined in a special form called Bellman Equation (Rastogi, 2017):

$$V_*(s) = \max_a \sum_{s'} p(s'|s, a) [R(s, a, s') + \gamma V_*(s')] \quad (3.7)$$

$$Q_*(s, a) = \sum_{s'} p(s'|s, a) [R(s, a, s') + \gamma \max_{a'} Q_*(s', a')] \quad (3.8)$$

3.5 Q-learning and Sarsa Algorithms

After illustrating the key concepts and ideas behind Markov Decision Processes, the term of Reinforcement Learning (Sutton & Barto, 1998) can be introduced to solve the MDPs.

RL algorithms can be found implemented in robot control and board games like Tic-tac-toe, and chess. In these kinds of problems, the agent is modeled to learn through frequent interactions with their environment and the returns signal of these interactions; it learns from experience. These tasks deal with one learning agent only (single-agent). However, various real-world decision problems such as swarm robots and traffic are inherently composed of several tasks which demand models with multiple agents. Multi-agent models can simplify the complex problem by dividing knowledge among the agents.

Popular methods in RL are Q-Learning (Ho et al., 2006), and State Action Reward State Action (SARSA) (Chen & Wei, 2008). Q-learning is a model-free reinforcement learning method used for learning the optimal policy to select the best action in a Markov Decision Process.

More specific, Q-Learning estimate Q-Values for each state-action combination under policy π and update them frequently during the training process based on the Formula 3.9. Hence, these values describe the quality of an action taken from that state.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)] \quad (3.9)$$

In Q-Learning, a Q-Table is built to store Q-Values for all possible combinations of state and action pairs, which is a matrix with the vertical axis represents the states and the horizontal axis represents the actions. Table 1 shows an example of Q-table.

Table 1: Example of Q-Table

Q-Table		Actions				
		Action 1	Action 2	...	Action n-1	Action n
States	State 1	0.1239	4.2867	...	6.5434	0.4532
	State 2	2.3301	2.9863	...	2.3487	4.2857
	7.3265	1.7654
	State n-1	7.1122	1.9876	...	2.5478	4.0375
	State n	0.1211	2.1274	...	3.8712	7.5941

Q-Learning can be broken down into steps that make things much clearer. This is what it will seem to be:

1. Initialize all Q-values in the Q-table to 0.
2. For each time-step in each episode:
 - 2.1 Pick an action a , from the set of actions defined for that state (considering the exploration-exploitation trade-off)
 - 2.2 Perform action a
 - 2.3 Observe reward R and the next state s'
 - 2.4 Update the Q-value function using the Formula 3.9.

Similar to Q-learning, SARSA is a model-free RL technique that does not learn the policy function of the agent explicitly. The main difference between SARSA and Q-learning is that Q-learning is an off-policy method, while SARSA is an on-policy method. The effective difference between the two algorithms happens in the step where the Q-table is updated. The Q-Learning explores the action-values function (Q-value) for all possible actions in the given state then selects the maximum action value among them. On the other hand, SARSA uses the action-value function for the action a_t in state s_t according to the following updated formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (3.10)$$

Sarsa's steps can be summarize as:

1. Initialize all Q-values in the Q-table to 0.
2. For each time-step in each episode:
 - 2.1 Pick an action a , from the set of actions defined for that state (considering the exploration-exploitation trade-off)
 - 2.2 Perform action a
 - 2.3 Observe reward R and the next state s'
 - 2.4 Update the Q-value function using the Formula 3.10.

3.6 Exploration-Exploitation Trade-off

As previously stated, the agent should follow an optimal policy that dictates the selection of action a_t in the state s_t . In Q-learning, there exists a tradeoff between selecting random actions with a uniform distribution over the action space or selecting the currently expected optimal action. These two opposite behaviors are called exploration and exploitation tradeoff (Thrun, 1992; Wiering, 1999; Yahyaa, 2015). Initially, the agent must choose mainly random actions, regardless if they are not the

best possible actions. This enhances the agent to explore parts of the state space and actions that might be more rewarding than the ones that have not been encountered before. As the learning progresses, the agent will start to exploit the current knowledge which probably converged to a policy that is close to the optimal one in order to maximize the obtained reward. However, excessive exploration yields a lower accumulated reward, whereas excessive exploitation will trap the agent in a local optimum. Thus, it is important to find a balance between these two extremes. Popular existing strategies that attempt to deal with this dilemma are $\epsilon - greedy$ method and *softmax*.

The $\epsilon - greedy$ strategy uses $0 \leq \epsilon \leq 1$ as a parameter of exploration where the probability to select random actions decreases linearly from 1 to 0 (Tijms et al., 2016).

$$a_t = \begin{cases} a^* = \operatorname{argmax} Q_t(a) & \text{with probability } 1 - \epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases} \quad (3.11)$$

With a probability $(1 - \epsilon)$, the agent will choose the optimal action a^* that indicates the highest Q-value for the current state from the Q-table, while it will choose action randomly if the probability is (ϵ) . One drawback of $\epsilon - greedy$ exploration is that non-optimal actions are all considered the same during exploration. Therefore, it is better to assign a probability to the actions to be chosen that translates to its estimated value. One way to do that is by using a *Boltzmann or softmax* exploration that uses the Gibbs or Boltzmann distribution function. At each time step t , the agent will select an action a with a probability (Tijms et al., 2016):

$$\pi(s_t, a) = \frac{e^{Q_t(s_t, a)/T}}{\sum_{i=1}^m e^{Q_t(s_t, a^i)/T}} \quad (3.12)$$

where $\pi(s_t, a)$ denotes the probability when the agent selects action a in state s_t and $T \geq 0$ is a positive parameter called temperature that controls exploration and exploitation tradeoff. When $T = 0$ the agent does not explore at all, instead it always acts greedily and selects the strategy corresponding to the maximum Q-value. Whereas when $T \rightarrow \infty$ the agent selects random actions.

Chapter 4: System Design

4.1 Road Traffic Model

Figure 3 shows an example of the road network, where all possible positions for a vehicle on the road are represented by nodes. The N contains nodes that represent junctions J which involves a crossing over of two or more road segments R , where $j \in J$ and $r \in R$, thus N is defined as $N = (J, R)$.

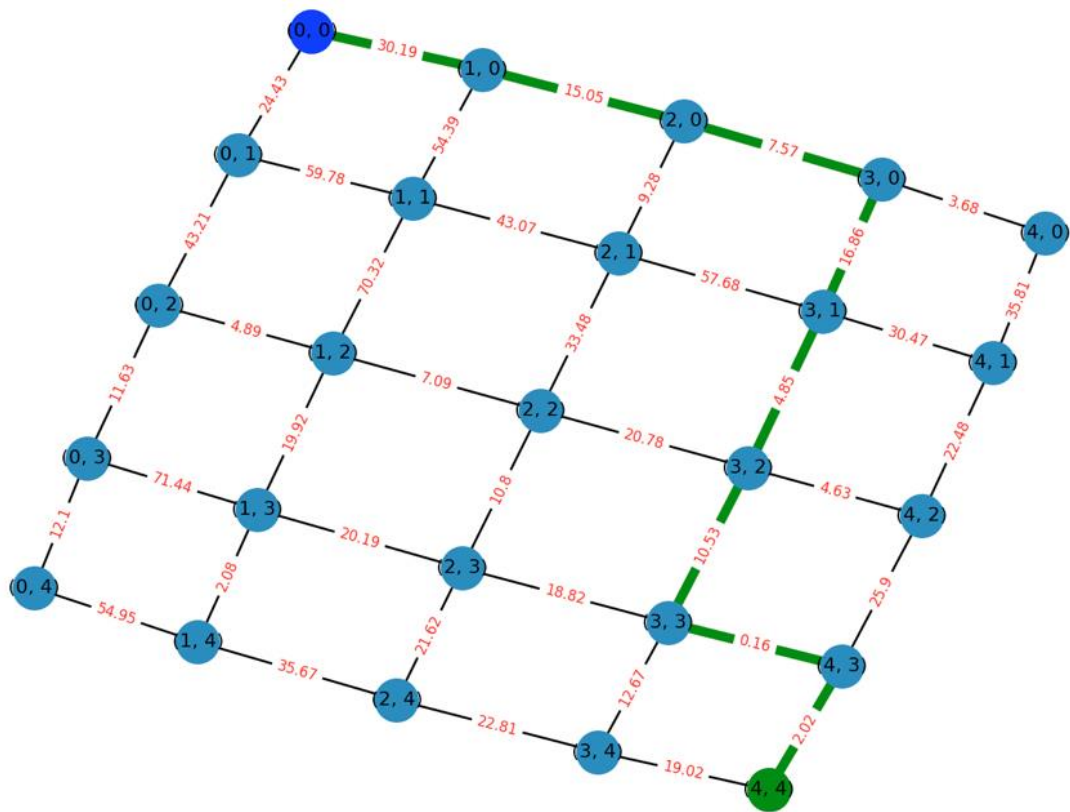


Figure 3: Example of road network environment

Assuming each road segment connected between two junctions j_1 and j_2 , the road segment can be defined as $r = (j_1, j_2)$. Since the road segments R depend on the number of rows NR and the number of columns NC , the R can be calculated using this equation:

$$R = (NC - 1 * NR) + (NR - 1 * NC) * 2 \quad (4.1)$$

After explaining how the junctions, road segments, and road networks are defined, the congestion can now be generated. In other words, there is a need to define how vehicle arrives at goal by avoiding the traffic density to minimize the travel time it takes. When it comes to how the vehicle arrives, it refers to which time step and which segment it has to select in the environment. For this purpose, a new parameter is defined called traffic congestion, denoted *load*, that serves as the number the vehicles generated for each road segment.

Basically, the congestion can be generated for a whole region or a specific road segment. When the vehicle is on the road, it collects information continuously about the state of traffic density of the road segment traveled through. Hence, a number of vehicles are distributed among the whole road segments and generate extreme traffic load on particular segments based on Algorithm 1, where N is the number of vehicles.

