ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

Automotive per una mobilità intelligente

Ciclo XXXVI

Settore Concorsuale: 09/F2 Telecomunicazioni

Settore Scientifico Disciplinare: ING-INF/03 Telecomunicazioni

6G enabled IoV Scenarios with Distributed Intelligence Mechanisms

Presentata da: ***Swapnil Sadashiv Shinde***

Coordinatore Dottorato	**Supervisore**
Prof. Nicolò Cavina	**Prof. Daniele Tarchi**

Esame finale anno 2024

6G ENABLED IOV SCENARIOS WITH DISTRIBUTED INTELLIGENCE MECHANISMS

Swapnil Sadashiv Shinde

Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi" - DEI

University of Bologna

MONTH 2024

6G enabled IoV Scenarios with Distributed Intelligence Mechanisms

by

Swapnil Sadashiv Shinde

Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi" - DEI

Submitted

in partial fulfillment of the requirements of the degree of Doctor of Philosophy

to the

University of Bologna

MONTH 2024

This thesis is dedicated . . .

. . .To my entire family *. . .*

for their love and support

*. . .****To my respected Supervisor Daniele*** *. . .*

for his continuous support and motivation

*. . .****To my colleagues and friends*** *. . .*
who were my companions throughout

Certificate

This is to certify that the thesis entitled **"6G enabled IoV Scenarios with Distributed Intelligence Mechanisms"**, submitted by **Mr. Swapnil Sadashiv Shinde** to the University of Bologna, for the award of the degree of **Doctor of Philosophy** in Automotive for Intelligent Mobility, is a record of the original, bona fide research work carried out by him under our supervision and guidance. The thesis has reached the standards fulfilling the requirements of the regulations related to the award of the degree.

The results contained in this thesis have not been submitted in part or in full to any other University or Institute for the award of any degree or diploma to the best of our knowledge.

Prof. Nicolò Cavina
(Coordinator of the PhD program of
Automotive for Intelligent Mobility)
Department of Industrial Engineering,
University of Bologna.

Prof. Daniele Tarchi
(Thesis Supervisor)
Department of Electrical, Electronic,
and Information Engineering
"Guglielmo Marconi",
University of Bologna.

Acknowledgements

This dissertation includes the research activities conducted over the past three years as part of my PhD studies. During this period, I had the opportunity to enhance my knowledge, critical thinking abilities, and decision-making skills, especially in challenging situations. I am deeply grateful for the support and assistance I received from various individuals, whom I would like to acknowledge.

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Prof. Daniele Tarchi, for his unwavering support, guidance, and motivation throughout this period. Without his assistance, I would not have been able to achieve this significant milestone in my life. He not only actively participated in all of these endeavors but also provided me with invaluable insights that have shaped me into an enthusiastic researcher. I firmly believe that the wisdom he has shared with me has not only been instrumental during this period but will also continue to guide me in the years ahead, helping me become a successful individual. I am also deeply grateful to Prof. Alessandro Vanelli Coralli and Prof. Giovanni Emanuele Corazza for granting me the opportunity to work within the Digicomm research group. My sincere thanks also go to Prof. Nicolò Cavina, the coordinator of the PhD program in Automotive for Intelligent Mobility, for his unwavering support. Your assistance has been invaluable, and it has significantly eased my journey throughout these years. Thank you for your contributions to my success. In addition, I would like to thank the reviewers of this thesis, for taking the time to read this dissertation and providing valuable comments.

During my time working in the Digicomm research group, I had the privilege of meeting several individuals who have become wonderful friends over time. My heartfelt thanks go out to my friend Bilal for his continuous support. I also want to express my gratitude to my friends Rabih, David, and all other group members for making this journey smooth and enjoyable. Special thanks to all my friends with whom I had planned this journey a long time ago, particularly Salil, Pradumna, Atul, and Ishtiyaq.

Finally, these achievements wouldn't have been possible without the unwavering support of my entire family. I extend my gratitude to my siblings and parents for their constant support. Thank you all for always being there when I needed it.

Abstract

The Internet of Vehicles (IoV) paradigm has emerged in recent times, where with the support of technologies like the Internet of Things and V2X (Vehicle to Everything), Vehicular Users (VUs) can access different services through internet connectivity. With the support of 6G technology, the IoV paradigm will evolve further and converge into a fully connected and intelligent vehicular system. However, this brings new challenges over dynamic and resource-constrained vehicular systems, and advanced solutions are demanded. This dissertation analyzes the future 6G enabled IoV systems demands, corresponding challenges, and provides various solutions to address them.

The vehicular services and application requests often come with stringent requirements and processing demands which are expected to intensify in future IoV systems. Therefore, proper data processing solutions with the support of distributed computing environments such as Vehicular Edge Computing (VEC) are required. While analyzing the performance of VEC systems it is important to take into account the limited resources, coverage, and vehicular mobility into account. These issues can be a bottleneck of VEC systems applicability for processing the VUs data effectively and advanced solutions can be demanded. Recently, Non-terrestrial Networks (NTN) have gained huge popularity for boosting the coverage and capacity of terrestrial wireless networks. Integrating such NTN facilities into the terrestrial VEC system can address the above-mentioned challenges. Additionally, such integrated Terrestrial and Non-terrestrial networks (T-NTN) can also be considered to provide advanced intelligent solutions with the support of the edge intelligence paradigm.

In this dissertation, we proposed an edge computing-enabled joint T-NTN-based vehicular system architecture to serve VUs. Next, we analyze the terrestrial VEC systems performance for VUs data processing problems and propose solutions to improve the performance in terms of latency and energy costs. Next, we extend the scenario toward the joint T-NTN system and address the problem of distributed data processing through ML-based solutions. We also proposed advanced distributed learning frameworks with the support of a joint T-NTN framework with edge computing facilities. In the end, proper conclusive remarks and several future directions are provided for the proposed solutions.

Contents

Certificate

Acknowledgements

Abstract

Contents

List of Figures

List of Tables

List of Publications

Abbreviations

1 Introduction **1**
- 1.1 Introduction . . . 1
- 1.2 Key Challenges . . . 4
  - 1.2.1 Heterogeneous Nodes with Resource Limitations . . . 4
  - 1.2.2 Advanced Service Demand . . . 5
  - 1.2.3 Advanced ML Solutions . . . 5
- 1.3 Motivation . . . 5
- 1.4 Contributions . . . 8
- 1.5 Organization . . . 9

2 Internet of Vehicles **11**
- 2.1 Introduction . . . 11
- 2.2 Scenario Background . . . 15
- 2.3 Enabling Technologies and Challenges for Futuristic Vehicular Networks 18

2.3.1 6G 18
2.3.2 Network Softwarization 19
2.3.3 Vehicular Edge Computing 22
2.3.4 Machine Learning 25
2.3.5 Non-terrestrial Networks 33
2.4 Multiple Edge Computing Platforms Enabled Joint Terrestrial and Non-terrestrial Network Architecture for Vehicular Scenarios 34

3 Future IoV Network, Key Challanges and Possible Solutions 41
3.1 Introduction 41
3.2 Distributed Data Processing for Vehicular Users 42
3.3 Distributed Intelligence for Vehicular Users 43
3.4 Considered Problems and Outcomes 44
3.4.1 Vehicular Data Processing Problem 45
3.4.2 Distributed Edge Intelligence for Vehicular Users 47

4 Distributed Data Processing for IoV -Terrestrial Case 49
4.1 Introduction 49
4.2 Joint Network Selection and Offloading over VN: Single-Service Case 51
4.2.1 System Model and Problem Formulation 53
4.2.1.1 VU Mobility and Sojourn Time 55
4.2.1.2 VU-EN Assignment, Offloading Process and Resource Allocation 56
4.2.1.3 Problem Formulation 64
4.2.2 MDP Formation 65
4.2.2.1 VU Scenarios Defintion 66
4.2.2.2 MDP Elements 69
4.2.3 MDP-Based Joint Network Selection and Computation Offloading 77
4.2.4 Benchmark Approaches 81
4.2.5 Numerical Results 83
4.2.6 Conclusion 90
4.3 Joint Network Selection and Offloading: Multi-service Case 92
4.3.1 System Model and Problem Formulation 93
4.3.1.1 RSU Selection 95
4.3.1.2 Task Processing 97
4.3.1.3 Vehicle Mobility and Sojourn Time 99
4.3.1.4 Problem Formulation 100
4.3.2 Q-Learning Based Joint Network Selection and Computation Offloading 101
4.3.2.1 Collaborative Q-Learning Solutions for Joint Network Selection and Offloading 106

4.3.2.2 Deep Learning Based Solutions 110
4.3.3 Limited Search-space based Heuristic Approach 112
4.3.4 Numerical Results 114
4.3.5 Conclusion 125

5 Distributed Data Processing for IoV -Joint-Terrestrial and Non-Terrestrial Case 129
5.1 Introduction 130
5.2 Joint Network Selection and Offloading with joint T-NTN Vehicular Scenario - Metaheuristic Solution 131
5.2.1 System Model and Problem Formulation 133
5.2.1.1 VUs Mobility and Distance Measures 135
5.2.1.2 LEO Satellite Mobility and Distance Measures 136
5.2.1.3 Network Selection and Task Offloading Process . . . 137
5.2.1.4 Problem Formulation 141
5.2.2 Proposed Solutions 142
5.2.2.1 Adaptive GA Process (A-GA) 145
5.2.2.2 Benchmark Solutions 147
5.2.3 Numerical Results 148
5.2.4 Conclusions 155
5.3 Network Selection and Offloading with joint T-NTN Vehicular Scenario-HRL Solution 155
5.3.1 System Model and problem Formulation 157
5.3.1.1 VUs Mobility and Distance Measures 159
5.3.1.2 LEO Satellite Mobility and Distance Measures 160
5.3.1.3 Network Selection and Task Offloading Process . . . 161
5.3.1.4 Task Offloading Process 164
5.3.1.5 Problem Formulation 166
5.3.2 Hierarchical Reinforcement Learning Solution 167
5.3.2.1 MDP Models 168
5.3.2.2 Deep Q Network based solution 171
5.3.2.3 Benchmark Solutions 176
5.3.3 Numerical Results 177
5.3.4 Conclusions 184
5.4 Joint Service Placement, Network Selection and Offloading: Multi-time Scale Approach 185
5.4.1 System Model and Problem Formulation 187
5.4.1.1 VU Mobility Model 189
5.4.1.2 Multi-time Scale Approach 190
5.4.1.3 Problem Formulation 195
5.4.2 Multi-time Scale Optimization 197
5.4.2.1 Service Placement MDP 198

5.4.2.2 Network Selection MDP 199
5.4.2.3 Computation Offloading MDP 201
5.4.2.4 Reward Function 202
5.4.3 Deep Q-Learning for Service Placement, Network Selection, and Computation Offloading 203
5.4.4 Numerical Results 211
5.4.5 Conclusion 217

6 Distributed Intelligence for IoV 219
6.1 Introduction 219
6.2 FL-based Computation Offloading in IoV 220
6.2.1 System Model and Problem Formulation 222
6.2.1.1 Vehicular Mobility Model 224
6.2.1.2 Partial Offloading Model 225
6.2.1.3 Task Offloading Process 226
6.2.1.4 Local VU Computation Process 228
6.2.1.5 Partial Offloading Problem 229
6.2.1.6 Federated Learning Model 231
6.2.1.7 FL Computation Model 234
6.2.1.8 FL Communication Model 234
6.2.1.9 Joint Offloading and Federated Learning Model . . . 236
6.2.1.10 Problem Formulation 238
6.2.1.11 Federated Offloading parameter estimation 239
6.2.2 Proposed Solutions 243
6.2.2.1 Clustered Approach 244
6.2.2.2 Distributed Approach 246
6.2.2.3 Genetic Algorithm 247
6.2.2.4 Limited Search-based Heuristic Approach (LS-HuA) 251
6.2.2.5 Optimal Offloading Parameter 252
6.2.3 Numerical Results 254
6.2.4 Conclusion 262
6.3 Distributed FL over Joint Air-ground Networks for Vehicular Applications 264
6.3.1 System Model and Problem Formulation 265
6.3.1.1 VU Mobility Model 267
6.3.1.2 Distributed FL Platform for Vehicular Applications . 269
6.3.1.3 Network Selection Parameters 271
6.3.1.4 FL Process Cost Analysis 274
6.3.1.5 Number of FL Iterations Performed 280
6.3.1.6 Problem Formulation 283
6.3.2 MDP Based SOlution Approach 285

6.3.2.1 Local Environment based Multi-dimensional MDP Model 285
6.3.2.2 MDP-Based FL Network Selection Strategy 292
6.3.3 Benchmark Methods 296
6.3.3.1 Conventional Centralized FL Process (C-FL) 296
6.3.3.2 Minimum Distance Based FL Process (MD-FL) . . . 296
6.3.3.3 Random Assignment Based FL Process (RA-FL) . . 297
6.3.3.4 FedCPF inspired RSU-based benchmark solution for the considered scenario (FedR-FL) 297
6.3.4 Performance Evaluation 298
6.3.4.1 Numerical Results 299
6.3.5 Conclusion 307

7 Conclusion **309**

Scope for Future Work **312**

Bibliography **313**

List of Figures

1.1 Novel Technologies, Main Issues, and Research Challenges. 7

2.1 Vehicular edge computing framework. 24
2.2 Centralized machine learning. 29
2.3 Distributed machine learning. 30
2.4 Federated learning. 31
2.5 Multiple EC enabled VN. 35
2.6 The Multi-EC framework for the vehicular scenario. 37