Algorithm 1: Generating Traffic Load

```

load = defaultdict(lambda: 5)
for i from 0 to Number_of_columns - 3
    for s in (Number_of_columns/2-1, Number_of_columns/2)
        r, c = (i, s)
        j0 = r, x
        j1 = r, x-c
        j2 = r, x+c
        j3 = r+1, c
        load[(j0, j1)] = load[(j0, j2)] = load[(j0, j3)] = N
        load[(j1, j0)] = load[(j2, j0)] = load[(j3, j0)] = N
    end
end

```


Accordingly, the vehicle itinerary will be the sum of all passed road segments from the starting point to the destination point. The traffic load will be calculated as the sum of vehicles available in these segments. After calculation, the vehicle will compare all routes results to find the best one to guide it through less traffic load. The lower the result of steps and traffic load, the lower the travel time it takes by vehicle to reach its goal.

Since the selected path may be either the shortest path with high load and vice versa, a new factor is defined called "weight," denoted as w , where it indicates the importance weight giving to the path length and traffic load in the measure. Thus, they can be calculated it in one formula:

$$F = w * \sum_{i=0}^N R + (1 - w) * \sum_{i=0}^N V \quad (4.2)$$

Where $\sum_{i=0}^N R$ is the sum of passed road segments in the selected path, and $\sum_{i=0}^N V$ is the sum of vehicles in this path.

4.2 Reinforcement Learning

Reinforcement learning strategy that concerns learning agents to maximize the cumulative reward they receive from the environment. RL is modeled as a Markov Decision Processes (MDPs), which is a mathematical framework that models sequential decision-making problems. As previously stated, the MDP consist of a finite set of states S , a finite set of actions A , transition function T which is a probability of making transitions between states, and reward function R . Thus, the road traffic congestion problem is formulated as a Markov Decision Process (MDP).

4.2.1 State Space

The set of environmental states S is defined as the finite set $[s_t, \dots, s_N]$ where N is the size of the state space, i.e. $|S| = N$. As stated earlier, junctions represent all possible locations vehicle could inhabit at the road. These positions are called "states" in the reinforcement learning system, that present an agent in a particular instance of time. Thus, all junctions are mapped as states in the system. Figure 4 present a virtual environment that has been divided into cells, in which obstacles and congestions are occupied some of these cells. Each cell represents a state of the road with information about what is in the cell at that moment. If the agent falls into the obstacle, it counts as a collision. While if the agent falls into congestion, it will receive a penalty with a negative value.

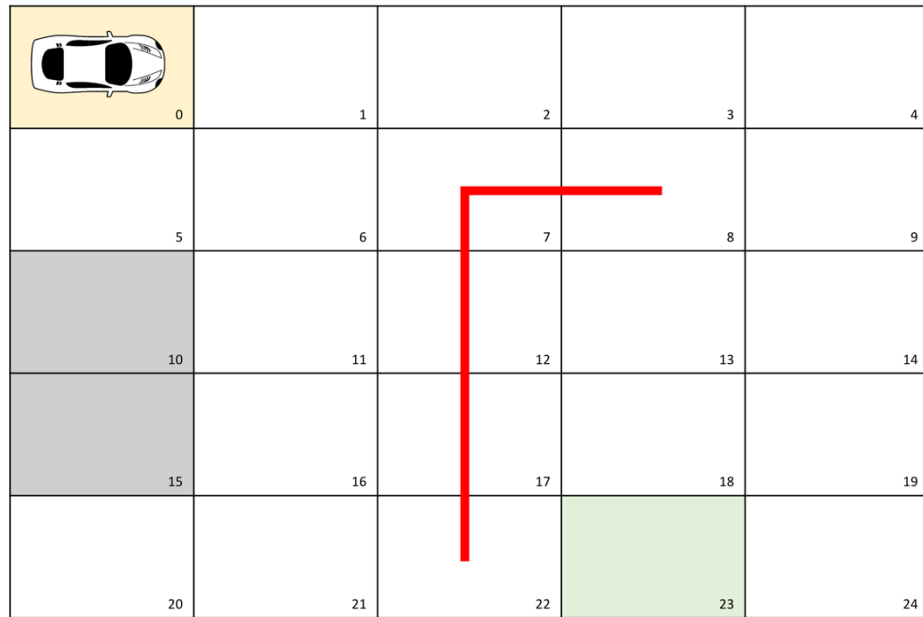


Figure 4: A virtual environment divided into cells

Assuming that the position of the vehicle obtained through Vehicular Ad-hoc Network (VANET) and the vehicle will transmit from the current state s to a new state s' based on a "discrete" action being passed. At each time step, the vehicle will pass through a segment which will be occupied with a number of vehicles $N \vee r$.

4.2.2 Action Space

Now there is a need to define the possible actions that the vehicle can take. It is obvious that a particular action should lead to one move, and vice versa; one move is the result of only one action. As shown in Figure 5, the vehicle can move diagonally by choosing between moving forward, moving backward, moving left, moving right. However, in certain cases, some actions can be "impossible". Precisely, if an action's corresponding move is forbidden in the system, the vehicle will disregard this action by considering another one. A forbidden move is a situation where the vehicle attempts to move beyond the walls or boundary of the environment.

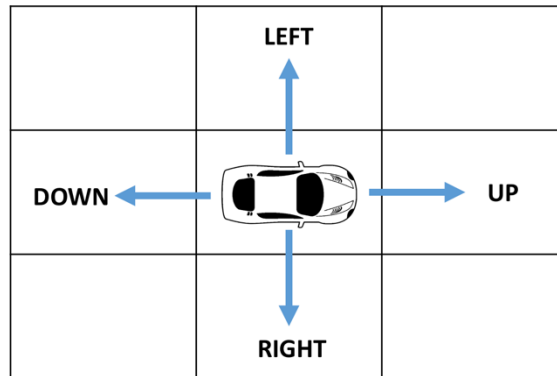


Figure 5: Four possible actions

The key purpose of the system is to reduce the travel time of vehicles by selecting the optimal path with the least traffic congestion. One way to achieve this intention is by letting the agent learn how to avoid collision with obstacles in the

environment. It should not choose a move whose outcome is an accident. As an alternative, it will select the move in their preference order that leads to reaching the destination.

The strategy of selection actions could be either exploration by selecting random action or exploitation through choosing the action with the highest Q-value for its current state from the Q-table. To get a balance between exploitation and exploration, two widely strategies are used in this work; ϵ – *greedy* and *softmax*.

At the first stage, it is necessary to investigate the environment as best as possible by choosing a random action. As the vehicle moves from one state to another, the Q-table will be updated based on the obtained values from the selected actions. Then, the vehicle exploits the knowledge that it has found for the current state s by choosing the most prioritize action that maximizes $Q[s, a]$.

4.2.3 Reward Function

In Reinforcement Learning algorithms, the purpose for the agent is to learn an optimal or nearly-optimal policy that maximizes the cumulative rewards. The state reward function is defined as $R: S \rightarrow \mathbb{R}$, and it identifies the reward obtained by the agent based on the taken action. R is the most important factor in the RL system since it provides feedback to a reinforcement learning model about the performance of the chosen actions to converge to an optimal policy. Hence, defining an appropriate reward value is critical to guide the learning process accurately, which in turn helps to take the best action policy.

A reward function is designed that encompassing different road and congestion metrics in calculating the near-optimal paths. When the vehicle passes across road segments, it will observe a load that represents the negative reward (penalty). Thus,

the vehicle's objective is to move towards junctions' states by selecting the optimal road segments to its destination in order to reduce the travel time. The reward function is designed as:

- If the vehicle reaches the goal, it will receive a reward of 500.
- If the vehicle crashed into a wall or obstacle it will be given a penalty of -500.
- At each time of step, the vehicle will receive a penalty of $-N V r$ for each passed segment, where $N V r$ indicates the number of vehicles in one segment r .

The possible outcomes are called goal; if the vehicle reaches its goal and it called obstacles; if it crashes with obstacles. In case the agent reaches one of these outcomes, the episode will be terminated and the reward value will be given immediately. While the agent moves to cells occupied with low or large congestion, the reward function will be calculated and the agent will complete moving until the episode is done. The explanation of the pseudo-code is presented in Algorithm 2.

Algorithm 2: *Reward Function*

```

If state = goal then
    |   reward = 500
    |   done   = True
elif state = obstacle then
    |   reward = -500
    |   done   = True
else
    |   reward = -load
    |   done   = False
end

```

4.2.4 RL Algorithms

Most of the previous works used the q-learning and sarsa algorithms in a successful way to deal with the problem of robot path planning and navigation in either simulated or real environments. Thus, Q-learning, and Sarsa will be use, with two exploration strategies $\epsilon - greedy$ and *softmax*. Then, a performance comparison of each algorithm based on different criteria. The algorithms of each technique are illustrated below:

Algorithm 3: Q-learning- $\epsilon - greedy$

```

Initialize  $Q(s,a), \forall s \in S, a \in A(s)$ , arbitrarily
for episode = 1 to M do
    Initialize s
    for step = 1 to T do
        Choose a from s using policy (3.11)
        Excute a, observe reward  $r_t$  and next state  $s'$ 
         $Q(s,a) \leftarrow Q(s,a) + \alpha [r + \gamma \max_a Q(s',a) - Q(s,a)]$ 
         $s \leftarrow s'$ 
    end
end
end

```

Algorithm 4: Q-learning-softmax

```

Initialize  $Q(s,a), \forall s \in S, a \in A(s)$ , arbitrarily
for episode = 1 to M do
    Initialize s
    for step = 1 to T do
        Choose a from s using policy (3.12)
        Excute a, observe reward  $r_t$  and next state  $s'$ 
         $Q(s,a) \leftarrow Q(s,a) + \alpha [r + \gamma \max_a Q(s',a) - Q(s,a)]$ 
         $s \leftarrow s'$ 
    end
end
end

```

Algorithm 5: Sarsa- ϵ - greedy

```

Initialize  $Q(s,a), \forall s \in S, a \in A(s)$ , arbitrarily
for episode = 1 to M do
    Initialize s
    for step = 1 to T do
        Choose a from s using policy (3.11)
        Excute a, observe reward  $r_t$  and next state  $s'$ 
        Choose  $a'$  from  $s'$  using policy derived from Q
        (e.g.,  $\epsilon$ -greedy or softmax)
         $Q(s,a) \leftarrow Q(s,a) + \alpha [r + \gamma Q(s',a') - Q(s,a)]$ 
         $s \leftarrow s'; a \leftarrow a'$ 
    end
end
end

```

Algorithm 6: Sarsa-softmax

```

Initialize  $Q(s,a), \forall s \in S, a \in A(s)$ , arbitrarily
for episode = 1 to M do
    Initialize s
    for step = 1 to T do
        Choose a from s using policy (3.12)
        Excute a, observe reward  $r_t$  and next state  $s'$ 
        Choose  $a'$  from  $s'$  using policy derived from Q
        (e.g.,  $\epsilon$ -greedy or softmax)
         $Q(s,a) \leftarrow Q(s,a) + \alpha [r + \gamma Q(s',a') - Q(s,a)]$ 
         $s \leftarrow s'; a \leftarrow a'$ 
    end
end
end

```

Then, the simulation parameters have to define to characterize the road network environment and the learning parameters that influence the performance of each proposed algorithms. These parameters can be independently modified to achieve the best performances on computing the optimized vehicle routes. The simulation and learning parameters are presented in Table 2.

Table 2: Simulation and learning parameters

Simulation Parameters	Learning Parameters
State space	Reward
Action space	Number of episodes
Number of goals	Learning rate/alpha
Number of objects	Discount factor/gamma
Number of segments	Epsilon and Temperature
Number of congestion segments	Epsilon decay
Number of vehicles	

Chapter 5: Evaluation and Performance Analysis

Inference about the performance and validity of proposed algorithms was conducted with a set of experiments described in this chapter. Besides, a comparison of the performance analysis is thoroughly described for each experiment.

5.1 Experimental Environment

To evaluate the performance of each proposed algorithms, various experimental scenarios were conducted in 2-dimensional virtual environments; 6×6 , 10×10 , and 20×20 . The program was written in Python 3 with specific libraries. Table 3 shows the configuration of the simulated parameters for considered environments. For a simulation, each of these parameters can be modified individually. However, some of them must be consistent: for example, the maximum number of congestion segments that can fit in the maze can't exceed the total number of road segments R .

Table 3: Simulation parameters of 6×6 , 10×10 , and 20×20 , respectively

Simulation Parameters		Simulation Parameters		Simulation Parameters	
State Space	36	State Space	400	State Space	100
Action Space	4	Action Space	4	Action Space	4
Number of Goals	1	Number of Goals	1	Number of Goals	1
Number of Objects	5	Number of Objects	13	Number of Objects	12
Number of Segments	120	Number of Segments	1520	Number of Segments	360
Number of Load Segments	12	Number of Load Segments	204	Number of Load Segments	84
Number of Vehicles	720	Number of Vehicles	16780	Number of Vehicles	3900

The maps vary in size, placement of the goal, number of blocks, vehicles, and road congestion. Figure 6, Figure 7 and Figure 8 present the three environments in which experiments were conducted. The agent's goal is to learn a near-optimal route from the start junction, yellow, to the goal junction, green. It has to avoid the gray junctions representing the obstacles, and the extreme traffic load appears as red segments. The blue and green arrows present the shortest path and least traffic congestion path, respectively.