3.1 Considered Problems and Outcomes. 45

4.1 System Architecture 54
4.2 Proposed MDP Model. 66
4.3 Cost Function 85
4.4 Percentage of VUs with Handover Requirements 87
4.5 Percentage of VUs with service time constraint violation. 88
4.6 Avg. Latency Cost 89
4.7 Avg. VU Energy Cost 90
4.8 Avg. EN Energy Cost 91
4.9 Avg. Number of Active ENs 92
4.10 The multi-service IoV system architecture. 96
4.11 DQN Architecture. 112
4.12 Average Joint Latency and Energy Cost for variable number of VNs. 119
4.13 Average Number of RSUs Handover Requests for variable number of VNs. 121
4.14 Average number of Service Time Failures for variable number of VNs. 122
4.15 Average Task Completion Latency for variable number of VNs. 123
4.16 Average Energy Consumption for Task Completion for variable number of VNs. 124
4.17 Q-Learning Training Performance in Terms of a Service Time Failures for a variable number of iterations. 126

5.1 System Model 133
5.2 Multi Service VN with Non-terrestrial EC Layers 135

5.3 Chromosome Example . . . 144
5.4 Joint Latency and Energy Cost. . . . 150
5.5 Latency Cost. . . . 151
5.6 Energy Cost. . . . 152
5.7 Service Latency Failures. . . . 153
5.8 Sojourn Time Failures. . . . 154
5.9 VU-EN Assignment for A-GA Method. . . . 154
5.10 NTN Edge Computing System Model . . . 158
5.11 Multi Service VN with Non-terrestrial EC Layers . . . 159
5.12 Functional scheme for the proposed HRL solution. . . . 174
5.13 Joint Latency and Energy Cost. . . . 179
5.14 Total Task Processing Latency. . . . 180
5.15 Energy Requirements. . . . 181
5.16 Service Time Failures. . . . 182
5.17 Sojourn Time Failures. . . . 183
5.18 Service Handover Requirements. . . . 184
5.19 The T/NT Integrated Scenario. . . . 189
5.20 Multi-scale MDP Model for the Service Placement, Network Selection, and Offloading Problem. . . . 198
5.21 Proposed Deep Q-Learning Solution . . . 207
5.22 Performance results in terms of overall cost function with variable number of active vehicles. . . . 213
5.23 Percentage of VUs with Handover Requirements. . . . 214
5.24 Percentage of VUs with service time constraint violation. . . . 215
5.25 Avg. No. of EN Selected for Offloading. . . . 217

6.1 System Architecture . . . 223
6.2 VU Mobility Scenarios and Corresponding Distance Matrices . . . 225
6.3 Proposed scheme for the joint FL and task-offloading processes optimization. . . . 232
6.4 Truncated Normal Distribution of α_m as a function of the FL iterations. 241
6.5 FL process impact over the offloading parameter value . . . 241
6.6 FL and Task processing time sharing. . . . 242
6.7 Cost Function . . . 258
6.8 Percentage of VUs with sojourn time constraint violation . . . 259
6.9 Percentage of VUs with service time constraint violation . . . 260
6.10 Percentage of VUs violating the Energy Constraint in (6.33) . . . 261
6.11 Avg. Offloading Error . . . 262
6.12 GA Performance Vs Iterations . . . 263
6.13 The T/NT Integrated Scenario. . . . 268
6.14 Distributed FL Platform. . . . 269
6.15 Distributed FL Process . . . 271

6.16 Distributed FL Process Latency Analysis. 279
6.17 Performance results in terms of overall cost function with variable number of active vehicles. 302
6.18 FL latency with variable number of active vehicles. 303
6.19 Performance results in terms of energy consumption for the FL process with variable number of active vehicles. 305
6.20 Performance results in terms of FL Penalty value with variable number of active vehicles. 306
6.21 Performance results in terms of average number of FL iterations with variable number of active vehicles. 308

List of Tables

2.1 Advantages and disadvantages of different EC platforms for VN services and applications. 26
2.2 Characteristics of the ML Models over VNs. 33
2.3 EC platforms' characteristics. 40

4.1 Simulation parameters 84
4.2 Simulation parameters 115
4.3 Complexity Analysis 117
4.4 Average Percentage of Data Offloading. 125

5.1 Simulation parameters 149
5.2 Simulation parameters 178
5.3 Simulation parameters 212
5.4 Average Percentage of Data Offloading. 216

6.1 Simulation Parameters 255
6.2 Simulation parameters 299

List of Publications

Journals

1. Shinde, Swapnil Sadashiv, and Daniele Tarchi. " A Multi-level Sequential Decision Making Process for Non-Terrestrial Vehicular Edge Computing Environments." Submitted to IEEE Transactions on Machine Learning in Communications and Networking (2023).
2. Shinde, Swapnil Sadashiv, and Daniele Tarchi. " Multi-Time Scale Markov Decision Process for Joint Service Placement, Network Selection, and Computation Offloading in Aerial IoV Scenarios." Submitted to IEEE Transactions on Network Science and Engineering (2023) (Under Major Revision).
3. Lorenzo Ridolfi, David Naseh, Shinde, Swapnil Sadashiv, and Daniele Tarchi. " Implementation and Evaluation of a Federated Learning Framework on Raspberry PI Platforms for IoT 6G applications." Future Internet 15, no. 11 (2023): 358.
4. Girelli Consolaro, Niccolò, Swapnil Sadashiv Shinde, David Naseh, and Daniele Tarchi. "Analysis and Performance Evaluation of Transfer Learning Algorithms for 6G Wireless Networks." Electronics 12, no. 15 (2023): 3327.
5. Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Joint Air-Ground Distributed Federated Learning for Intelligent Transportation Systems." IEEE Transactions on Intelligent Transportation Systems (2023).
6. Shinde, Swapnil Sadashiv, and Daniele Tarchi. "A Markov Decision Process Solution for Energy-Saving Network Selection and Computation Offloading in Vehicular Networks." IEEE Transactions on Vehicular Technology (2023).

7. Adelantado, Ferran, Majsa Ammouriova, Erika Herrera, Angel A. Juan, Swapnil Sadashiv Shinde, and Daniele Tarchi. "Internet of Vehicles and Real-Time Optimization Algorithms: Concepts for Vehicle Networking in Smart Cities." Vehicles 4, no. 4 (2022): 1223-1245.

8. Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Collaborative Reinforcement Learning for Multi-Service Internet of Vehicles." IEEE Internet of Things Journal 10, no. 3 (2022): 2589-2602.

9. Muscinelli, Eugenio, Swapnil Sadashiv Shinde, and Daniele Tarchi. "Overview of distributed machine learning techniques for 6G networks." Algorithms 15, no. 6 (2022): 210.

10. Shinde, Swapnil Sadashiv, Dania Marabissi, and Daniele Tarchi. "A network operator-biased approach for multi-service network function placement in a 5G network slicing architecture." Computer Networks 201 (2021): 108598.

11. Shinde, Swapnil Sadashiv, Arash Bozorgchenani, Daniele Tarchi, and Qiang Ni. "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems." IEEE Transactions on Vehicular Technology 71, no. 2 (2021): 2041-2057.

12. Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Towards a novel air–ground intelligent platform for vehicular networks: Technologies, scenarios, and challenges." Smart Cities 4, no. 4 (2021): 1469-1495.

Presentations and proceedings in International/National Conferences

1. Shinde, Shinde, Swapnil Sadashiv, and Daniele Tarchi. " Time Continuous Federated Learning for Latency Critical Vehicular Applications." Accepted for Presentation in IEEE Wireless Communications and Networking Conference (WCNC) (2024).

2. Shinde, Shinde, Swapnil Sadashiv, and Daniele Tarchi. " Joint Aerial-Ground Offloading for Dependency-Aware IoV Multi-Task Services." Submitted to EuCNC & 6G Summit (2024).

3. David Naseh, Shinde, Shinde, Swapnil Sadashiv, and Daniele Tarchi. " Multi-Layer Distributed Learning for Intelligent Transportation Systems in 6G NTN." Submitted to EuCNC & 6G Summit (2024).

4. Shinde, Swapnil Sadashiv, David Naseh, and Daniele Tarchi. " In-Space Computation Offloading for Multi-layer LEO Constellations."European Wireless (2023).

5. David Naseh, Shinde, Swapnil Sadashiv, and Daniele Tarchi. " Enabling Intelligent Vehicular Networks Through Distributed Learning in the Non-Terrestrial Networks 6G Vision." European Wireless (2023).

6. Abdullah Abbasi, Swapnil Sadashiv Shinde, and Daniele Tarchi." Networked Federated Learning-based Intelligent Vehicular Traffic Management in IoV Scenarios." Globecom (2023)

7. Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Network Selection and Computation Offloading in Non-Terrestrial Network Edge Computing Environments for Vehicular Applications." In 2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC), pp. 1-8. IEEE, 2022.

Abbreviations

3GPP	**3**rd **G**eneration **P**artnership **P**roject
AI	**A**rtificial **I**ntelligence
BS	**B**ase **S**tation
CDNs	**C**ontent **D**elivery **N**etworks
COTS	**C**ommercial **O**ff-**t**he-**S**helf
DNN	**D**eep **N**eural **N**etworks
DRL	**D**eep **R**einforcement **L**earning
EC	**E**dge **C**omputing
FedAvg	**Fed**erated **Av**era**g**ing
FL	**F**ederated **L**earning
HAP	**H**igh **A**ltitude **P**latforms
IoV	**I**nternet **o**f **V**ehicles
IoT	**I**nternet **o**f **T**hings
ITS	**I**ntelligent **T**ransportation **S**ystem
KPIs	**k**ey **P**erformance **I**ndicators
LAP	**L**ow **A**ltitude **P**latforms
LEO	**L**ow **E**arth **O**rbit
LOS	**L**ine **o**f **S**ight
MDP	**M**arkov **D**ecision **P**rocesses
MEC	**M**ultiaccess **E**dge **C**omputing
ML	**M**achine **L**earning
NFs	**N**etwork **F**unction
NFV	**N**etwork **F**unction **V**irtualization
NR	**N**ew **R**adio
NTN	**N**on-**T**errestrial **N**etworks
OBUs	**O**n**b**oard **U**nits

QoS	**Q**uality **o**f **S**ervice
RSUs	**R**oad-**S**ide **U**nits
SDN	**S**oftware-**D**efined **N**etworking
TNs	**T**errestrial **N**etworks
T-NTN	**T**errestrial and **N**on-**T**errestrial **N**etworks
UAVs	**U**nmanned **A**erial **V**ehicles
V2I	**V**ehicle-**to**-**I**nfrastructure
V2P	**V**ehicle-**to**-**P**erson
V2R	**V**ehicle-**to**-**R**oad Side Units
V2S	**V**ehicle-**to**-**S**ensors
V2V	**V**ehicle-**to**-**V**ehicle
VANETs	**V**ehicular **A**d-hoc **Net**work**s**
VEC	**V**ehicular **E**dge **C**omputing
VMs	**V**irtual **M**achines
VNFs	**V**irtual **N**etwork **F**unctions
VUs	**V**ehicular **U**sers
XR	e**X**tended **R**eality

Chapter 1

Introduction

1.1 Introduction

The next generation of mobile networks is expected to be developed under the umbrella term 6G to enable a fully connected, digitized, and intelligent society [1]. The 6G vision is set to serve the evaluated version of current 5G applications and some novel scenarios such as holographic communication, Immersive Extended Reality (XR), etc [2]. Among others, 6G technology is also expected to revolutionize the traditional transportation system into an Intelligent Transportation System (ITS) through the support of the Internet of Things (IoT), Artificial Intelligence (AI), Machine Learning (ML), edge computing, and advanced modes of communication i.e., V2X. With this, the concept of the Internet of Vehicles (IoV) has emerged popularizing new data-intensive and latency-critical applications and services in the highly dynamic vehicular environment [3]. IoV paradigm integrates novel technologies (i.e., vehicle-to-everything (V2X), IoV), to allow Vehicular Users (VUs) to connect through the internet and have access to the different services. However, novel services bring a massive amount of communication and computation demands with stringent requirements. Even though new vehicles have more performant communication and computation-integrated devices, they are not able to satisfy the demand for the new services. Hence, the concept of Vehicular Edge Computing (VEC), a natural extension of Multiaccess Edge Computing (MEC) from wireless networks into vehicular cases has emerged in the last decade [4]. For the case of vehicular scenario, such VEC facilities can be integrated through the deployments of a series of Road-Side Units (RSUs) and corresponding edge servers alongside the road infrastructures. Such RSU-based edge computing facilities effectively solve the major challenges of traditional cloud computing paradigms including long transmission delays, security constraints, and backhaul congestion. VUs can offload some task portion towards these edge servers in the proximity for enabling latency-critical and data-intensive applications and services with demanded Quality of Service. Additionally, with the support of VEC facilities, IoT devices, and novel ML frameworks, proper intelligent solutions can be enabled in the proximity of VUs. However, the limited computation and communication resources of VEC servers along with the restricted coverage ranges of RSU nodes can create new challenges in vehicular scenarios, especially with the growing demands of new services and applications.

With the advance of various new platforms, Non-Terrestrial Networks (NTN) have recently acquired a central place in the 6G research [5]. Both aerial and orbital platforms including Low Altitude Platforms (LAPs), High Altitude Platforms (HAPs), and different satellite constellations can play an important role in creating sustainable and intelligent vehicular systems through their added coverage and capacity boosts [6]. In particular, NTN platforms can enable edge computing facilities in space through the integration of computation and communication resources onboard.

Integrating such edge computing facilities in the traditional VEC systems can effectively boost performance. It can tackle several challenges of VEC systems including resource limitations, vulnerability to natural disasters, coverage limitations, etc. Various NTN platforms located at different altitudes in space with differing mobility patterns can serve ground-based VUs effectively.