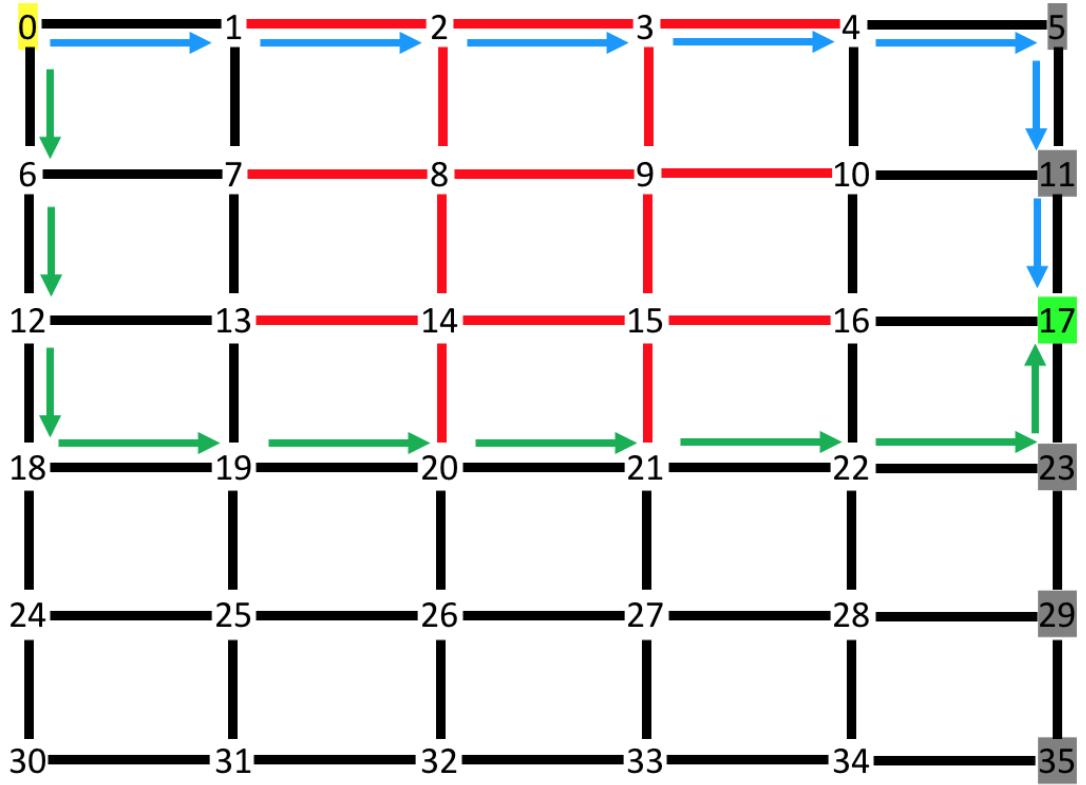


Figure 6: 6×6 map

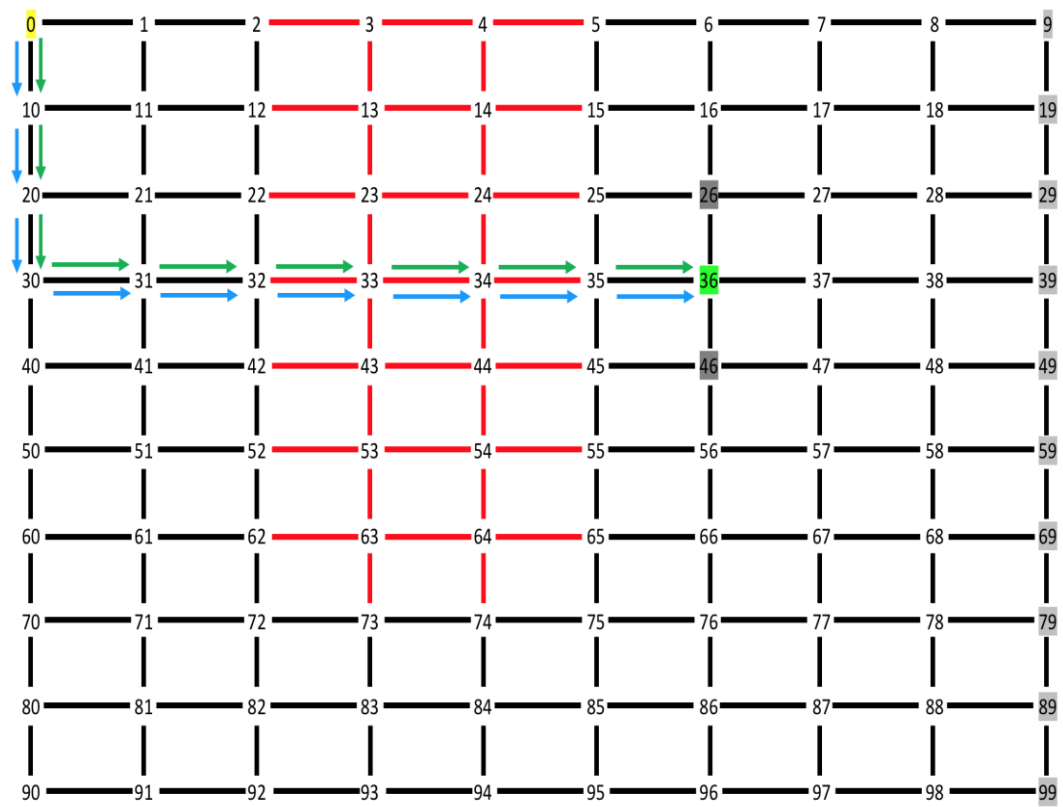


Figure 7: 10×10 map

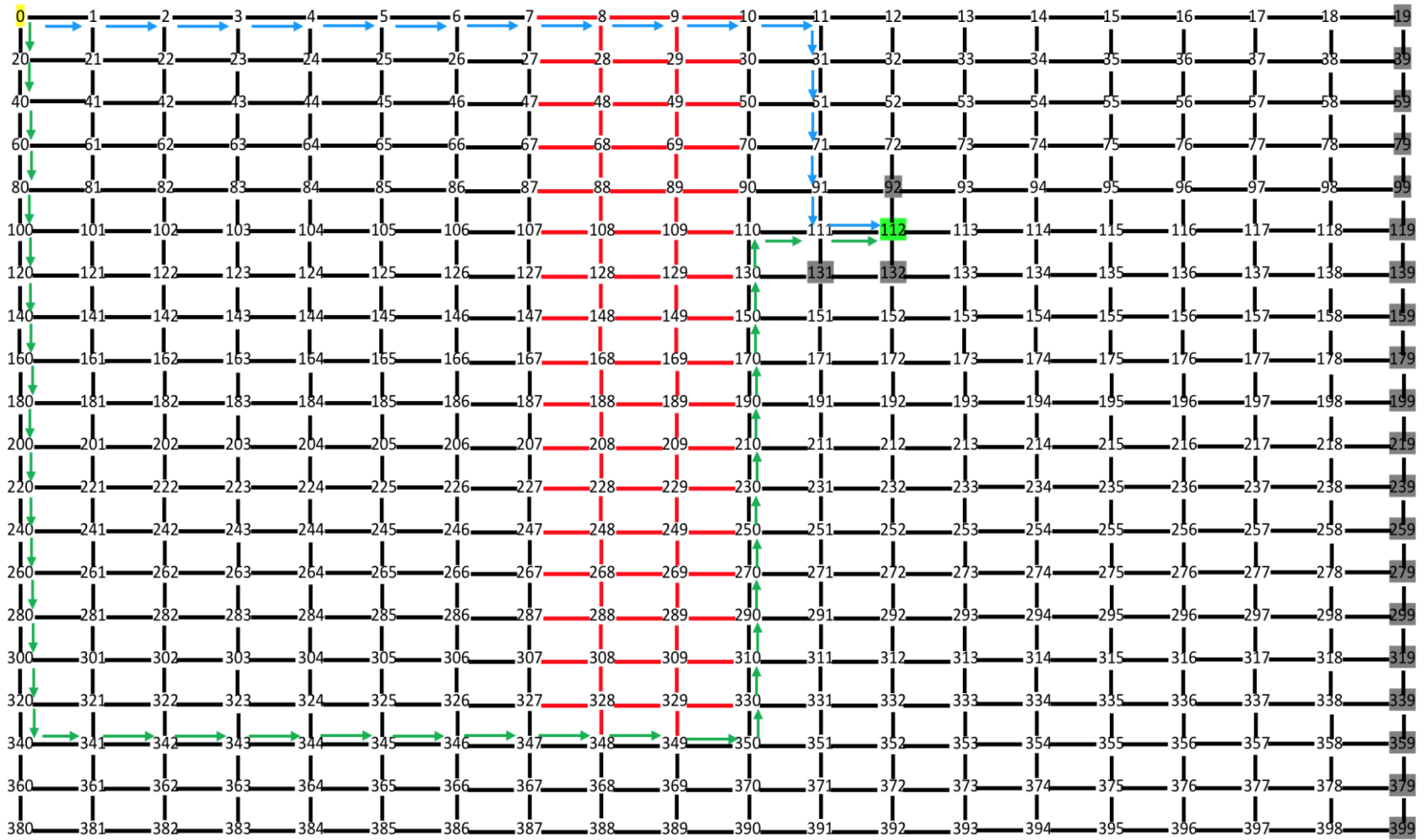


Figure 8: 20x20 map

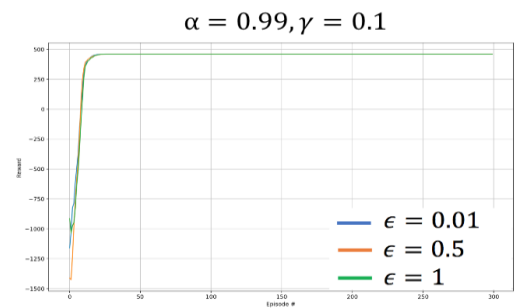
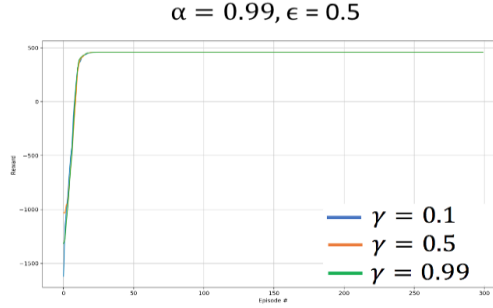
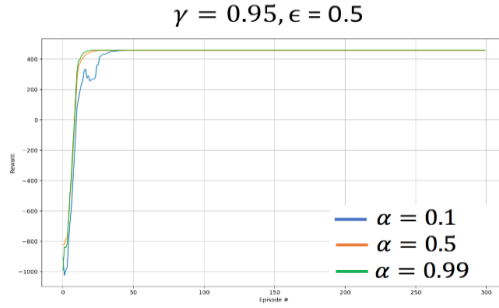
For each simulation, will evaluate:

1. The length of the path and its traffic load.
2. The average cumulative rewards.
3. The average visited states.
4. The average training times.

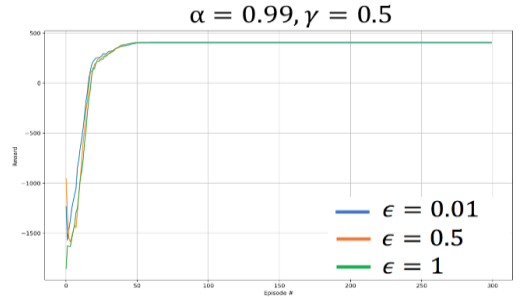
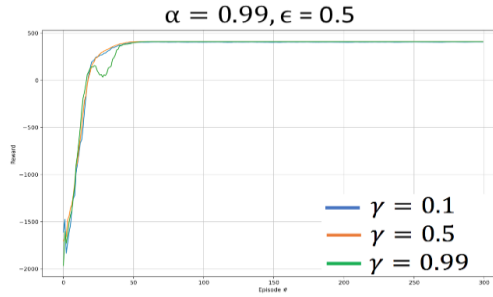
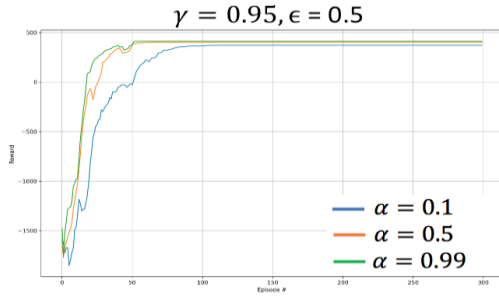
5.2 Tuning Learning Parameters

Setting the correct values for parameters of reinforcement learning algorithms is critical to ensure good performance in its execution and convergence. Thus, adjustment of these parameters is done manually at the initial stage of training. An evaluation of three distinct values of learning rate α , reward discount γ , epsilon ϵ , and temperature T for each proposed algorithm is done. This was accomplished by running the program for 500 episodes and repeat it five times to compute the average. Figures 9-12 show comparison of the cumulative reward per episode using Sarsa and Q-learning in conjunction with: (a) ϵ -greedy, (b) Softmax. It is observable from the graphs that different parameter values obtained different behavior in each exploration method. The best found parameters based on cumulative rewards for all algorithms with the three different experimental setups are shown in Table 4.

6x6 map



10x10 map



20x20 map

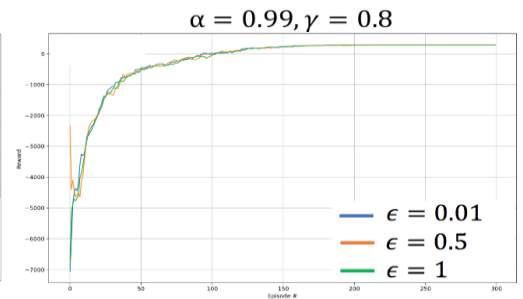
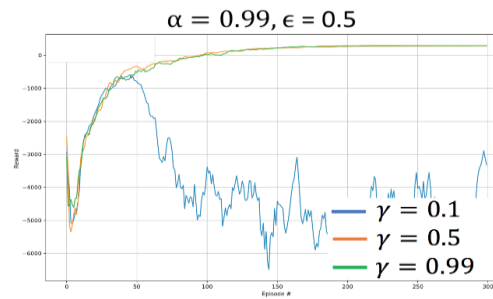
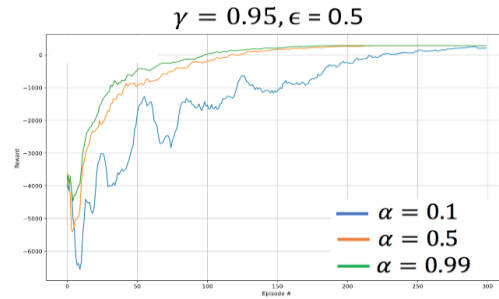


Figure 9: Comparison of parameters for Q-learning- e-greedy in term of cumulative reward with the three different experimental setups

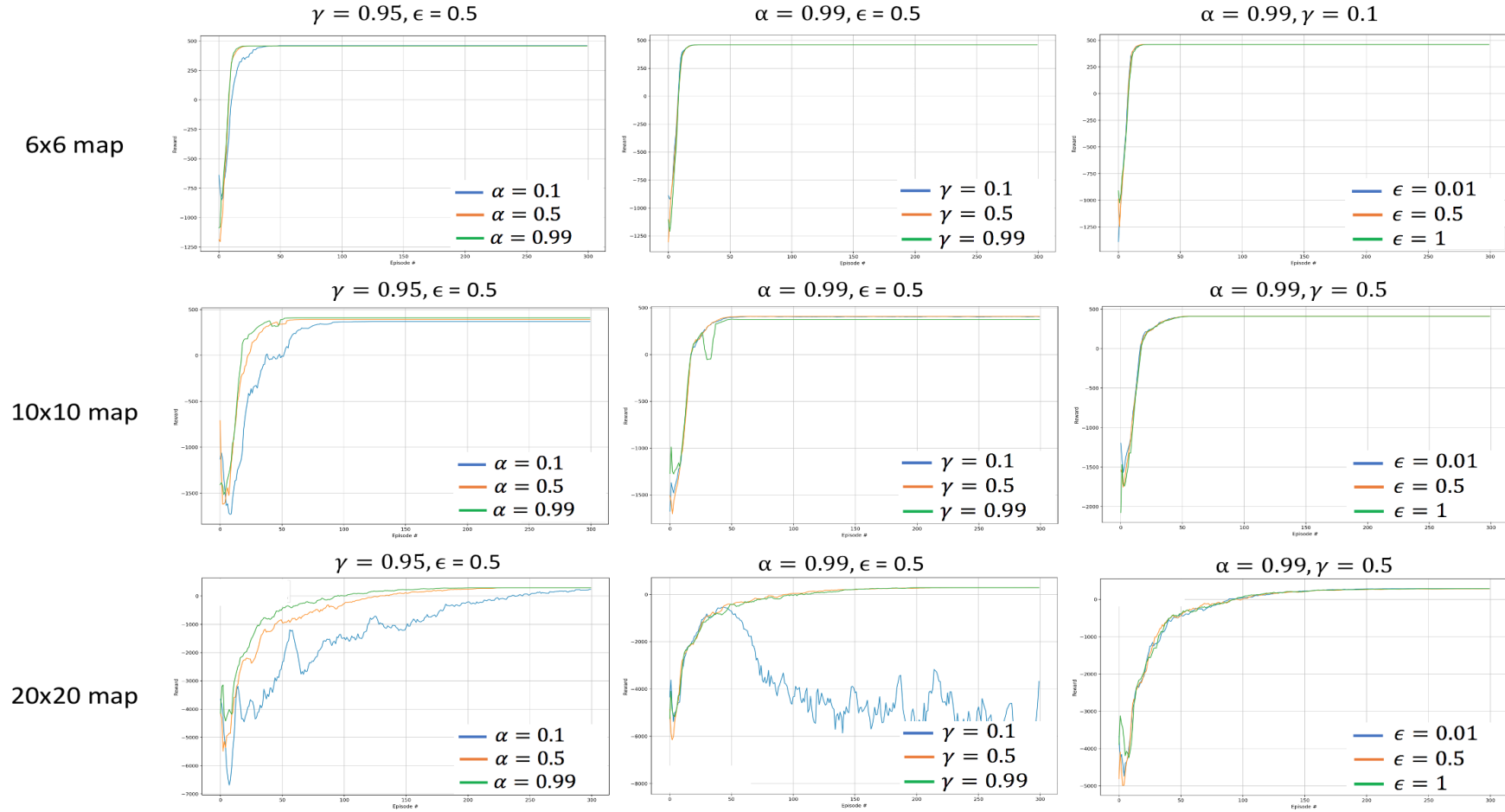


Figure 10: Comparison of parameters for Sarsa- e-greedy in term of cumulative reward with the three different experimental setups

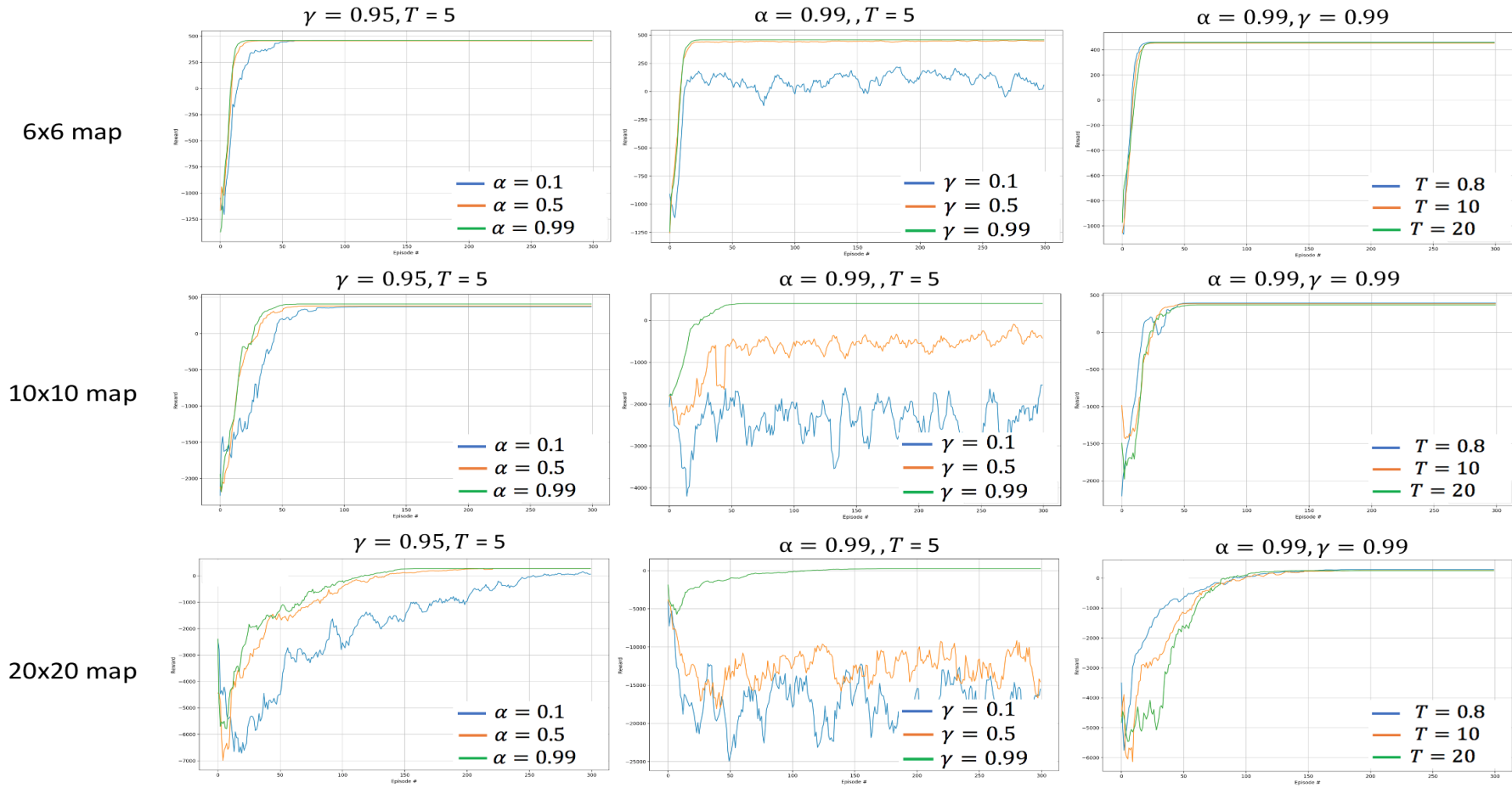


Figure 11: Comparison of parameters for Q-learning-softmax in term of cumulative reward with the three different experimental setups

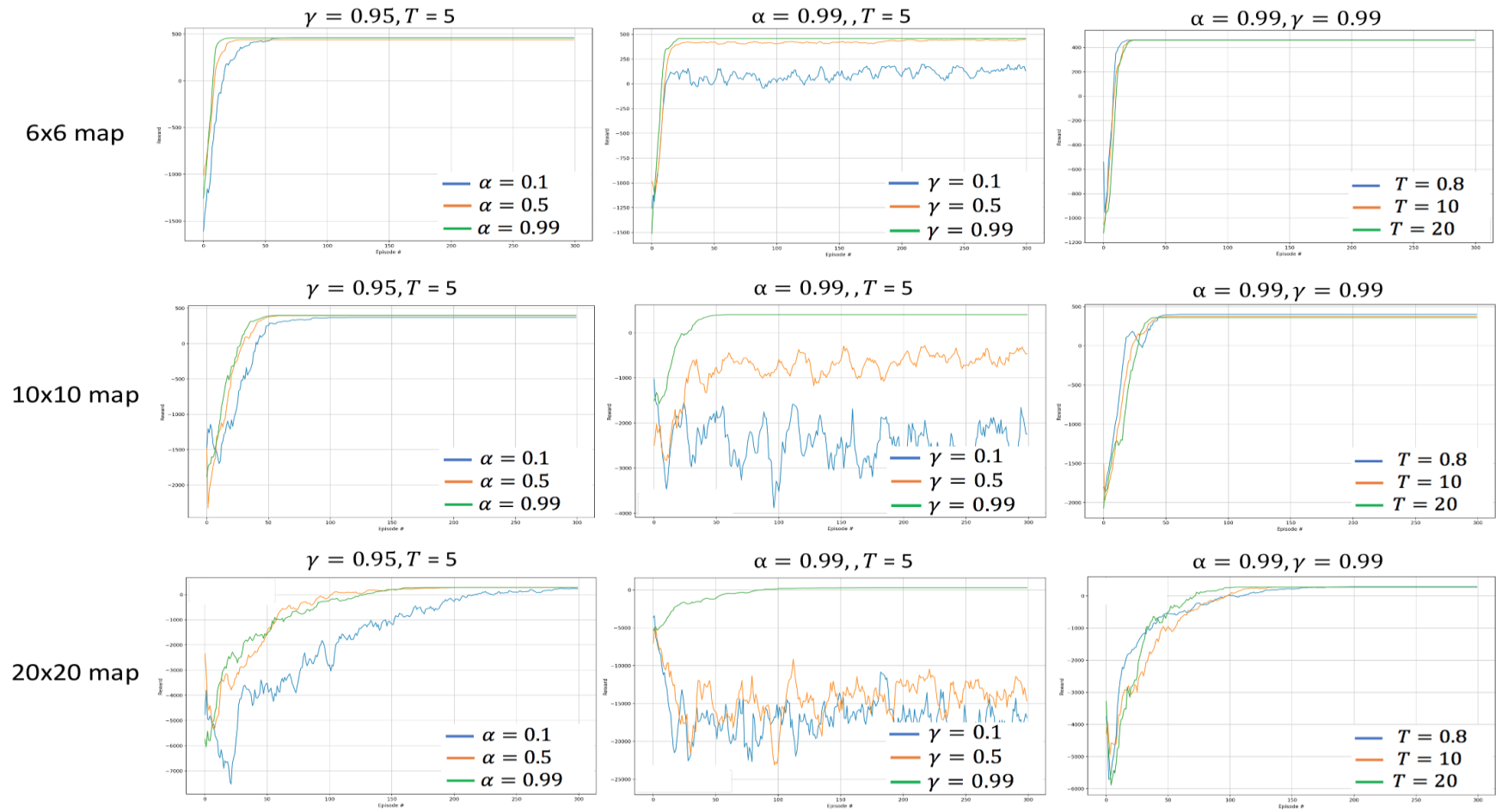


Figure 12: Comparison of parameters for Sarsa- softmax in term of cumulative reward with the three different experimental setups

Table 4: Optimal parameters for each type of experiment

	Q – ϵ – greedy	Q – softmax	Sarsa – ϵ – greedy	Sarsa – softmax
	α, γ, ϵ	α, γ, T	α, γ, ϵ	α, γ, T
6x6 map	0.99, 0.1, 1	0.99, 0.99, 0.8	0.99, 0.1, 0.01	0.99, 0.99, 0.8
10x10 map	0.99, 0.5, 0.01	0.99, 0.99, 0.8	0.99, 0.5, 0.01	0.99, 0.99, 0.8
20x20 map	0.99, 0.8, 0.01	0.99, 0.99, 0.8	0.99, 0.5, 0.7	0.99, 0.99, 0.8

After finding the best parameters for each proposed methods, the measurements are calculated by running the code for 10 repetitions. Each repetition has 500 episodes that make the algorithms converge.