With the rapid deployments of 5G systems, initial 5G readings are available. With this, the beyond 5G specifications being a stepping stone into the 6G world are being defined. The 6G vision aims to enable an intelligent society through several new intelligent services and applications. Vehicular Networks (VNs) in particular are expected to go through rapid change and converge into intelligent systems. With these VUs are expected to request a large number of heterogeneous services with stringent requirements. This can put a huge burden on network computation and communication resources. In addition to this, with the presence of a large number of services, providing all possible services at the network edge can be challenging, mainly due to insufficient storage resources. Therefore, optimal service placements over resource-constrained edge nodes become important. Thus, it is important to analyze the performance of distributed computing environments enabled through the integrated Terrestrial and Non-Terrestrial Networks (T-NTN) offering single/-multiple services to vehicular nodes.

The growing demand for intelligent services from VUs is another concern that requires attention. It is important to take into account the resource limitations, mobility, stringent service demands, and heterogeneous nature of networking platforms while providing intelligent solutions for VUs. The edge intelligence paradigm can integrate novel ML techniques with edge computing facilities for providing ML solutions in the proximity of end users. For vehicular cases, it is important to analyze such solutions with special attention to resource and service-related constraints.

1.2 Key Challenges

The next generation of VN, with the support of 6G technology, is expected to revolutionize the road experience with the added intelligent services. In addition to this IoT, NTN, edge computing and ML frameworks will support to enable the fully connected and intelligent VN system. The possible integration of these technologies in the VN can have several benefits in terms of novel services, added intelligence, reduced costs, etc. However, several new challenges can also arise in VN that demand further consideration. It includes novel service demands, dynamic networks, resource limitations, intelligent solution demands, data management, privacy and security concerns, reliability and resilience, scalability, spectrum management, weather and environment challenges, infrastructure maintenance, etc. This dissertation, in particular, focuses on the following key challenges.

1.2.1 Heterogeneous Nodes with Resource Limitations

The next generation of VN is expected to be supported by novel edge computing technology for processing the VU's data. The ground-based VEC facilities can be complemented by additional NTN layers of LAP, HAP, and satellite networks. Each of these platforms can have a different amount of resources, mobility characteristics, distance parameters, node density, coverage, etc. On the other hand, VU's mobility and service demands can add additional constraints. While processing the VUs data at the edge with multiple edge computing facilities it is important to take into account the heterogeneous nature of EC nodes and their characteristics to have proper benefits and cost reductions.

1.2.2 Advanced Service Demand

The next generation of VNs is expected to demand several high-quality services with a heterogeneous nature. Each service can have specific demands in terms of resource requirements. With this, it is impossible to enable the complete sets of services at the EC nodes due to their limited storage resources. Users can have specific demands while accessing the services from edge networks. Thus it is important to take into account the multi-service vehicular scenarios with user and service-centric demands while researching the next-generation VN.

1.2.3 Advanced ML Solutions

With the recent advancement in ML domains and hardware technologies, several new ML solutions are being explored for solving the wireless communication challenges. Enabling effective intelligent solutions in VN with the support of distributed networks is of utmost importance for next-generation VN. The edge intelligence paradigm can merge ML technologies with edge computing frameworks to enable intelligent solutions in the proximity of end users. For vehicular cases, various distributed learning solutions can be analyzed and further optimized with the support of edge networks, advanced communication modes in VN, and distributed IoT data.

1.3 Motivation

The next generation of VN is expected to be a fully connected, digitized, and intelligent system serving users with several new applications and services. Distributed computing facilities enabled through edge networks can have several potential benefits in terms of vehicular data processing and edge intelligence solutions for VN

[7]. The forthcoming intelligent IoV wonderland will generate tons of computation data, and VEC can be a viable solution for providing computation services to the resource-constrained vehicular nodes [8]. However, for having the benefits of VEC resources over multi-user IoV networks, selecting a proper edge server and offloading amount is an important problem. In the past, several authors have tried to solve the computation offloading problem by either finding a proper edge node or the amount to be offloaded. In [9], authors have proposed the adaptive task offloading strategy in the MEC-based vehicular networks environment with a pre-allocation algorithm for vehicle tasks. In [10], the authors focus on an energy-efficient approach for computation offloading in VEC networks. Two heuristic approaches are proposed for solving the problem under different configurations. In [11], authors have studied a reliable computation offloading and task allocation problem over integrated fixed and mobile edge computing enabled vehicular users through the adaptation of software to define networking technology. In [12], the authors proposed a multi-armed bandit approach for optimally selecting the network to be used for computation offloading. In this case, both online and offline approaches are considered. In [13] the authors propose an energy-constrained approach for managing in-vehicle device offloading operations. The problem is solved through a consensus-based approach. It is important to analyze the performance of VEC systems enabled through terrestrial networks that serve VUs with heterogeneous services. In particular, the resource-limited terrestrial VEC facilities can only serve a reduced number of users, proper user-server assignments, offloading, and service placements are required. Additionally with the growing interest in intelligent solutions proper distributed learning solutions enabled through the integration of novel ML approaches and VEC facilities are demanded. Additionally, with the growing interest in NTN systems, integrated T-NTN-based VNs with integrated edge computing facilities should be analyzed for

serving VUs. However, it is important to take into account the heterogeneous nature of EC networks, service demands, user requirements, mobility characteristics, etc while considering such multi-layered network architectures. Growing demands of heterogeneous vehicular service should be taken into account while providing edge computing-related solutions for VUs. VUs demands for intelligent solutions should also considered while investigating the vehicular systems. In particular, enabling efficient ML solutions with the integration of IoT subsystems, novel communication techniques, distributed computing environments and novel distributed learning techniques is one of the major requirements.

With this motivation in mind, I have performed several activities during my Ph.D. studies to analyze the performance of edge computing-enabled VN with a single/multi-service nature. In particular, this dissertation proposes several novel solutions for enabling distributed data processing and edge intelligence solutions in VNs. Figure 1.1 details the main technologies, corresponding issues, and major research challenges considered in this thesis.

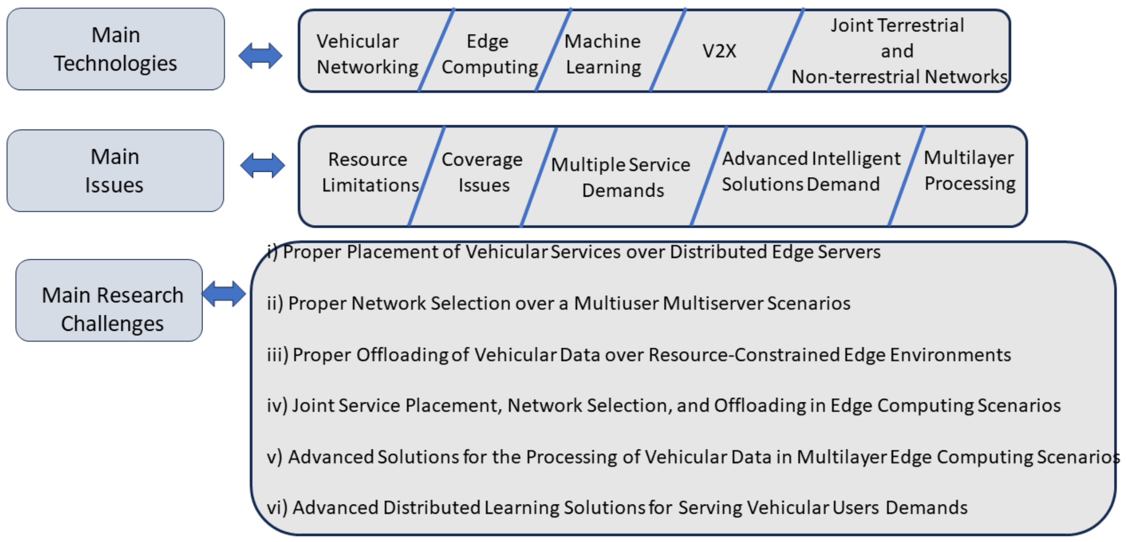

Main Technologies
Vehicular Networking
Edge Computing
Machine Learning
V2X
Joint Terrestrial and Non-terrestrial Networks
Main Issues
Resource Limitations
Coverage Issues
Multiple Service Demands
Advanced Intelligent Solutions Demand
Multilayer Processing
Main Research Challenges
i) Proper Placement of Vehicular Services over Distributed Edge Servers
ii) Proper Network Selection over a Multiuser Multiserver Scenarios
iii) Proper Offloading of Vehicular Data over Resource-Constrained Edge Environments
iv) Joint Service Placement, Network Selection, and Offloading in Edge Computing Scenarios
v) Advanced Solutions for the Processing of Vehicular Data in Multilayer Edge Computing Scenarios
vi) Advanced Distributed Learning Solutions for Serving Vehicular Users Demands

FIGURE 1.1: Novel Technologies, Main Issues, and Research Challenges.

1.4 Contributions

The main contribution of this thesis work includes,

- We proposed novel multiple edge computing platforms enabled T-NTN architecture for vehicular scenarios. We analyze the benefits of integrating terrestrial and non-terrestrial edge computing facilities to serve VUs with distributed computing environments.
- Next, we investigated the vehicular scenario with single and multiple services provided by terrestrial VEC systems. The studies are performed to enable the efficient processing of VUs data with stringent latency demands. Novel ML-based solutions are proposed to minimize the latency and energy costs through a partial offloading process.
- We extend the considered terrestrial VEC scenarios towards the case of joint T-NTN-based edge computing networks. We analyze the distributed computing environment for efficient processing of vehicular data with added capacity and coverage. The problem of proper network selection, offloading, and service placement is taken into account to minimize the overall costs in terms of latency and energy. We have proposed several solutions based on metaheuristic and ML techniques.
- Next, we have also investigated the joint T-NTN-based VNs for enabling distributed learning solutions. In particular, we propose cost-efficient FL solutions for resource-limited vehicular systems. We have developed an FL solution for enabling the efficient data offloading process in VEC with joint optimization of FL and offloading processes. We have also proposed a novel distributed FL framework to enable a cost-efficient FL solution for VUs. In a proposed

framework, the FL process is distributed over the air-ground edge computing facilities to reduce the FL latency and energy costs.

1.5 Organization

The following chapters of the dissertation are organized as:

- **Chapter 2:** In Chapter 2, we have proposed a multiple edge computing facility-enabled joint T-NTN-based VN architecture. We have introduced the importance of the integration of NTN-based edge computing facilities into traditional VEC systems.
- **Chapter 3:** In Chapter 3, we have highlighted several key challenges from a future 6G enabled IoV systems perspective. We also provide a brief overview of the considered problems and the outcome of this dissertation while addressing them.
- **Chapter 4:** In Chapter 4, we provide the results for a terrestrial VEC-based distributed data processing problem. In particular, we analyze the terrestrial VEC system for solving the joint network selection and offloading problem to minimize the latency and energy costs.
- **Chapter 5:** In Chapter 5, we include the results for the activities performed over a joint T-NTN-based vehicular system. We analyze the importance of network selection and offloading problems over a multi-layered edge computing framework and provide metaheuristic and ML-based solutions.
- **Chapter 6:** In Chapter 6, we include the proposed solutions for enabling distributed intelligence solutions in vehicular systems.

- **Chapter 7:** In Chapter 7, we provide conclusive remarks and several future directions for the work done in this dissertation.

Chapter 2

Internet of Vehicles

Some content of this chapter is based on the following article [6];

"Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Towards a novel air–ground intelligent platform for vehicular networks: Technologies, scenarios, and challenges." Smart Cities 4, no. 4 (2021): 1469-1495.".

2.1 Introduction

The traditional transportation system is rapidly converging into an Intelligent Transportation System (ITS) mainly through the integration of innovative technologies, e.g., Internet of Things (IoT), and wireless technologies (e.g., 5G, 6G) [14]. With new connected vehicles, having different communication and computing technologies, Vehicular Networks (VNs) are radically moving toward the Internet of Vehicles (IoVs) paradigm, where new applications and services are growing. With the integration of modernized technologies such as IoT and various new communication modes, including Vehicle-to-Vehicle (V2V), Vehicle-to-Road Side Units (V2R), Vehicle-to-Infrastructure of Cellular Networks (V2I), Vehicle-to-Sensors(V2S), and Vehicle-to-Person (V2P), IoV can expand the traditional Vehicular Ad hoc Networks (VANETs) capability by adding several new services and applications. IoV is able to support various functions, including intelligent traffic management, traffic safety enhancements, dynamic information services, intelligent vehicle control for improving the overall road experience of users [15, 16, 17, 18]. However, vehicle onboard computation capacity is falling short when new latency-critical and data-intensive applications are considered; hence, new challenges are arising in VNs. The Cloud computing facilities are able to reduce the computation burden of new services; therefore, Vehicular Users (VUs) can transmit the portion/complete task to the cloud servers having enormous computation and communication power. In general, the cloud facilities are located far from the users, in the core network, and introduce some drawbacks, e.g., huge transmission costs, backhaul link congestions, data security threats due to long-distance communications. Such issues can be addressed by integrating the Edge Computing (EC) facilities into VNs, bringing cloud computing resources in the proximity of end-users [19, 20].

In the case of VNs, EC facilities can be integrated through the deployment of several EC servers alongside the road network co-located with Roadside Units (RSUs). Such an approach is known as Vehicular Edge Computing (VEC) and has achieved lots of success by enabling new latency-critical services into VNs [21]. However, the limited capacity and coverage of EC servers/RSUs is the main bottleneck while exploiting VEC advantages. VEC-based VN can also be complemented by adding additional EC servers located on the terrestrial cellular base stations (i.e., 5G-gNB). Such ground-based multi EC platforms can compliment VUs by providing additional services and reducing the overall burden of the VEC servers.