5.3 Comparisons

5.3.1 Comparing the Length of the Path and Traffic Load

As stated earlier, the first objective is to select the near-optimal route for the vehicle to its destination in terms of the minor steps, trip time, and traffic load. Table 5 represents a comparison performance for each proposed algorithm in terms of the most frequently occurring itinerary distance and its load in three different environments.

Table 5: Comparison of the path length and load on three different experimental setups

		Q-e-greedy	Q-softmax	Sarsa-e-greedy	Sarsa-softmax
6x6 map	Path length	9.0	9.0	9.0	9.0
	Traffic Load	40.0	40.0	40.0	40.0
10x10 map	Path length	19.0	19.0	19.0	9.0
	Traffic Load	90.0	90.0	90.0	90.0
20x20 map	Path length	43.0	17.0	43.0	17.0
	Traffic Load	210.0	215.0	210.0	215.0

The table shows that all learning algorithms obtained the same number of steps and load in a 6×6 map. On the other hand, in the 10×10 environment, the Sarsa in conjunction with *softmax* policy significantly outperforms other strategies to determine the near-optimal route with the least congestion. Interestingly, in 20×20 map it have been notice that *softmax* in both Q-learning and Sarsa selects a shorter route with 17 steps but with a higher traffic load compared to *ε-greedy*. Hence, the Formula 4.1 was used by giving different importance weight for the route length and congestion as shown in Table 6. When the weight assigns to = 0.1, it means that the high priority is assigns to traffic load and low advantage to the length of the route. As the value of weight increases, the preference to select the path based on the distance is increased. Based on that, different performances are obtained for each method when the weight changed. The lower the results, the lower the travel time it takes by vehicle to reach its goal.

Table 6: Different performances obtains from different weight factors in 20×20 map

	W				
	0.1	0.3	0.5	0.7	0.9
Q-e-greedy	193.300	159.900	126.500	93.100	59.700
Q-softmax	195.200	155.600	116.000	76.400	36.800
Sarsa-e-greedy	193.300	159.900	126.500	93.100	59.700
Sarsa-softmax	195.200	155.600	116.000	76.400	36.800

5.3.2 Comparing the Average Cumulative Rewards

An efficient way to observe and analyze an agent's success during training is to plot its cumulative reward at the end of each episode. Figures 13-15 show the training process of each considered algorithm in 6×6, 10×10, and 20×20 maps, respectively.

The plots demonstrate almost no performance difference is observable when using $\epsilon - greedy$ and *Softmax* policies in a 6×6 environment. However, the $\epsilon - greedy$ converges faster than the *softmax* policy when using Q-learning and Sarsa in 10×10 and 20×20 maps. The agent finds the destination point with the least number of episodes compared to *softmax*, and its award for committed actions grows. The average reward sums obtained in the three tested maps are presented in Table 7.

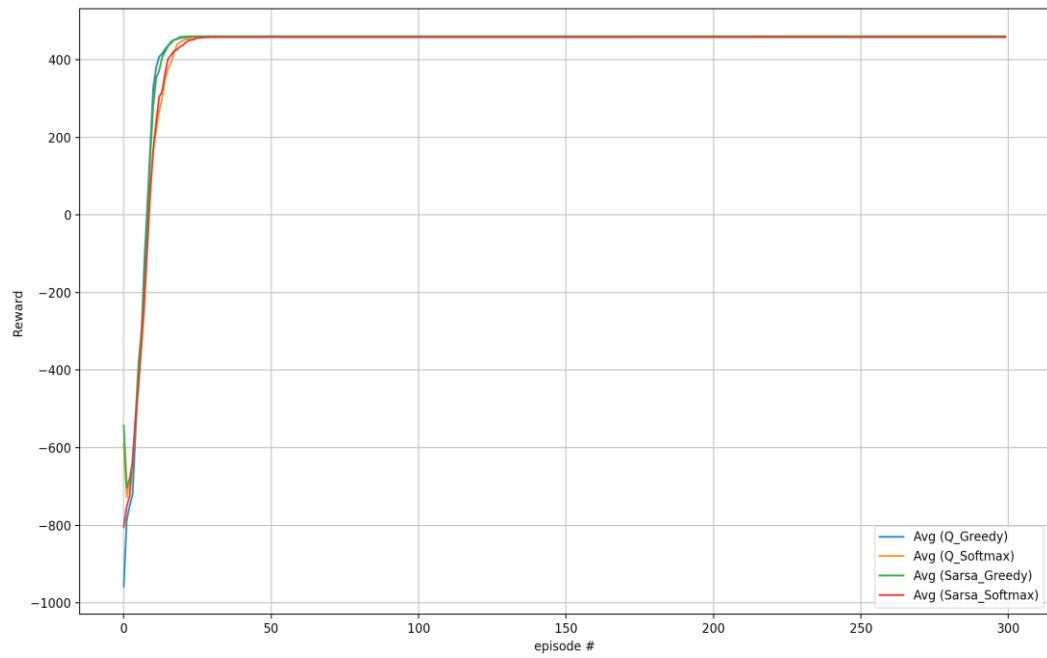


Figure 13: Learning curve of the proposed algorithms tested in 6×6 map

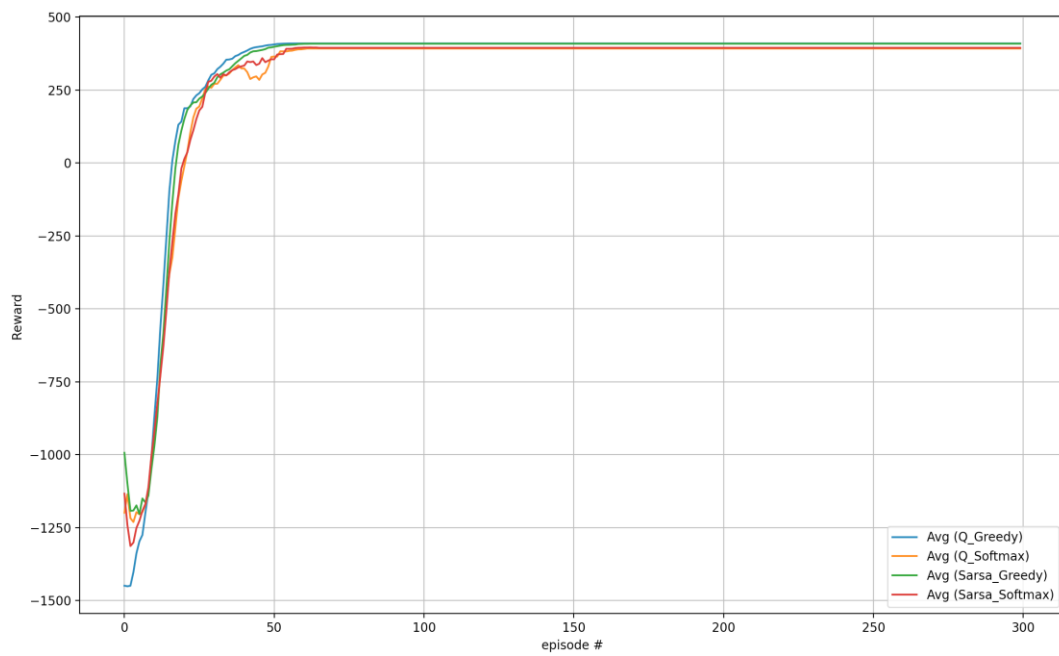


Figure 14: Learning curve of the proposed algorithms tested in 10×10 map

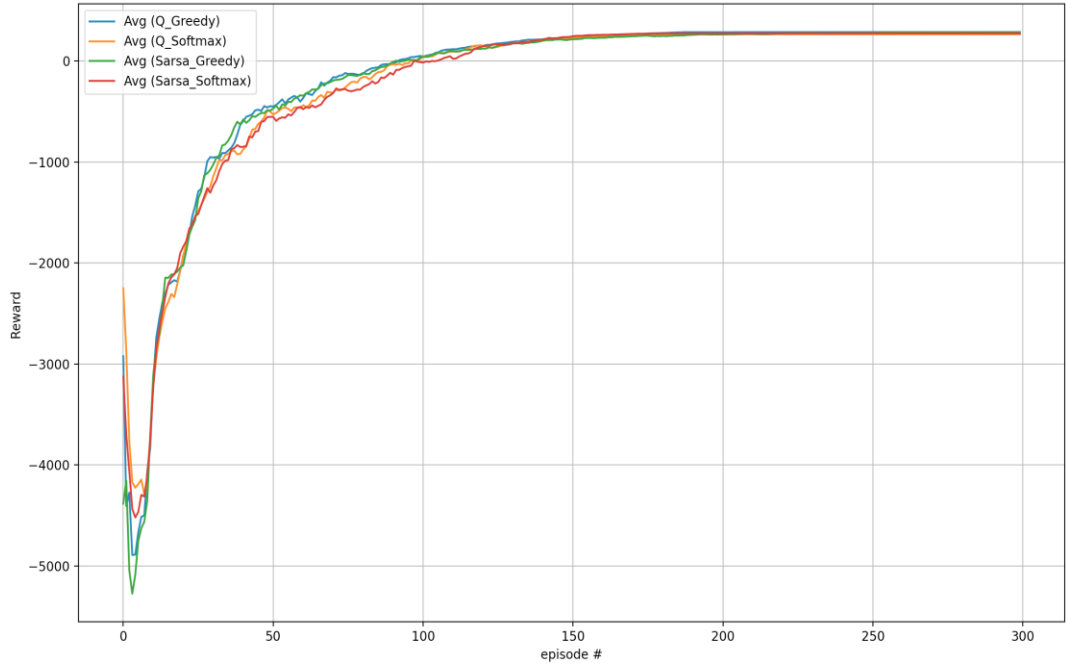


Figure 15: Learning curve of the proposed algorithms tested in 20×20 map

Table 7: Comparison of average cumulative rewards

	Q-e-greedy	Q-softmax	Sarsa-e-greedy	Sarsa-softmax
6x6 map	440.983	437.39	441.03	435.798
10x10 map	340.73	314.055	336.05	317.235
20x20 map	-132.756	-166.243	-150.028	-173.285

The table shows that $Q - \epsilon - greedy$ consistently performs best in most cases, while $Sarsa - softmax$ gives the worst-case reward/episode.