However, in recent times, Terrestrial Networks (TNs) are becoming more and more used, with many new users requesting services with specific demands. Limited coverage into rural and remote areas, unreliable service during natural disasters like tsunamis and earthquakes, new security challenges, poor link budget with additional interferences are some of the main challenges that need to be considered while utilizing TN-based EC platforms into VNs. In addition, VUs are not the only user group requesting services from the TN-based EC platforms. With the presence of different users, the dynamically changing resources of EC servers with vehicular mobility adds additional challenges while integrating the TN-based EC services into VNs. With this limitation, integrating TN-based EC platforms into VN alone can not be a sufficient solution for the new futuristic vehicular services and applications, which will have more stringent requirements in terms of latency and computation resources.

Encouraged by the new technological developments and the additional interest shown by several tech giants (i.e., Facebook, Google, etc.), the Non-Terrestrial Networks (NTN), including space and air networks are growing these days mainly for providing global connectivity [22]. Several new platforms such as new satellite constellations, Unmanned Aerial Vehicles (UAVs) swarms, small fueled aircraft, balloons have been deployed at different heights from the ground users to achieve the global connectivity challenge [23]. Better connectivity, scalability, reliability are some of the advantages of NTN based communication platforms. With the addition of modern communication technologies, such as multi-beam antennas, the NTN platforms can also provide EC-based services with an onboard computing server [24, 25, 26]. Such NTN-based EC platforms can complement the VNs for solving several problems, including RSUs' limited capacity and coverage. Compared with space networks, Low Earth Orbit (LEO) Satellites and new aerial platforms such as Low Altitude and High Altitude Platforms (LAPs and HAPs) have a considerable advantage with reduced transmission distances, low deployment time and costs, and reduced communication channel losses. Therefore LEO Satellites, LAPs and HAPs can be excellent solutions for complimenting the VN for providing new innovative and intelligent services to the end-users.

Every connected vehicle in a VN is equipped with several sensors, able to generate tones of data (i.e., big data) in real-time. These data can be analyzed and exploited for improving the quality of VNs services and applications [27]. Recently machine learning (ML) techniques are used for solving challenging problems over wireless networks [28, 29]. With new hardware technologies and the availability of a large amount of data through IoT devices, ML research is grown fast. Several new ML tools and techniques have been developed and utilized for solving real-world problems on a daily basis. The use of new innovative ML-based solutions for analyzing the vehicular data for improving the vehicular environment can be beneficial [30]. However, proper infrastructures with communication and computing resources are required, for embedding the ML-based solutions into VNs, failure of which can introduce higher process costs (i.e., cost induced by ML model training, inference, etc). With limited resources and dynamic movements in VNs, it is challenging to implement ML techniques. The EC resources can be exploited for integrating the ML-based solution techniques into VNs.

Traditional ML approaches such as centralized ML require VUs to transmit their data to the centralized, more powerful servers [31]. However, such an approach can introduce higher costs in terms of communication latencies and energy requirements. With resource limitations and critical service requirements performing centralized ML over VN can be challenging. On the other hand, new learning approaches such as Federated Learning (FL), VUs are able to perform the training operations by themselves [32]. In each FL round, VUs perform the training operations to learn local ML model parameters, to be then sent towards the centralized server. After receiving all VUs parameters from its coverage area, the server is able to perform an averaging operation for creating a global model to be broadcast back to the VUs. Thus, the VUs can reduce the data communication cost by performing local training operations and also exploit the presence of other VUs, by learning experiences/data through the averaging. For these reasons, recently, FL-based solutions are preferred for VN application when solving challenging problems [32, 33, 34]. For benefiting from the collaborative learning into FL, a server platform with better coverage characteristics and channel conditions is required. NTN platforms, such as HAPs, can assist VN in implementing an efficient FL process with their better coverage characteristics, moderate transmission distances, and better channel conditions.

Therefore, there is clear scope for integrating NTN layers, with ground-based EC resources, into VNs. Such an approach can solve the limited capacity and coverage problem of conventional ground-based EC platforms and can provide more intelligent services and applications with stringent requirements. Therefore, here we provide a novel multiple EC platforms-enabled VN architecture by integrating ground and NTN-based EC resources. Different enabling technologies, corresponding challenges, and opportunities are discussed in detail. We further analyze several vehicular scenarios and the benefits of using the proposed VN architecture for solving the challenges associated with them.

2.2 Scenario Background

Developing a proper ITS is a key to creating a connected society. Different technologies are merging with the goal of forming sustainable, safe, and intelligent VNs for serving VUs. In [35], the authors surveyed several ML-based solutions employed in VNs communication and networking parts. Differently, the importance of network softwerization technology in the VN and corresponding challenges is surveyed in [36]. In [37], the authors have proposed a software-defined collaborative EC platform for the vehicular scenario. They have mainly focused on ground-based EC technologies, including Mobile Edge Computing (MEC), fog computing, cloudlet, etc. However, limited capacity and coverage issues of ground-based EC platforms limit the performance of traditional VNs.

Recently, the importance of NTN platforms in wireless communication has been highlighted in different projects and research works. Several ongoing projects including EdgeSAT [38], SATis5 [39], Expanse [40] are aimed at integration (with terrestrial networks (TNs)), softwarization, and expansion of EC facilities over NTN platforms. Several study items such as IoT over NTN, New Radio (NR) over NTN, satellite components in 5G architecture, and unmanned aerial systems are part of 3rd Generation Partnership Project (3GPPs) Release 17 studies, which was finalized at the end of the first quarter of 2022 [41]. It is also expected that 3GPP will reconsider many of these items in an upcoming release (i.e., Release 18) [42]. Special attention has been provided in 3GPPs release 17 toward different NTN technologies to enable a strong foundation for direct communication between mobile users and satellites. 3GPP has worked on adapting the 5G-NR, narrowband IoT, and LTE for machine-type communications to provide satellite connectivity. These studies are spaned over two 3GPP tracks called 3GPPs NR NTN and IoT NTN studies. In Release 17, the main focus of 3GPP NTN studies was to provide communication-related services to consumers through satellite networks. Two different architectures are considered in 3GPP NTN studies to realize the satellite communication systems. In general, feeder links are considered to connect the satellite radio payload to the core network through NTN gateways. Communication services can be provided by satellite networks to user equipment through service links. Though 3GPP release 17 has specified the transparent NTN architecture, the studies can be enhanced to support the regenerative architecture. The spot beam approach where a service area is divided and covered by individual beams is considered in modern satellite communication. However, this can induce frequent handover between cells adding more challenges, especially for LEO satellites with high mobility. To counter this a beam steering mechanism can be adapted satellite beams are steared on a fixed surface of earth for longer durations. 3GPP release 17 supports both of these approaches. 3GPP solutions also include solutions for critical challenges such as large round trip time and Doppler shifts. User equipment equipped with a Global Navigation Satellite System (GNSS) module is expected to compensate for the delay and Doppler shifts by accessing the position and velocity information of satellites and using it to determine their position before accessing the network. Other studies include the support for downlink transmission polarization signaling, extension or offset start of various timers, enhancements for cell selection, etc. Another 3GPP trach called IoT NTN is considered for adaptation of narrowband IoT and LTE machine type communication for enabling them to support NTN. In general, the idea is to study the 3GPPs NR- NTN findings and adapt them to IoT NTN. Both NR NTN and IoT NTN work is going to expand in Release 18. New NR-NTN studies include coverage enhancements, advanced mobility procedures, and methods for the network to independently verify the reported user equipment location. Also, for IoT NTN, studies include methods to disable HARQ feedback, mobility enhancements, and improvements for discontinuous covergae-related studies.

In [5], the authors presented the new opportunities and challenges ahead for integrating the NTN technologies into upcoming 6G networks. The work in [43] analyzes the importance and challenges of integrating NTN networks into VN. A space-air-ground integrated VN architecture for supporting different vehicular services in diverse scenarios is proposed. Furthermore, in [44], the authors have presented a space-air ground integrated VN architecture highlighting the key features of each platform layer. It is possible to enable several EC services over different NTN layers by deploying proper computing resources. In [25], authors proposed EC-enabled UAV platforms for improving computation performance and reducing the execution latency of MEC systems. Recently in [45], authors studied the energy performance of an offloading strategy designed over EC-enabled satellites and HAP networks for ground-based users. In [46], the importance of the UAV-assisted VEC system and corresponding implementation issues are highlighted.

2.3 Enabling Technologies and Challenges for Futuristic Vehicular Networks

A suitable VN architecture able to serve users with new innovative services and applications has to exploit together several key technologies. However, with limited EC capacities and coverage restrictions, it is challenging to integrate these high-level technologies into the vehicular environment. Here, we introduce the main technologies for designing futuristic VNs and corresponding challenges.

2.3.1 6G

The 6G networks are expected to evolve the traditional 5G application scenarios towards new dimensions. Additionally, new scenarios that were beyond the capabilities of 5G systems are also integrated into the 6G vision. With extended KPIs, support towards novel application scenarios, inherent intelligence, global coverage, and sustainable networking, 6G technology is expected to be a main driving force for enabling intelligent VNs. The extreme demands of IoV use cases in terms of ultra-low end-to-end latencies, ultra-high reliability, connectivity, data rates, and advanced positioning are expected to be satisfied through new 6G capabilities. The 6G technology is expected to support novel application scenarios such as tactile internet, digital twin, wireless brane machine interface, holographic communication, emergence rescue communication, and Immersive XR. Several of these scenarios can be integrated into the IoV systems to satisfy the end users' demands. The 6G is expected to have native intelligence support with the help of edge networks. This can be utilized to satisfy the demands of intelligent solutions in the IoV case. With the integrated terrestrial and non-terrestrial networks in 6G technology, the coverage and capacity of traditional ground-based networks can be improved. With several of these novel features, 6G technology can be extremely important to enable fully connected, digitized, and intelligent vehicular systems.

2.3.2 Network Softwarization

Network softwarization is a key trend that uses Software-Defined Networking (SDN), Network Function Virtualization (NFV), and network slicing techniques for providing additional programmability, flexibility, and modularity in different parts of the wireless communication networks [47]. Network softwarization allows a flexible deployment and control of different vehicular services once adapted into VNs. Here, we introduce the main technologies that allows to create a flexible and programmable VN.

Network Function Virtualization (NFV)

NFV is a key technological trend to tackle the flexibility and scalability problems associated with traditional hardware-based Network Functions (NFs). In the past, different NFs such as firewalls, Content Delivery Networks (CDNs), Network Address Translation (NAT) were installed as dedicated hardware-based appliances. With this, the implementation of new services and applications were restricted by the deployment of specific hardware-based elements. NFV decouples this network functions from proprietary hardware appliances and runs them as software instances in Virtual Machines (VMs)/containers as Virtual Network Functions (VNFs) [48]. Through NFV, standard network resources such as compute, storage, and network functions can be virtualized and kept on Commercial Off-the-Shelf (COTS) hardware like x86 servers. In addition, multiple VMs/containers, through the proper assignments of virtualized resources, can run on a single server for improving server resource utilization. With the NFV technology, different network functions can be placed in different locations of the networks elements such as data centers, EC servers, etc.

In the case of a Multiple EC (Multi-EC) platforms-enabled-VN architecture, NFV can potentially bring several benefits, including reduced network cost, less time-to-market for new services, higher resource efficiency, and better scalability. However, each VNF demands specific computation resources based upon the service types. Furthermore, since VNFs can be deployed at multiple locations, it is important to find proper function placement strategies for implementing many hybrid vehicular services over a resource-constrained VN.

Software-Defined Networking

With the unprecedented increase in demand for heterogeneous services in VN, there is a need for a platform that can dynamically adapt the network and service according to the demand request. SDN technology can create a more flexible and programmable VN for supporting the critical requirements of the services and applications [49, 50]. The SDN forms a fully programmable wireless network by logically separating the data and control plane, where all control operations are performed through the centralized controller unit. The controller can have a global view of network topology, traffic load, network states, and link failures. In the case of an EC-enabled VN, having different EC layers of heterogeneous hardware having a proper centralized controller assisted by the individual layers local controller can be useful for providing flexible VN services. In the past several studies have shown the importance of SDN-based VNs, where SDN technology is integrated into VN to manage different parts with improved performance in terms of network flexibility, throughput, and flexible deployments of new services and applications. However, various issues need to be handled carefully during the integration of SDN into VN including, possible security attacks over the data plane, the vulnerability of a centralized controller in terms of single-point failures, issues related to the heterogeneous hardware of different EC platforms, various access network technologies, etc. Over the years, researchers have provided numerous solutions for these issues [37, 50, 51, 52, 53]. As an example, in [51], a 5G software-defined vehicular network having integrated SDN technology is proposed with clear separation of data, control, and application planes. In another study [37], the authors proposed a collaborative EC-based software-defined VN with multiple EC platforms. Furthermore, in [50], the authors have experimented with the different SDN controllers over a complete software-defined vehicular networking on hardware. Recently in [52], the authors have proposed a novel IoV automation and orchestration system using SDN for connected autonomous vehicles. In [53], the authors study different VN architectures and propose a localized intelligence augmented highly reconfigurable software-defined heterogeneous vehicular networking architecture for avoiding single-point failures.

Network Slicing

The network slicing technique takes advantage of NFV and SDN for creating multiple logical networks or network slices over a common physical infrastructure of VN [54, 55]. Multiple network slices can be configured over the same physical infrastructure to provide different services with diverse requirements. It provides dynamic resource management by enabling efficient resource sharing by considering various key Performance Indicators (KPIs) for each slice and can be an efficient technology over the VN with limited resources. Inherently, the network slice contains a chain of physical and virtual network functions that can be placed at different locations based upon the service requirements. In the case of an EC-enabled VN with users requesting services with different KPIs, each slice function needs to place carefully for satisfying the user demands.

2.3.3 Vehicular Edge Computing

With the development of IoT and new wireless communication technologies, many new services and applications have been enabled into the VN. Users are demanding services with tighter requirements in terms of latency, bandwidth, computational capability; with limited onboard resources VUs alone are not capable of providing a satisfying Quality of Service (QoS). One way to solve this issue is to use cloud computing, where each VU can transmit their workloads towards the computationally rich cloud servers located deep inside networks. Thus, cloud servers perform the computation operations on behalf of the VUs and send back the results. With high computation resources, the cloud server can serve many VUs with negligible computation latency. However, with a long transmission distance between VU and cloud platforms, this approach introduces large communication delays. Furthermore, user data can be exposed over a large transitional distance can be prone to the security challenges such as third-party attacks. With limited backhaul resources, issues like network connections are likely to happen when many VUs from a giver service area request cloud computing services. Thus, providing a satisfying QoS can be challenging over the cloud computing platform.

For solving cloud computing problems, the MEC approach was introduced [56]. MEC brings cloud computing services closer to the end-users by deploying several edge servers in proximity. Thus, users can transmit their workload to these servers without incurring large delays, security issues, or network congestion problems. MEC has gained lots of attention in the recent past and enabled several new latency-critical applications and services over wireless networks [57, 58]. In the vehicular scenario, the MEC framework is configured as the deployment of several RSUs along with the road networks and equipping them with the edge servers deployed in the proximity [21]. This approach is called vehicular edge computing (VEC) and has a huge potential for enabling latency-critical and data-intensive VN services [59]. Figure 2.1, shows the main elements of a reference VEC system which includes distributed VUs, several RSUs along with the road network, and EC servers located nearby RSU nodes. VUs can access VEC services provided by EC servers through RSU nodes with V2I communication links, while they can communicate among them through V2V communication links [60]. RSUs can also act as a gateway for uploading vehicular data to the BS and Cloud facilities.

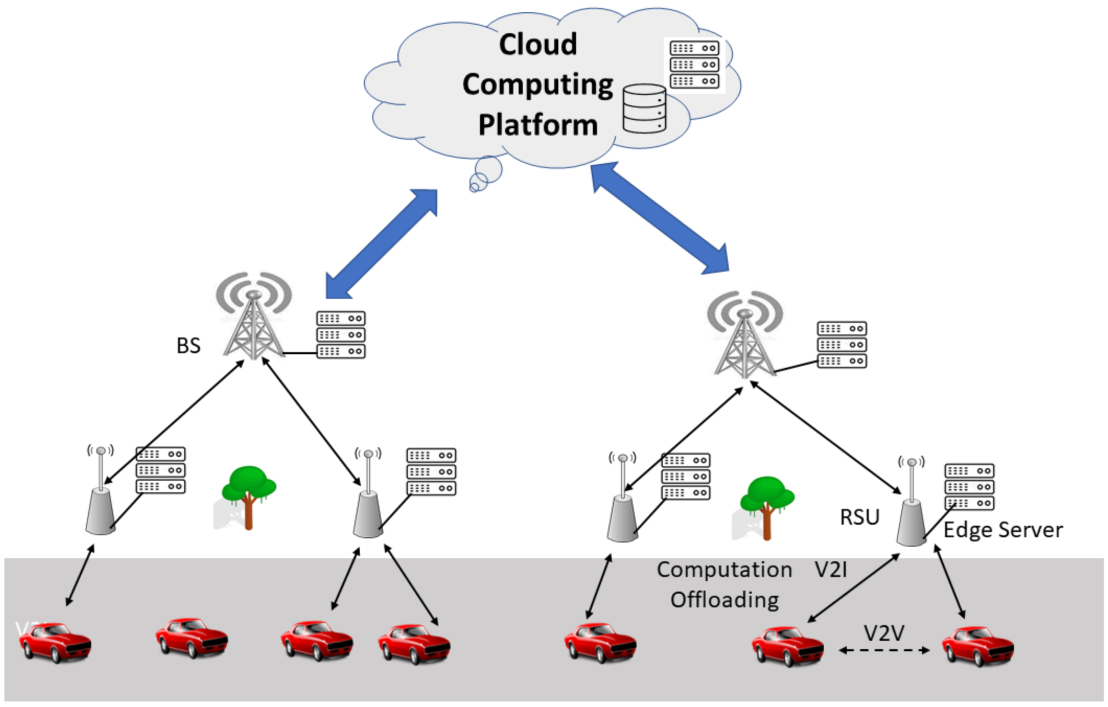


FIGURE 2.1: Vehicular edge computing framework.

The limited available resources and RSU coverage are the main bottlenecks for reducing the performance of a VEC system. The resource limitation often increases the VEC system cost in terms of computation latency when multiple VUs request services from the same EC server. The RSU coverage limitations, along with VUs mobility, can add an extra cost in terms of handover latencies. In some works, authors have considered hybrid system models in which both VEC and Cloud computing facilities are considered together for solving the capacity and coverage issues in VEC systems [19, 61]. However, such approaches can have some serious drawbacks, such as long transmission distances for accessing cloud facilities. A VN with multiple EC layers of different air and ground networks in proximity (e.g., base station, LAPs, HAPs) can be a better approach compared with a hybrid VEC-cloud system. It can reduce the transmission latencies or network congestion caused by multiple VUs requesting cloud resources and solve the coverage and capacity problems of VEC systems alone.

Within Multi-EC multi-service VN, selection of proper EC platforms and the amount to be offloaded can improve VNs performance. The offloading operations of the surrounding VUs can be useful for other vehicles for a proper offloading decision, such as selection of EC platform, and amount to be offloaded. Table 2.1 lists the advantages and disadvantages of different EC platforms in vehicular services.

2.3.4 Machine Learning

The traditional approaches used for solving VNs problems include convex optimization, game-theoretic approaches, and several other metaheuristics. These techniques mainly suffer from the heavy computational burden with exponentially increasing search space in large-scale scenarios; this is even more enforced when the considered VN system employs multiple EC nodes, and network softwarization solutions, enabling more flexible implementations. Heuristic approaches used for solving the VN problems with their NP-hardness are not able to adapt to the increasing complexity of the new applications.

Table 2.1: Advantages and disadvantages of different EC platforms for VN services and applications.

EC Platform	**Advantages**	**Disadvantages**
VEC	Reduced Transmission Distance with Line of Sight (LOS) Communication	Limited Resources, Coverage Range, Frequent Handovers
TN-EC (i.e., BS)	Higher Computation and Communication Resources, Better Coverage Range	High Transmission Delay with Degraded Channel Quality (NLOS Communication)
Cloud	Unlimited Resources and No Coverage Issues	Huge Transmission Delay, Backhaul Network Congestion, Security Issued Due to Long Distance Communication Channels
LAP-EC	Reduced Transmission Distances with LOS communication, Reduced Deployment and Maintenance Time and Costs	Limited Resources, Low Flight Time
HAP-EC	Moderate Transmission Distances with LOS communication, Can Have High Resources, Solar Energy Source	High Deployment and Maintenance Time and Costs Compare with LAP, Communication can be Affected by Rain Fading
Satellite-EC	High Computation and Communication Resources	Large Transmission Distances (not suitable for latency critical VN), Large Deployment and Maintenance Cost

With the addition of IoT techniques, enabling what is usually referred to as IoV [18, 60, 62, 63], vehicles become an excellent element for training data that can be utilized for solving vehicular problems (i.e., ML-based solutions). Several complex vehicular problems such as dynamic resource allocation, traffic predictions, cooperative congestion control, content caching, computation offloading, intrusion detection, anomaly detection can be solved by using popular ML methods such as supervised learning, unsupervised learning, reinforcement learning, etc. [35]. Among other ML techniques, Deep Reinforcement Learning (DRL) is a potential solution for many complex vehicular scenarios, which allows exploiting Deep Neural Networks (DNN) for analyzing VNs data without requiring any prior knowledge of the VN environment, which is hard to capture [64], e.g., correct state transmission matrix over VN states for Markov Decision Processes (MDP) based solutions. ML solutions outperform the heuristic and one-shot-based optimization techniques with better long-term performance.

Different approaches are available for the integration of ML-based solutions into vehicular environments. In each of these approaches, various communication, and computation strategies can be involved during the training process of an ML model. For example, a centralized ML model training approach requires each VN to send its data towards a centralized, more powerful server. In another case, a centralized ML server can further split the training data and corresponding operations with nearby servers to reduce the required computation time. A new collaborative learning-based approach such as FL can allow participating VUs to train ML models locally. The powerful central server can be employed for collecting and averaging the local training ML models parameters to combine the VUs learning experience. These technologies can have certain advantages, disadvantages and can face several new challenges in vehicular environments. In the following, we describe each of these approaches in detail.

Centralized ML

Even though VUs are capable of training fairly complex ML algorithms by themselves, it is yet not recommended due to several reasons. First, with their limited resources, training complex ML algorithms over Vehicles Onboard Units (OBUs) will be computationally expansive and can introduce large latency and energy costs. Complex ML techniques such as DNN require a large amount of data during training, which individual VUs are not capable of providing. Furthermore, dynamic environments like VNs are continuously changing, and the surrounding environments can affect the VUs performance while performing ML model training.

In the centralized learning approach, several randomly distributed VUs transmit their raw data towards a powerful centralized server (ML-server) with rich communication and computation resources. With this approach, fairly complex ML techniques like DNN can be adapted to solve VNs problems with better performance. With these advantages, some challenges need to be considered while implementing centralized ML-based solutions over vehicular environments. Though centralized computation servers are rich with computational resources, ML model training costs can grow exponentially with the increasing complexity of ML techniques (DNN with a high number of layers), which ensures large computation delays at servers. Furthermore, the transmission of the whole dataset towards centralized servers can be challenging with VUs limited communication resources and dynamic channel environments. In the case of VNs, with changing environment dynamics and corresponding renewed datasets, it is important to update the trained ML models for a short duration of time to avoid issues like model drift. Thus, the centralized ML model training approach can have several issues when considering it for solving VNs problems.

Figure 2.2 shows an example of a centralized ML model training over VN where a centralized ML server collects training data from individual VUs for performing training operations.

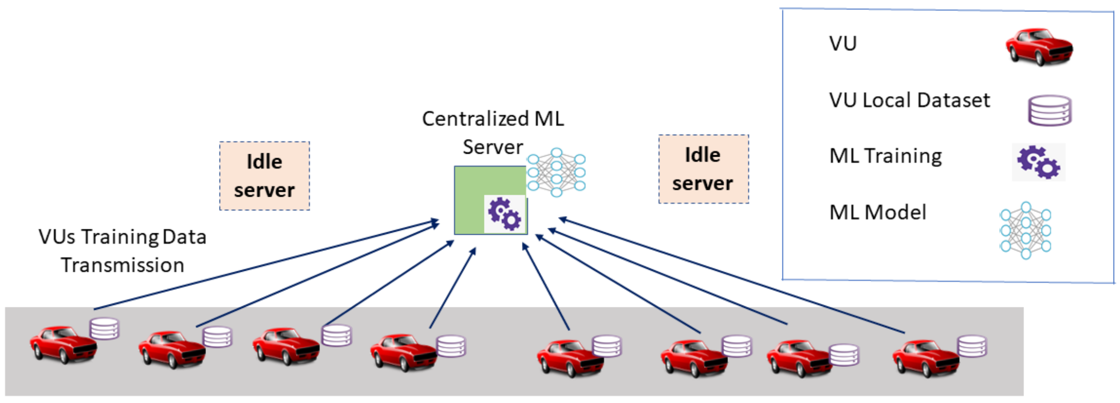


FIGURE 2.2: Centralized machine learning.

Distributed ML

For reducing the model training cost at a centralized ML server, the workload distribution methods can be adapted, in which the main server can select a set of powerful computing servers around it and allocate the model training tasks. This approach is known as a distributed learning approach in which multiple powerful servers collaboratively perform the training process. This allows to limit the model training latency at the centralized server and allows to train fairly complex ML models with acceptable latency. However, this approach does not solve the communication overhead problem faced by the individual VUs and thus can have limited performance over VNs.

Figure 2.3 shows the distributed ML model training over VN. In the case of distributed learning, the centralized server employs the nearby idle server resources for reducing the overall training latency.

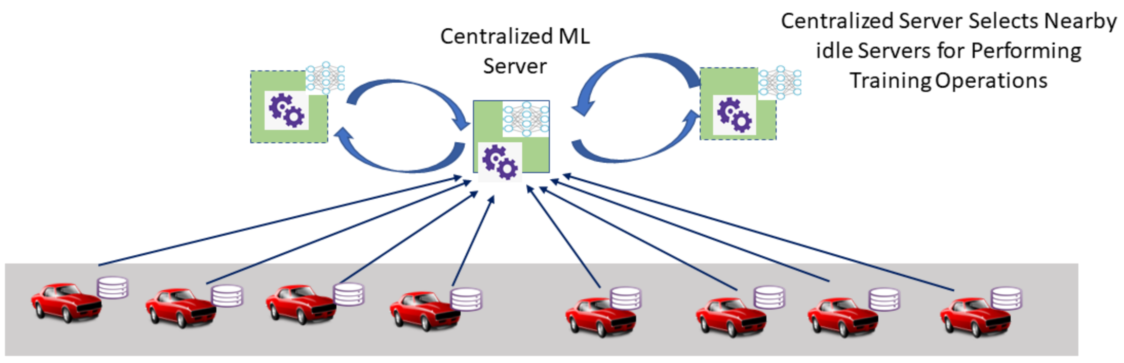

Centralized ML Server
Centralized Server Selects Nearby idle Servers for Performing Training Operations

FIGURE 2.3: Distributed machine learning.