5.3.3 Comparing the Average Training Times

A comparison of the average training time in milliseconds needed for training Sarsa and Q-learning in conjunction with: (a) ϵ -greedy, (b) Softmax is shown in Table 8.

Table 8: Comparison of average training times

	Q-e-greedy	Q-softmax	Sarsa-e-greedy	Sarsa-softmax
6x6 map	116.562	111.659	109.567	96.674
10x10 map	303.466	254.833	272.768	176.150
20x20 map	1196.90	1052.909	1070.23	830.342

The table's data is transformed into a chart to observe with greater insight, as illustrated in Figure 16. The *softmax* policy seems is better optimized compared to ϵ – *greedy*, although there is a less clear difference between *Q – learning – softmax* and *Sarsa – softmax*.

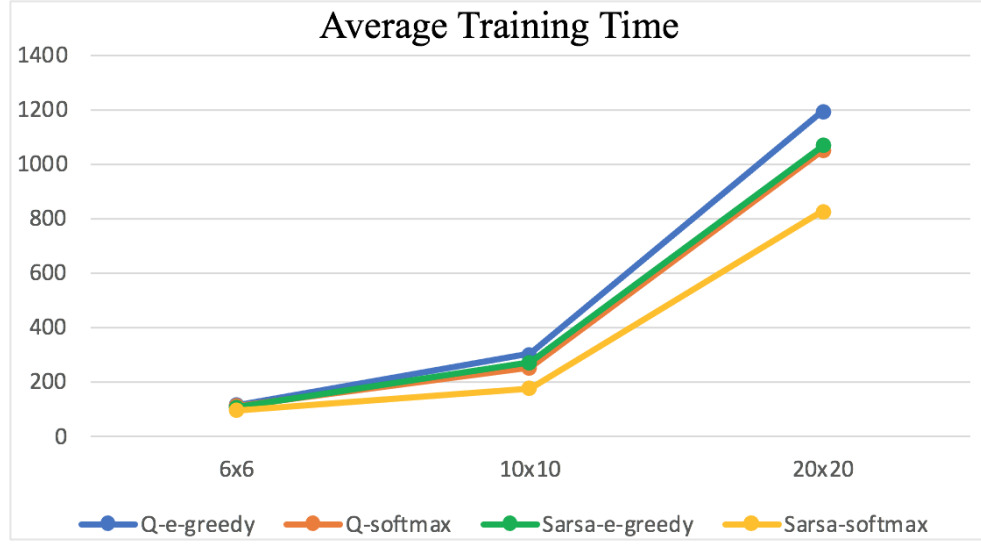


Figure 16: Average training time of the proposed algorithms tested in 6×6, 10×10 and 20×20 map

5.3.4 Comparing the Average Visited State

As the agent visits states and tries different actions, it keeps exploring different paths and learns the optimal Q-values for all possible state-action pairs. Based on that, there is a need to make sure that the agent continues exploring enough to figure out which route is considered the optimal one. Table 9 presents the average number of visited states for each exploration strategy. In 6×6, 10×10, and 20×20 maps, When Q-learning or Sarsa combined with e-greedy, they discover almost the same average of visited states in all experimental environments. Similarly, when they combined with softmax. However, softmax strategy explores almost all states in the three tested environments.

Table 9: Comparison of average visited states

	Q-e-greedy	Q-softmax	Sarsa-e-greedy	Sarsa-softmax
6x6 map	33.8	35.7	33.9	35.4
10x10 map	89.8	96.6	91.5	97.4
20x20 map	395.9	399.4	394.4	399.6

5.4 Discussion

The proposed model has three road network environments to mimic the behavior of traffic in which junctions' states represent the state space, and the process of selecting road segments across the junctions represents the action. Traffic congestion was then generated and distributed among particular road segments based on a specific algorithm. The objective of the vehicle is to select the minor traffic load segments and least path distance to the destination in order to reduce the total traveling time.

This work studied the impact of having different values for learning parameters of each reinforcement learning algorithm on computing efficient vehicle trajectories. These parameters are learning rate α , discounted rate γ , epsilon ϵ , and temperature T . Finally, the efficiency of obtained results was compared, and the optimal parameters of the considered algorithms for the tested environments were found.

An efficient reward function was designed to capture the driving environment and encompass different road and congestion metrics in calculating the near-optimal paths. This reward function has been evaluated in four proposed algorithms; Q-

Learning- ϵ -greedy, Q-Learning-softmax, Sarsa- ϵ -greedy, and Sarsa-softmax. Experiments for comparison of each type of learning algorithm and strategies were also done in order to visualize the difference in the behavior of the agent. From simulation results, it has been found that Sarsa in conjunction with softmax is better optimized compared to other algorithms concerning finding the least congested road and distance in all tested maps. The same holds for taking the least training time and highest number of visited states. However, it performs worst-case in terms of cumulative rewards. On the contrary, Q-learning- ϵ -greedy consistently outperforms all other algorithms in most cases.

Chapter 6: Conclusion

Traffic congestion is a major contemporary issue in many urban areas. Many conventional traffic management methods have been designed to manage the traffic load. An efficient way to solve this problem is to let the vehicle learn how to determine the optimal route based on current traffic conditions. This thesis investigated the use of reinforcement learning methods on calculating the optimized vehicle itineraries in VANET. An efficient reward function was built and evaluated using different techniques and policies. The analysis results show that sarsa-softmax outperforms other strategies to find the optimized path, take the least training time, and explore more states. In contrast, Q-e-greedy performs the best in maximizing the cumulative rewards. During the experiments, the best learning parameters of each approach were discovered, and their effectiveness in maximizing cumulative rewards was compared.

In future work, an experiment will be conducting it on a real simple environment to verify the effectiveness of the proposed system. Also, the use of deep reinforcement learning on computing the optimized trajectories in more complex and more extensive environments will be investigated. In addition, study the effects of other exploration strategies such as UCB-1 and pursuit on the learning performance.

References

- Akhtar, M., Raffeh, M., Zaman, F. ul, Ramzan, A., Aslam, S., & Usman, F. (2020). Development of Congestion Level Based Dynamic Traffic Management System Using IoT. *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 1–6. <https://doi.org/10.1109/ICECCE49384.2020.9179375>
- Ayodele, T. O. (2010). Machine Learning Overview. *New Advances in Machine Learning*, 9-19. <https://doi.org/10.5772/9374>
- Azimian, A. (2011). Design of an Intelligent Traffic Management System (Doctoral dissertation, University of Dayton). Retrieved from <https://etd.ohiolink.edu>
- Babu, V. M., Krishna, U. V., & Shahensha, S. K. (2016). An Autonomous Path Finding Robot Using Q-learning. *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, 1–6. <https://doi.org/10.1109/ISCO.2016.7727034>
- Chang, I.-C., Tai, H.-T., Yeh, F.-H., Hsieh, D.-L., & Chang, S.-H. (2013). A VANET-Based A* Route Planning Algorithm for Travelling Time- and Energy-Efficient GPS Navigation App. *International Journal of Distributed Sensor Networks*, 9(7), 794521. <https://doi.org/10.1155/2013/794521>
- Chen, S., & Wei, Y. (2008). Least-Squares SARSA(Lambda) Algorithms for Reinforcement Learning. *2008 Fourth International Conference on Natural Computation*, 2, 632–636. <https://doi.org/10.1109/ICNC.2008.694>
- Chhatpar, P., Doolani, N., Shahani, S., & Priya, R. L. (2018). Machine Learning Solutions to Vehicular Traffic Congestion. *2018 International Conference on Smart City and Emerging Technology (ICSCET)*, 1–4. <https://doi.org/10.1109/ICSCET.2018.8537260>
- Coskun, M., Baggag, A., & Chawla, S. (2018). Deep Reinforcement Learning for Traffic Light Optimization. *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 564–571. <https://doi.org/10.1109/ICDMW.2018.00088>
- Cleofe, M. (2021). Dubai Drivers Spend 80 Hours Stuck In Traffic Jams In One Year, According To New Inrix Scorecard. *Gulf News*. Retrieved 29 January 2021, from <https://gulfnews.com/uae/dubai-drivers-spend-80-hours-stuck-in-traffic-jams-in-one-year-according-to-new-inrix-scorecard-1.1550387107128#:~:text=UAE's%20Hope%20Probe->

,Dubai%20drivers%20spend%2080%20hours%20stuck%20in%20traffic%20jams%20in,according%20to%20new%20Inrix%20scorecard&text=The%20report%2C%20released%20by%20traffic,and%2079th%20in%20the%20world.