Federated ML

For overcoming the problems of the centralized and distributed approaches, the FL technique is proposed, where individual devices can perform model training by themselves exploiting local data and transmit only the ML model updates towards a centralized server In some articles, authors do not distinguish between Distributed Learning and Federated Learning. However, here we have considered these two as separate ML model training techniques with different features [65]. The main differences between FL and distributed learning approaches are listed in Table 2.2. Once receiving updates from all vehicles in the coverage area, a centralized server performs an averaging operation (i.e., Federated Averaging (FedAvg)) for creating a centralized global model, whose parameters are then transmitted back to the vehicles. The averaging process allows individual vehicles to take advantage of other vehicles' data and learning experiences for improving their ML models while the local device training process improves the time and energy efficiency of the model training process. Thus, a single FL process communication round includes several steps, such as individual VUs performing the local training operations, the transmission of the local model parameters towards the centralized server, after receiving parameters from an individual VUs, the averaging operation performed at centralized server for creating a global model that allows VUs to take advantage of other VUs training experiences and retransmission of model parameters back to VU. Such communication rounds can be performed for creating a suitable ML model with lesser estimation errors. The complete FL process over VN is shown in Figure 2.4.

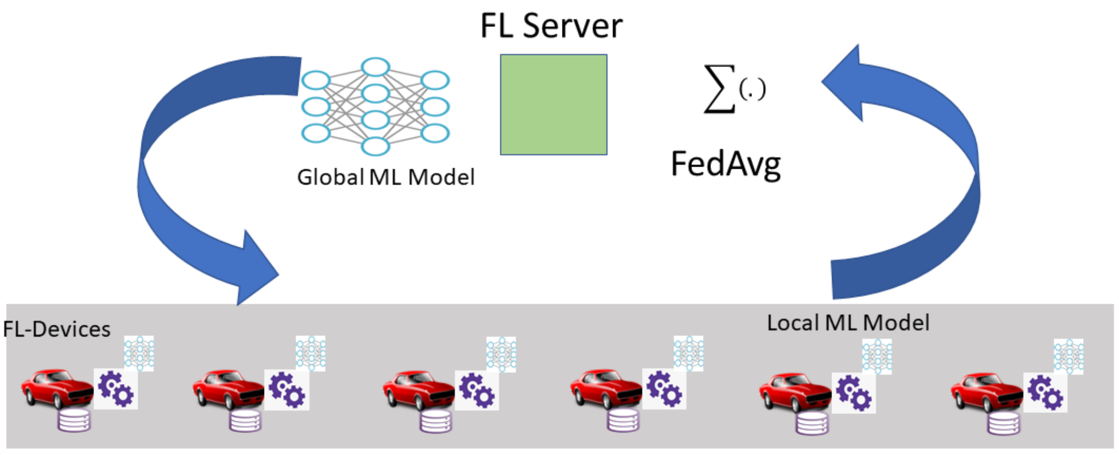


FIGURE 2.4: Federated learning.

Algorithm 1 lists the possible steps involved during the FL process. FL process requires several input parameters, including the number of FL devices participating (M), their datasets ($\{D_m\}$), and the maximum number of FL communication rounds ρ. It should be noted that the FL communication rounds limit can also be replaced by any other stopping criteria such as model convergence parameter, loss function value, etc. The initialization step begins the FL process by defining the initial value of a global FL model parameter and FL communication round to zero (Line 1). At the beginning of each round, local model parameters are updated by the previous rounds' global parameter values (Line 3). Next, FL devices perform the ML model training for generating local ML model w_m^{it} at it^{th} iteration in parallel (Lines 4–5), where η is a learning rate. After completing the local training process, each device forwards its parameters to the FL server (Line 6), where it performs the FedAvg operation to create a new global model (Line 8) which is then used by VUs in next round of communication. FL process continues over ρ communication rounds.

Algorithm 1 Federated Learning.

```
Input: $M, \rho, \{D_m\}$
Output: $w_G^{\rho}$

1: Initialize $w_G^0 \wedge it = 0$
2: for each $it = 1, \cdots \rho$ do
3:     initialize $w_m^{it} = w_G^{it-1}$
4:     for each $m = i, \cdots M$ do in parallel
5:         Update $w_m^{it}$ based upon the $\eta$ and the local device learning process.
6:         send $w_m^{it}$ to FL Server
7:     end for
8:     FL server collects all the $w_m^{it}$ and performs averaging
9:     $w_G^{it} = \frac{1}{M} \sum_{m=1}^{M} w_m^{it}$
10: end for
11: return $w_G^{\rho}$
```

In general FL FL-based solutions can have many benefits in VN environments, including reduced communication overheads compared with centralized/distributed learning models. However, in the case of a Multi-EC enabled VN, selecting a proper FL server, FL devices, and the proper number of FL communication rounds can be beneficial in terms of training costs.

Table 2.2: Characteristics of the ML Models over VNs.

Characteristics	**Centralized Learning**	**Distributed Learning**	**Federated Learning**
Learning Entity	Centralized Server	Distributed Centralized Servers	On Device (VUs)
Communication Cost/Latency	High	High	Limited
Computation Cost/Latency	High	Limited	Limited
VUs Sensitive Data Privacy	Less	Less	High
Useful for	Training Fairly Complex ML models (i.e., DNN with limited number of layers)	Training Complex ML Models	Training Models with Limited Complexity

2.3.5 Non-terrestrial Networks

Non-terrestrial networks have added advantages in terms of global coverage, resilience towards natural disasters, flexible and cost-efficient deployments, and reduced energy costs compared to terrestrial networks. The next generation of wireless technologies is expected to utilize the different NTN layers for providing capacity and coverage boosts for terrestrial networks. The joint T-NTN has been considered an important enabling technology in the 6G vision. The VNs can benefit from NTN platforms to satisfy the end users' demands. The different NTN platforms such as LAP, HAP, and LEO satellites can be considered to enable latency critical services in VNs.

2.4 Multiple Edge Computing Platforms Enabled Joint Terrestrial and Non-terrestrial Network Architecture for Vehicular Scenarios

In this section, we propose a VN architecture with multiple EC layers to serve VUs with diverse service requests. We characterize different EC platform layers in detail and highlight their importance for serving VUs.

TN-based infrastructures are playing an important role in creating a fully functional intelligent VN for futuristic transportation systems. However, they alone are not capable of providing adequate services to dynamic VUs, which often request services with extremely low latency and high reliability. TN is vulnerable to ground-based security attacks due to its fixed positions. Furthermore, in the case of natural disasters such as tsunamis, and earthquakes users often fail to connect to the services of TN. Low accessibility into remote areas mainly because of the unwillingness of mobile operators to provide services in low revenue parts needs to be considered while integrating TN into VNs. NTN has a considerable advantage over TN in terms of availability, reliability, scalability, and low deployment costs. With these advantages, they can play an important role in complimenting TN for providing better quality services to VUs. Both TN and NTN platforms can enable EC-based services through the placement of edge computing servers along with their distributed infrastructures. Therefore, a Multilayered joint T-NTN constituted by different EC platforms over TN and NTN can be utilized for providing heterogeneous services requested by VUs with demanded quality.

In Figure 2.5, we propose a joint T-NTN architecture having multiple EC platform layers and jointly exploiting TN and NTN for serving VUs.

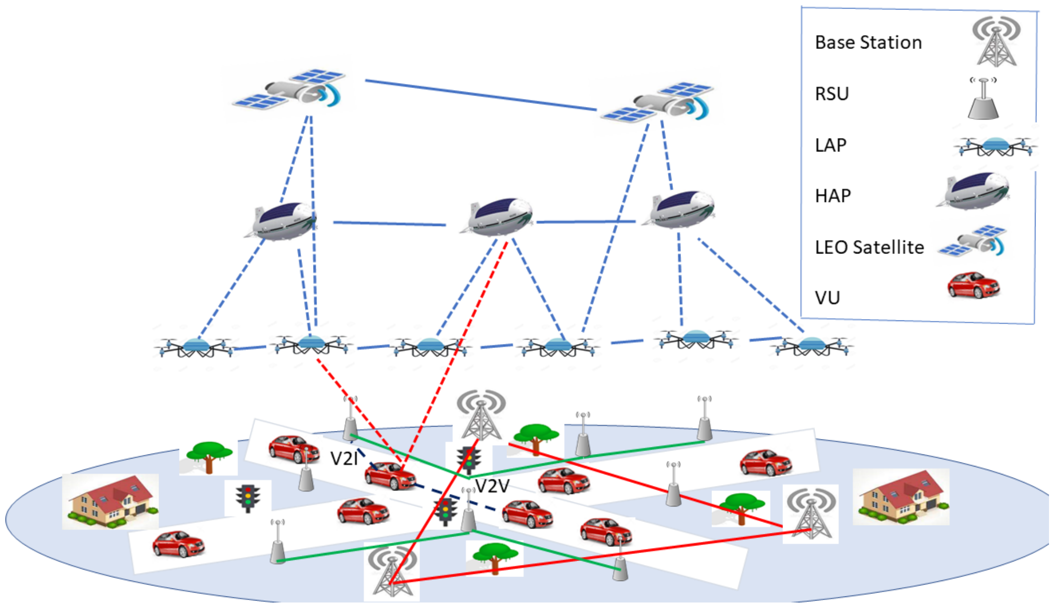


FIGURE 2.5: Multiple EC enabled VN.

Several VUs are randomly distributed in the considered service area and demand many latency-critical and data-intensive services. The proposed network architecture is constituted by several elements as shown in Figure 2.5: a set of VUs, RSU, BS elements, LAPs, HAPs, and LEO satellites. For avoiding redundancies, in the following, we omit the legend from the figures. A set of RSUs is deployed alongside roads having limited capacity EC servers for providing computation services to the VU. Each RSU can serve a limited set of VUs in its coverage range. VUs are also supported by the BS elements equipped with EC services. BS elements can sever VUs over larger coverage areas compared with RSUs and can have powerful EC servers. However, the required roundtrip time for sending and receiving back the VUs task can be much higher. Thus, RSUs and BC servers jointly form a multilayered TN-based EC service platform for supporting VUs. VUs are also complemented by a swarm of LAPs (i.e., UAVs) deployed on top of them. UAVs can enable limited EC services with a pre-installed EC server. UAVs can have limited coverage and flight time which often limits their service range. VUs are also covered by multiple decentralized HAPs having better coverage and computing capacity compared with UAVs. Additionally, an LEO satellite constellation is also considered for serving VUs with better coverage. Thus, Jointly, UAVs, HAPs, and LEO satellites form a multilayered NTN-based EC platform for supporting VUs with better quality services and intelligent applications. Jointly both TN and NTN-based EC platforms serve VUs with their available computation and communication resources.

As shown in Figure 2.6, the considered network infrastructure can be split into several layers of ground-based and areal networking infrastructures. The TN is constituted by several connected VUs in Layer 1 which can form a small vehicular cloud, RSUs equipped with the EC servers in Layer 2, and multiple BS with EC facilities in Layer 3. The NTN has two layers of LAPs and HAPs belonging to the areal networks. LAP nodes are located at a relatively low distance from the ground compared with the HAPs and have limited computation and communication resources. Though HAPs are at a long distance from the VUs layer they can serve large coverage areas. The space-based LEO satellites can further the coverage and capacity of the network. Below we, describe each of these networking layers in detail.

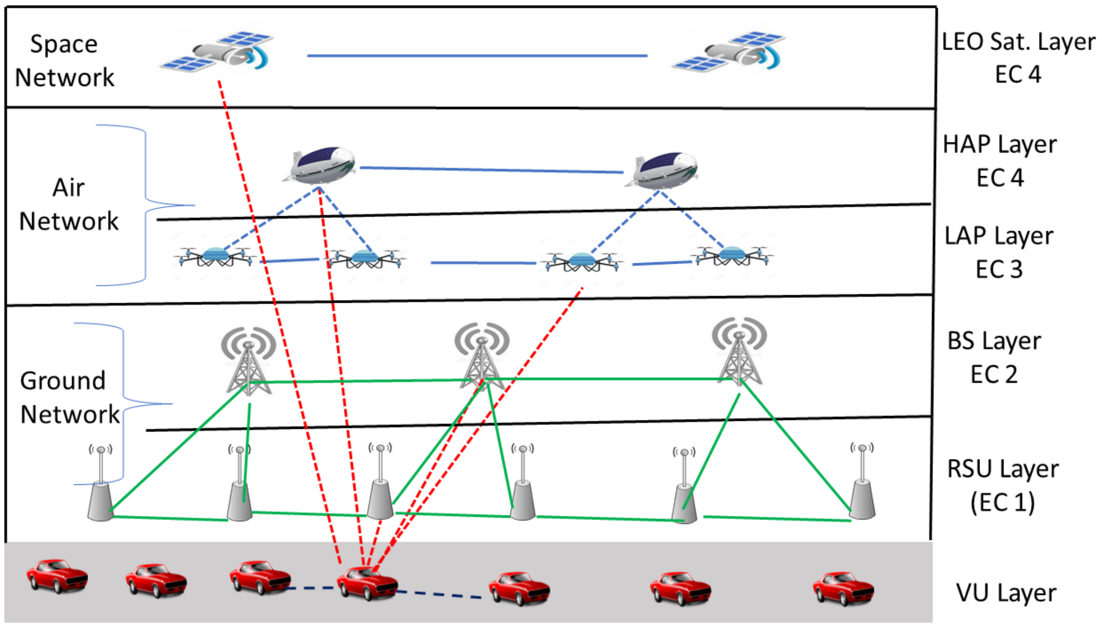


FIGURE 2.6: The Multi-EC framework for the vehicular scenario.

Layer 1 Connected VUs layer Several connected VUs having communication and computation capabilities are grouped into Layer 1. Each VU can communicate with its neighboring VUs for possible information sharing through V2V communication technologies and use V2I links for interacting with other EC layers. VUs can communicate over a limited distance to share important information for enabling several safety-related, traffic flow management services that make drivers' lives easy on the road. Through V2I communication, VUs can share their workloads with EC servers in proximity to enable latency-critical and data-intensive applications. VUs often generate different task requests with specific requirements for computation, communication, and storage resources. These task requests often come with additional requirements (i.e., critical latency requirements), for which VUs often need assistance from the EC platforms in the proximity.

Layer 2 RSU-Edge Computing (RSU-EC) Layer For enabling the EC facilities into VNs, several RSUs have been deployed alongside road infrastructures. RSUs can have communication technologies installed for communicating with VUs and other higher-lever networking layers (i.e., cellular BSs, cloud infrastructures, etc.). EC servers having limited computation and storage resources can be deployed alongside RSUs to integrate the EC services into VNs. Thus, RSUs equipped with EC servers and communication technologies constitute an EC layer in the proximity of VUs for providing low-latency services. However, with their limited resources and coverage, RSUs can serve a limited number of VUs. Furthermore, VUs mobility often restricts them from accessing RSU services for longer periods of time.

Layer 3 5G Base Station (BS) Layer 5G base stations (5G-gNB) have integrated modern communication and computation technologies that can be exploited as an EC platform. With larger coverage areas, they can serve a higher number of VUs. However, the BS-EC platform can have additional transmission delays due to longer communication distance compared with RSUs.

Layer 4 Low Altitude Platform (LAP) Layer LAP platforms such as UAVs equipped with EC servers can add several benefits in terms of a reduced transmission time (less than 1 msec), reduced deployment and maintenance time, line of sight communications with better channel quality, etc. They can also act as a relay node for allowing VUs to transmit their information towards higher air networking layers such as HAP with reduced latency and energy costs. With the limited size and reduced flight times, UAVs can only have limited communication and computation resources.

Layer 5 High Altitude Platform (HAP) Layer A HAP network constituted by several nodes able to communicate amongst themselves and other infrastructures is forming another EC layer in the air network. Each HAP node can have a powerful EC server and the communication resources for serving VUs under its coverage. Better coverage, additional renewable energy sources, and better stability are some of the HAPs main advantages over LAPs. However, HAP performance can be reduced by longer communication distances, additional channel loss in terms of rain fading high deployments, and maintenance time and costs. The HAPs coverage area can depend upon its altitude. In general HAP platform can provide coverage between a few 10s of Kms to up to several 100s of Km [66, 67].

Layer 6 Low Earth Orbit (LEO) Satellite Layer A set of LEO satellites can enable the space-based distributed communication and computation facilities for serving VUs with better coverage can capacity. The distance from the VUs can be much higher compared to the other air networking platforms.

Communication, Computation and Storage Characteristics

Each EC platform layer of the proposed network architecture can adapt different computation and communication strategies. Several virtualization techniques (i.e.,VMs, containers) can be used for the efficient utilization of EC resources. Furthermore, an SDN-based centralized control approach can be applied for managing the computation and storage resources of individual EC platforms. Multiple operators based communication technologies can be adapted for enabling the communication between EC nodes of the same and distinct layers. Table 2.3 lists the most important characteristics of individual EC layers considered in the proposed network architecture.

Table 2.3: EC platforms' characteristics.

EC Platform	**VU Cloud**	**RSU EC**	**LAP EC**	**5G-BS EC**	**HAP EC**	**LEO EC**
Computation Resources	Low	Limited	Limited	High	High	High
Communication	Low	Limited	Limited	High	High	High
Storage	Low	Limited	Low	High	Limited	Limited
Coverage	few 10s m^2	few 100 m^2	few km^2	few km^2	up to 200 km^2	100s of km^2
Energy Source	Electric/Fuel Cells	Electric Grid	Fuel Cells	Electric Grid	Fuel Cells/Solar	Fuel Cells/Solar

In this dissertation, we aim to explore the proposed network architecture for enabling distributed data processing and machine learning solutions for VUs. In the beginning, we explore the possibility of considering terrestrial edge computing networks for vehicular data processing with a single and multi-service approach. The scenarios are then explored towards non-terrestrial networks and joint T-NTN cases. In the different vehicular scenarios considered throughout the thesis work.

Chapter 3

Future IoV Network, Key Challanges and Possible Solutions

3.1 Introduction

IoV technology can integrate IoT scenarios with advanced communication modes such as V2X, cellular communication, etc, for providing connectivity and data communication services to vehicular users. With the support of 6G technology, the concept of IoV is expected to evolve and converge into a more advanced fully connected, and intelligent system. However, some key challenges are required to be addressed to create a reliable vehicular network with inherent intelligence and advanced services. With the support of IoV technology vehicular users are expected to demand various high-quality services with extreme requirements in terms of latency, energy, processing, and intelligence. This induces the challenge of processing the vehicular data effectively in edge-based distributed computing environments. Additionally, with the ever-growing demands of intelligence in vehicular networks, proper distributed learning frameworks with the support of vehicular IoT data and edge networks are required.

3.2 Distributed Data Processing for Vehicular Users

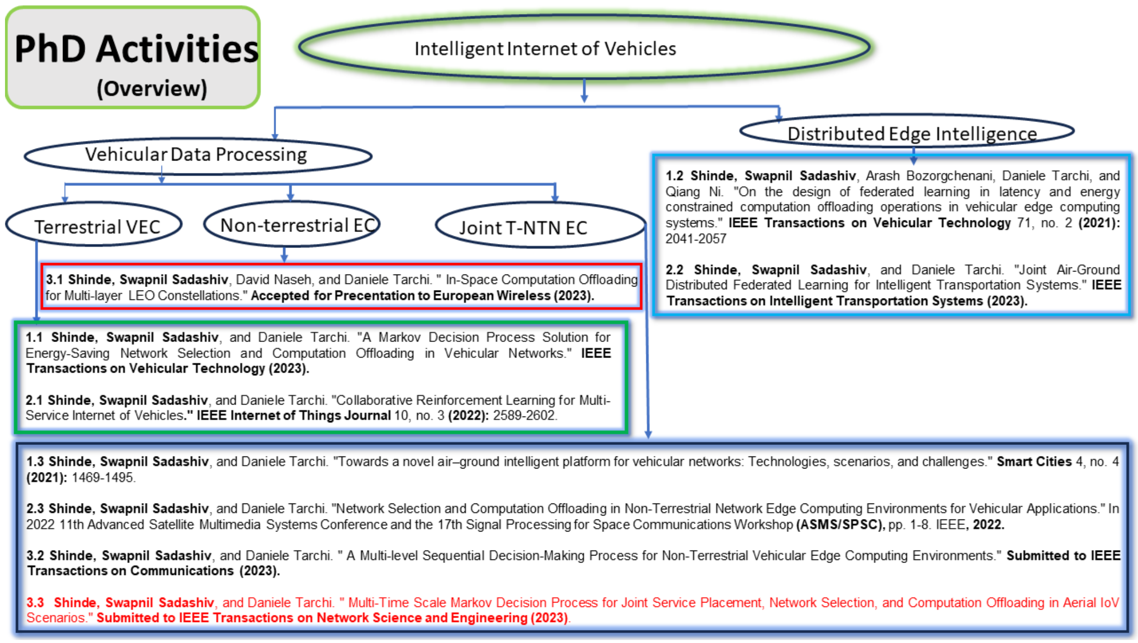
The novel vehicular services demand much stronger KPIs in terms of latency, reliability, data rates, etc. Additionally, each vehicular service request comes with a large data processing demand. Traditional cloud computing frameworks were considered to process the vehicular user's data. In general cloud facilities are often located far away from the end users and the overall transmission distances can be large. This can induce additional challenges in terms of longer transmission delays, backhaul congestions, privacy issues, etc. To overcome this challenge in the last decade the concept of mobile edge computing (MEC) was proposed. MEC technology brings the cloud computing facilities in the proximity of end users with the deployment of computation and communication resources alongside access networks. This car addresses the major concerns of cloud-based systems by allowing end users to process their data efficiently. In the case of vehicular networks, edge computing facilities can be enabled through the integration of edge resources along with roadside units (RSUs) and cellular base stations. This integration of MEC facilities in the vehicular case is known as Vehicular Edge Computing (VEC) and is considered as a promising technology to enable futuristic vehicular services. VEC technology can enable latency-critical services in vehicular networks by allowing vehicular users' data to be processed at the edge. Such distributed computing frameworks can be extremely important for different IoV scenarios. However, to have proper benefits from the resource-limited VEC facilities several challenges should be addressed. In particular, the RSU often has a limited coverage range, and with vehicular users' mobility, the handover can be a big challenge. With the limited coverage range, only a few users can be served by each RSU terminal. Additionally with limited storage resources the number of services provided by VEC facilities can be limited. The limited computation resources can restrict the amount of data processed by each VEC server before latency costs become unbearable. Therefore, for the case of multi-user vehicular scenarios, it is important to select a proper edge node for processing the vehicular user's data. This can be defined as a *network selection* problem for multi-user edge computing scenarios. With advanced vehicular nodes, some amount of data can be processed by vehicular terminals. With this, partial computation offloading becomes essential where a parallelization can be added by splitting the vehicular data into two or more parts for joint processing at the edge and vehicular sides. However, it is important to optimize the amount of data to be processed at the edge for proper benefits. This can be defined as a *partial offloading* problem in edge environments. For the case of multi-service scenarios, the number of services provided by each edge node can be limited given the restricted storage resources. Therefore proper service placement over edge networks can be important for providing the requested services. This can be defined as a *service placement* problem in edge environments.

3.3 Distributed Intelligence for Vehicular Users

To enable intelligent solutions in vehicular networks and to achieve the goal of fully connected and intelligent vehicular networks proper machine learning frameworks are required. In the case of latency-critical vehicular scenarios considering the traditional centralized ML solutions can be difficult and costly. On the other hand with the expansion of IoT technology, a large amount of distributed vehicular data is present at the device level in vehicular systems. This data can be effectively used to enable large-scale intelligence into vehicular networks. To this scope novel learning paradigms such as distributed learning can be extremely important. In the case of different distributed learning frameworks, the data can be processed and analyzed in distributed manners subtracting the need for collecting it at the centralized servers which is a central demand of traditional learning frameworks. Such distributed learning frameworks can be implemented in vehicular networks with the support of distributed vehicular nodes and edge computing facilities. The learning frameworks such as federated learning, collaborative learning, and multi-agent solutions can be adapted to enable efficient intelligent solutions in vehicular networks. Even though such distributed learning frameworks can have added advantages compared to the traditional case of centralized learning, further optimization can be helpful to satisfy future vehicular networks. Different technologies can be used to optimize the learning solutions such as federated learning over vehicular networks. This includes the V2X frameworks, multi-layered edge networking scenarios, advanced resource-sharing solutions, etc. This thesis work includes several of such solutions with the support of terrestrial and non-terrestrial edge computing platforms.

3.4 Considered Problems and Outcomes

As shown in figure 3.1, this thesis work explores the different IoV scenarios for enabling efficient data processing and intelligent solutions at the edge. The different vehicular scenarios, corresponding problems, and outcomes are highlighted in the figure.


PhD Activities
(Overview)
Intelligent Internet of Vehicles
Vehicular Data Processing
Distributed Edge Intelligence
Terrestrial VEC
Non-terrestrial EC
Joint T-NTN EC

3.1 Shinde, Swapnil Sadashiv, David Naseh, and Daniele Tarchi. " In-Space Computation Offloading for Multi-layer LEO Constellations." **Accepted for Precentation to European Wireless (2023).**

1.1 Shinde, Swapnil Sadashiv, and Daniele Tarchi. "A Markov Decision Process Solution for Energy-Saving Network Selection and Computation Offloading in Vehicular Networks." **IEEE Transactions on Vehicular Technology (2023).**

2.1 Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Collaborative Reinforcement Learning for Multi-Service Internet of Vehicles." **IEEE Internet of Things Journal** 10, no. 3 **(2022):** 2589-2602.

1.2 Shinde, Swapnil Sadashiv, Arash Bozorgchenani, Daniele Tarchi, and Qiang Ni. "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems." **IEEE Transactions on Vehicular Technology** 71, no. 2 **(2021):** 2041-2057

2.2 Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Joint Air-Ground Distributed Federated Learning for Intelligent Transportation Systems." **IEEE Transactions on Intelligent Transportation Systems (2023).**

1.3 Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Towards a novel air–ground intelligent platform for vehicular networks: Technologies, scenarios, and challenges." **Smart Cities** 4, no. 4 **(2021):** 1469-1495.

2.3 Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Network Selection and Computation Offloading in Non-Terrestrial Network Edge Computing Environments for Vehicular Applications." In 2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop **(ASMS/SPSC),** pp. 1-8. IEEE, **2022.**

3.2 Shinde, Swapnil Sadashiv, and Daniele Tarchi. " A Multi-level Sequential Decision-Making Process for Non-Terrestrial Vehicular Edge Computing Environments." **Submitted to IEEE Transactions on Communications (2023).**

3.3 Shinde, Swapnil Sadashiv, and Daniele Tarchi. " Multi-Time Scale Markov Decision Process for Joint Service Placement, Network Selection, and Computation Offloading in Aerial IoV Scenarios." **Submitted to IEEE Transactions on Network Science and Engineering (2023).**

FIGURE 3.1: Considered Problems and Outcomes.

3.4.1 Vehicular Data Processing Problem

- **Terrestrial Case :** In the beginning, the vehicular data processing problem is explored for terrestrial VEC cases where edge computing facilities are enabled through the ground-based RSU and base station units. In particular, a joint network selection and offloading problem is considered for minimizing the latency and energy costs. The cost analysis includes both edge node side and vehicular side costs. A Markov decision process-based solution is proposed with the support of V2X technology. Next, we expand the considered scenario for multi-service cases where users can demand diverse service types. The considered offloading problem is solved through the innovative collaborative reinforcement learning-based approach. In particular, two collaborative deep reinforcement learning-based solutions based on V2V and V2I technologies are proposed.