- Dere, E., & Durdu, A. (2018). Usage of the A* Algorithm to Find the Shortest Path in Transportation Systems. *International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES 2018)*, 415–417. Retrieved from https://www.icatces.org/2018/home_files/proceeding_book_2018.pdf
- Dijkstra, E. W. (1959). A Note On Two Problems In Connexion With Graphs. *Numerische Mathematik*, 1(1), 269–271. <https://doi.org/10.1007/BF01386390>
- Festag, A. (2014). Cooperative Intelligent Transport Systems Standards In Europe. *IEEE Communications Magazine*, 52(12), 166–172. <https://doi.org/10.1109/MCOM.2014.6979970>
- Fragkiadaki, K. (2018). Markov Decision Processes [PowerPoint slides]. Retrieved from <https://www.andrew.cmu.edu/course/10-703/>
- Gao, P., Liu, Z., Wu, Z., & Wang, D. (2019). A Global Path Planning Algorithm for Robots Using Reinforcement Learning. *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 1693–1698. <https://doi.org/10.1109/ROBIO49542.2019.8961753>
- Geng, Y., Liu, E., Wang, R., & Liu, Y. (2020). Deep Reinforcement Learning Based Dynamic Route Planning for Minimizing Travel Time. *ArXiv:2011.01771 [Cs, Eess]*. <http://arxiv.org/abs/2011.01771>
- Gottesman, O., Johansson, F., Meier, J., Dent, J., Lee, D., Srinivasan, S., Zhang, L., Ding, Y., Wihl, D., Peng, X., Yao, J., Lage, I., Mosch, C., Lehman, L. H., Komorowski, M., Komorowski, M., Faisal, A., Celi, L. A., Sontag, D., & Doshi-Velez, F. (2018). Evaluating Reinforcement Learning Algorithms in Observational Health Settings. *ArXiv:1805.12298 [Cs, Stat]*. <http://arxiv.org/abs/1805.12298>
- Ho, J., Engels, D. W., & Sarma, S. E. (2006). HiQ: A Hierarchical Q-Learning Algorithm To Solve The Reader Collision Problem. *International Symposium on Applications and the Internet Workshops (SAINTW'06)*, 4–91. <https://doi.org/10.1109/SAINT-W.2006.20>

- Javaid, S., Sufian, A., Pervaiz, S., & Tanveer, M. (2018). Smart Traffic Management System Using Internet Of Things. *2018 20th International Conference on Advanced Communication Technology (ICACT)*, 393–398. <https://doi.org/10.23919/ICACT.2018.8323770>
- Jayapal, C., & Roy, S. S. (2016). Road Traffic Congestion Management Using VANET. *2016 International Conference on Advances in Human Machine Interaction (HMI)*, 1–7. <https://doi.org/10.1109/HMI.2016.7449188>
- Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement Learning In Robotics: A Survey. *The International Journal of Robotics Research*, 32(11), 1238–1274. <https://doi.org/10.1177/0278364913495721>
- Koh, S. S., Zhou, B., Yang, P., Yang, Z., Fang, H., & Feng, J. (2018). Reinforcement Learning for Vehicle Route Optimization in SUMO. *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 1468–1473. <https://doi.org/10.1109/HPCC/SmartCity/DSS.2018.00242>
- Koh, S., Zhou, B., Fang, H., Yang, P., Yang, Z., Yang, Q., Guan, L., & Ji, Z. (2020). Real-Time Deep Reinforcement Learning Based Vehicle Navigation. *Applied Soft Computing*, 96, 106694. <https://doi.org/10.1016/j.asoc.2020.106694>
- Lee, K.-B., A. Ahmed, M., Kang, D.-K., & Kim, Y.-C. (2020). Deep Reinforcement Learning Based Optimal Route and Charging Station Selection. *Energies*, 13(23), 6255. <https://doi.org/10.3390/en13236255>
- Liang, X., Du, X., Wang, G., & Han, Z. (2019). Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks. *IEEE Transactions on Vehicular Technology*, 68(2), 1243–1253. <https://doi.org/10.1109/TVT.2018.2890726>
- Luo, W., Tang, Q., Fu, C., & Eberhard, P. (2018). Deep-Sarsa Based Multi-UAV Path Planning and Obstacle Avoidance in a Dynamic Environment. In Y. Tan, Y. Shi, & Q. Tang (Eds.), *Advances in Swarm Intelligence* (Vol. 10942, pp. 102–111). Springer International Publishing. https://doi.org/10.1007/978-3-319-93818-9_10

- Mejdoubi, A., Zytoune, O., Fouchal, H., & Ouadou, M. (2020). A Learning Approach for Road Traffic Optimization in Urban Environments. In S. Boumerdassi, É. Renault, & P. Mühlethaler (Eds.), *Machine Learning for Networking* (Vol. 12081, pp. 355–366). Springer International Publishing. https://doi.org/10.1007/978-3-030-45778-5_24
- Nafi, N. S., Khan, R. H., Khan, J. Y., & Gregory, M. (2014). A Predictive Road Traffic Management System Based On Vehicular Ad-Hoc Network. *2014 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, 135–140. <https://doi.org/10.1109/ATNAC.2014.7020887>
- Peters, A., Von Klot, S., Heier, M., Trentinaglia, I., Hörmann, A., Wichmann, H. E., & Löwel, H. (2004). Exposure To Traffic And The Onset Of Myocardial Infarction. *New England Journal of Medicine*, 351(17), 1721-1730. <https://doi.org/10.1056/nejmoa040203>
- Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, 2, 331-434. [https://doi.org/10.1016/S0927-0507\(05\)80172-0](https://doi.org/10.1016/S0927-0507(05)80172-0)
- Rahman, M., Ahmed, N. U., & Mouftah, H. T. (2014). City Traffic Management Model Using Wireless Sensor Networks. *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1–6. <https://doi.org/10.1109/CCECE.2014.6901145>
- Rastogi, D. (2017). Deep Reinforcement Learning for Bipedal Robots (Master Thesis, Delft University of Technology). Retrieved from <https://repository.tudelft.nl>
- Sang, K. S., Zhou, B., Yang, P., & Yang, Z. (2017). Study of Group Route Optimization for IoT Enabled Urban Transportation Network. *2017 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCoM) and IEEE Smart Data (SmartData)*, 888–893. <https://doi.org/10.1109/IThings-GreenCom-CPSCoM-SmartData.2017.137>
- Sheikh, M. S., & Liang, J. (2019). A Comprehensive Survey on VANET Security Services in Traffic Management System. *Wireless Communications and Mobile Computing*, 2019, 1–23. <https://doi.org/10.1155/2019/2423915>

- Sichkar, V. N. (2019). Reinforcement Learning Algorithms in Global Path Planning for Mobile Robot. *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, 1–5.
<https://doi.org/10.1109/ICIEAM.2019.8742915>
- Sjoberg, K., Andres, P., Buburuzan, T., & Brakemeier, A. (2017). Cooperative Intelligent Transport Systems in Europe: Current Deployment Status and Outlook. *IEEE Vehicular Technology Magazine*, 12(2), 89–97.
<https://doi.org/10.1109/MVT.2017.2670018>
- Smelser, N. J., & Baltes, P. B. (Eds.). (2001). International encyclopedia of the social & behavioral sciences (Vol. 11). Amsterdam: Elsevier.
- Sutton, R. S., & Barto, A. G. (1998). Introduction to Reinforcement Learning (Vol. 135). Cambridge: MIT press. <https://doi.org/10.1109/TNN.1998.712192>
- Tamilselvi, D., Shalinie, S. M., & Nirmala, G. (2011). Q Learning For Mobile Robot Navigation In Indoor Environment. *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, 324–329.
<https://doi.org/10.1109/ICRTIT.2011.5972477>
- Thrun, S. B. (1992). The role of exploration in learning control. In White, D. A., & Sofge, D. A. (Eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*. Van Nostrand Reinhold, New York, NY.
- Tijmsma, A. D., Drugan, M. M., & Wiering, M. A. (2016). Comparing Exploration Strategies For Q-Learning In Random Stochastic Mazes. *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8.
<https://doi.org/10.1109/SSCI.2016.7849366>
- Touluni, H., Nsiri, B., Boulmalf, M., Bakhouya, M., & Sadiki, T. (2014). An Approach To Avoid Traffic Congestion Using VANET. *2014 International Conference on Next Generation Networks and Services (NGNS)*, 154–159.
<https://doi.org/10.1109/NGNS.2014.6990245>
- Van der Pol, E., & Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*. Retrieved from <https://pure.uva.nl>
- Van Otterlo, M., & Wiering, M. (2012). Reinforcement Learning and Markov Decision Processes. In M. Wiering & M. van Otterlo (Eds.), *Reinforcement Learning* (Vol. 12, pp. 3–42). Springer Berlin Heidelberg.
https://doi.org/10.1007/978-3-642-27645-3_1

- Walraven, E., Spaan, M. T. J., & Bakker, B. (2016). Traffic Flow Optimization: A Reinforcement Learning Approach. *Engineering Applications of Artificial Intelligence*, 52, 203–212. <https://doi.org/10.1016/j.engappai.2016.01.001>
- Wiering, M. A. (1999, February 17). *Explorations in Efficient Reinforcement Learning* [Dissertation]. University of Amsterdam.
<http://localhost/handle/1874/20822>
- Xin, J., Zhao, H., Liu, D., & Li, M. (2017). Application Of Deep Reinforcement Learning In Mobile Robot Path Planning. *2017 Chinese Automation Congress (CAC)*, 7112–7116. <https://doi.org/10.1109/CAC.2017.8244061>
- Yahyaa, S. Q. (2015). Explorations in Reinforcement Learning: Online Action Selection and Value Function Approximation (Doctoral Dissertation, Vrije Universiteit Brussel). Retrieved from <https://www.researchgate.net>
- Zhan, F. B., & Noon, C. E. (1998). Shortest Path Algorithms: An Evaluation Using Real Road Networks. *Transportation Science*, 32(1), 65–73.
<https://doi.org/10.1287/trsc.32.1.65>