- **Non-terrestrial Case :** The non-terrestrial networks are expected to play a key role in enabling the next generation of intelligent vehicular systems. To this sense, we have explored the case of non-terrestrial network-based edge computing systems for serving remote users with different edge computing-based services. Multiple LEO satellite constellations along with the cloud facilities are considered for serving users with a diverse set of services. next, we proposed a hierarchical reinforcement learning-based solution for solving the network selection problem over multi-layered non-terrestrial edge computing facilities.

- **Joint terrestrial and Non-terrestrial Case :** The non-terrestrial networks can support the terrestrial edge computing facilities with capacity and coverage boosts. Such multi-layered edge computing-enabled network architectures can serve vehicular users with multiple services. Therefore we have proposed several novel solutions for enabling efficient data processing over such multi-layered joint terrestrial and non-terrestrial networks for vehicular user cases. The problem of network selection and computation offloading over a multi-layered ground-air and space network with multiple services is solved through an adaptive genetic algorithm for minimizing the latency and energy costs. Next, a multi-level sequential decision-making process is also adapted to solve the offloading problem over challenging multi-service, multi-layered edge computing frameworks. We have also explored the multi-time scale approach for solving the service placement, network selection, and offloading problem with different time scales. An innovative, Multi-time scale Markov decision process-based framework with time-dependent state transition probabilities is proposed to minimize the latency and energy costs.

3.4.2 Distributed Edge Intelligence for Vehicular Users

Novel machine learning solutions based upon vehicular IoT data and edge computing facilities are required to enable large-scale distributed intelligence in vehicular networks. One of the key requirements of 6G-enabled IoT systems. The edge intelligence framework can merge machine learning technologies with edge computing facilities to create proper learning solutions in the proximity of end users. Adapting such frameworks over vehicular systems with distributed computing resources can be of extreme importance. In particular novel distributed learning frameworks that can adapt according to vehicular users' demands, network resource availability and the dynamicity of the systems can be considered for building the future intelligent vehicular systems. Among others, federated learning is one of the highly explored distributed learning frameworks that can serve the end users effectively. In this thesis work, we have proposed a federated learning framework over a resource-limited vehicular system that can jointly optimize the federated learning and offloading process costs. Next, we have explored the joint air-ground network to enable the effective distributed learning framework for serving vehicular networks. In this sense, we have proposed a distributed federated learning solution that effectively considers the multi-layered distributed computing environment of air-ground networks.

In the following chapters, we describe each of these scenarios, the considered problem, and the proposed solutions with performance analysis in detail.

Chapter 4

Distributed Data Processing for IoV -Terrestrial Case

Some content of this chapter is based on the following articles [68, 69];

1) " Shinde, Swapnil Sadashiv, and Daniele Tarchi. "A Markov Decision Process Solution for Energy-Saving Network Selection and Computation Offloading in Vehicular Networks." IEEE Transactions on Vehicular Technology (2023).".

2) " Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Collaborative Reinforcement Learning for Multi-Service Internet of Vehicles." IEEE Internet of Things Journal 10, no. 3 (2022): 2589-2602.".

4.1 Introduction

New vehicular terminals are capable of providing novel services and applications to vehicular users (VUs) aiming at increasing road safety, avoiding traffic congestion, reducing pollution levels, providing new infotainment services, etc. However, modern applications and services come with stringent requirements in terms of high data processing and critical latency bounds. With limited onboard resources, vehicles alone cannot cope with such requirements and need support from additional platforms, e.g., cloud and edge computing [19]. Though cloud computing facilities have enormous computing resources, since they are located deep inside the core networks, high transmission delays often limit their uses for latency-critical VNs. Edge Computing (EC) technology can address the cloud computing problems by bringing the cloud resources in the proximity of end-users. EC has achieved great success in wireless networks when serving users with new innovative services [70]. In VNs, EC facilities can be enabled through the deployment of Road Side Units (RSUs) along the road facilitating several EC servers [71]. This approach, known as vehicular edge computing (VEC), has the potential to serve VUs with reduced transmission delays and energy requirements. The importance of VEC in the VN scenarios is highlighted by several works in the recent past, mainly for enabling latency-critical applications [72, 73].

VEC technology provides a computation environment to VUs for processing their tasks. VUs can transmit a portion of their computation load to the nearby VEC servers while performing the remaining computation locally. VEC servers perform the processing operations on behalf of the VUs and return the results. This approach is known as partial computation offloading, which allows VUs to complete a task processing operation in collaboration with VEC servers to reduce the overall latency and energy requirements during processing [71]. However, when coping with a large number of VUs demanding computation offloading services from VEC servers having limited computation/communication resources, energy limitations, storage capabilities, and coverage range, several new challenges arise in VEC-enabled VNs. Due to the dynamic nature of Vehicular Networks (VNs), offloading a large amount of data to the RSUs without considering vehicular mobility could seriously degrade the QoS. Indeed, with high mobility, each vehicle has a limited amount of time available for data offloading and collecting the results from the RSUs. If the vehicle passes through the RSU coverage without completing the offloading operation, it might end up paying higher latency costs due to, e.g., handover and service migration [13]. Moreover, each VEC server can offer a limited number of services to the nearby VUs, hence, selecting the proper edge server for offloading can avoid network congestion. These challenges are mainly characterized by a proper selection of when, where, and how much data needs to be offloaded to the VEC servers for adequate performance. This problem is also known as joint network selection and computation offloading, which aims to find a proper VEC server and the amount of data to be offloaded over dynamic vehicular environments [71].

In the following, we first attempt to solve the joint network selection and computation offloading problem for the case of a generic single-service VN. Next, we extend the idea for the multi-service case.

4.2 Joint Network Selection and Offloading over VN: Single-Service Case

In this work, we have proposed a joint network selection and computation offloading strategy over a mobile VN for overall latency and energy minimization of both vehicular and infrastructure nodes with additional energy-saving mechanisms at the edge infrastructure. An original MDP-based RL framework with time-dependent state transition probabilities is proposed, where local vehicular environment parameters are used effectively. The main contributions of this work are:

- **Joint Network Selection and Computation Offloading Problem Formulation:** We define a joint network selection and computation offloading problem for minimizing the overall latency and energy consumption over VN as a constrained optimization problem, where ENs can be in different energy-saving states, i.e., standby or active, for a more efficient energy-saving behavior.

- **MDP Model with Time-dependent State Transition Probabilities:** The problem is modeled as a sequential decision-making problem and incorporated into an MDP-based model. Various elements of the MDP process including state space, action-space, reward function, and environment dynamics with time-dependent state-transition probabilities are considered.

- **V2X-based on-road scenarios:** Exploiting V2X communication technologies and a proper mobility model, various on-road VUs scenarios are defined for solving the burden of the higher dimensional MDP process without hindering its performance.

- **Value Iteration Method for MDP Policy:** A value iteration-based approach is used for finding the optimal policy for the MDP process. In addition, a set of benchmark methods are considered to analyze the performance of the proposed scheme.

In the following parts, we discuss the considered system model, problem formulation, and the proposed solutions.

4.2.1 System Model and Problem Formulation

In this work, an urban Internet of Vehicles (IoV) scenario for intelligent transportation systems with connected and intelligent VUs is considered, where a set of randomly distributed VUs over the road network can communicate with the edge computing servers enclosed by RSUs and a Macro Base Station (MBS). In recent times, such urban IoV scenarios have gained a lot of attention from the vehicular research community [74, 75]. We refer to $\mathcal{V} = \{VU_1, \ldots, VU_m, \ldots, VU_M\}$ as the set of M VUs, and $\mathcal{R} = \{RSU_1, \ldots, RSU_n, \ldots, RSU_N\}$ as the set of N RSUs in the area.

The system is modeled in a time-discrete manner, and the network parameters are supposed to be constant over each time interval τ, where τ_i identifies the ith time interval, i.e., $\tau_i = \{\forall t | t \in [i\tau, (i+1)\,\tau]\}$ [76]. By focusing on the ith time interval, the mth VU is located in the position $\{x_m(\tau_i), y_m(\tau_i)\}$, while it moves at a speed $\vec{v}_m(\tau_i)$ along the multi-lane road-path in either direction and equipped with a processing capability equal to c_m Floating Point Operations per Second (FLOPS) per CPU cycle, while its CPU frequency is f_m. We have assumed resource-limited edge computing nodes equipped with muli-core computing hardware with restricted capacities and limited bandwidth resources [77, 78, 79]. Each RSU can be identified through a set of parameters where the nth RSU is located at the fixed position $\{x_n^R, y_n^R\}$ having height h_n^R, able to provide communication with a maximum bandwidth B_n^R, and having a multi-core CPU processor with $\mathcal{L}_n$ cores with c_n^R FLOPS per CPU cycle, while its CPU frequency is f_n^R. Similarly, the MBS can be identified through its position $\{x^{\bar{M}}, y^{\bar{M}}\}$, its height $h^{\bar{M}}$, maximum bandwidth $B^{\bar{M}}$, supposed to be equipped with a multi-core processor, where each core has a processing capability equal to $c^{\bar{M}}$ FLOPS per CPU cycle, while its CPU frequency is $f^{\bar{M}}$. Here, we do not put any limitation over the MBS CPU cores and assume that each VU can have access to only one CPU core.

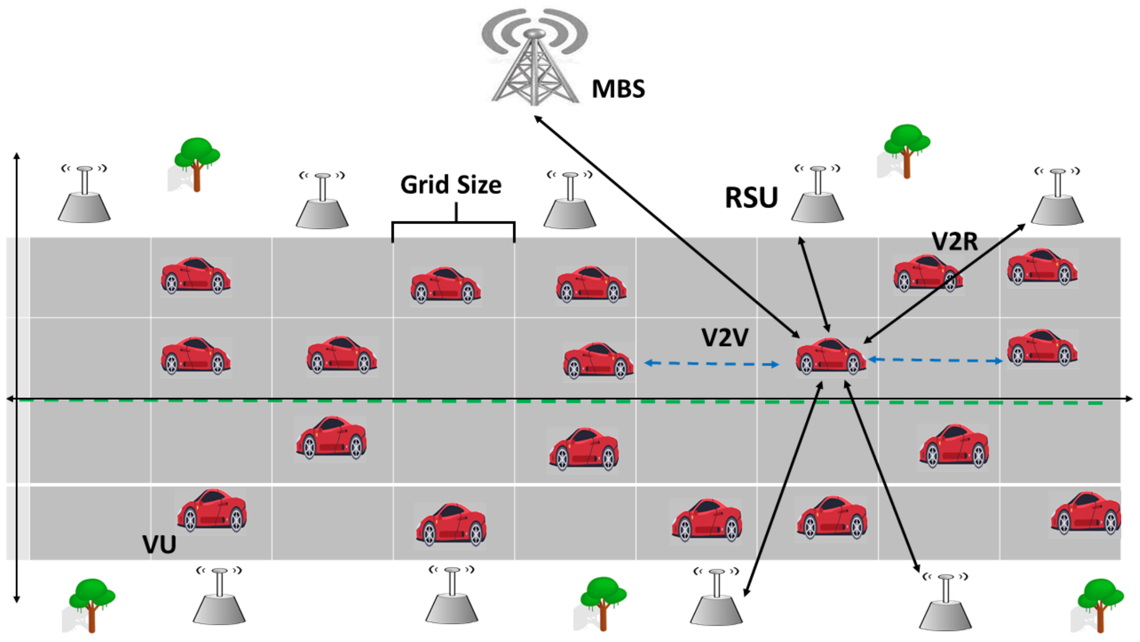


FIGURE 4.1: System Architecture

The nth RSU has a limited coverage range d_n, whose value depends on the communication technology and radio-propagation environment, and it is supposed to provide VEC services to the vehicles within the coverage area. Similarly, for the MBS, the coverage range d^M stands. Thus, VUs can offload data up to $N + 1$ ENs, i.e., N RSUs (i.e., $EN_1, \dots, EN_N$) and one MBS (i.e., EN_0). Each $VU_m \in \mathcal{V}$ is supposed to be active in each time interval with a probability p_a within which it generates a computation task request $\rho_m(\tau_i)$ identified through the tuple $\langle D_{\rho_m}, D^r_{\rho_m}, \Omega_{\rho_m}, T_{\rho_m} \rangle$ corresponding to a task with size D_{ρ_m} Byte, expected to give in output a result with size $D^r_{\rho_m}$ Byte, requesting Ω_{ρ_m} CPU execution cycles and a maximum execution latency T_{ρ_m}.

In Fig. 4.1, a possible IoV scenario is depicted, where randomly distributed VUs are able to offload their computation tasks to the nearby ENs. Also, each VU is covered by multiple RSUs along with one MBS. VUs can communicate with ENs over V2R links and with each other through V2V links for information sharing.

4.2.1.1 VU Mobility and Sojourn Time

Due to the VUs mobility, each offloading operation should be completed by the VU sojourn time, corresponding to the amount of time it remains under the coverage of the selected EN [80], for avoiding additional latency due to, e.g., vehicle handover, service migration, additional signaling for managing vehicles and service mobility. RSU handover process involves transferring the management of active communication from one RSU to another [13]. Such handover situations can occur if VU fails to get back the offloaded task results before it passes through the RSU coverage. The handovers can degrade the network-wide performance in terms of latency.

Individual VUs mobility parameters often depend upon the nearby VUs decisions. One of the most often considered mobility models for vehicular scenarios is based on the preceding car dynamics [81]. Here, we adopt a similar model for analyzing the VUs mobility. If $\vec{v}_{v_m}(\tau_i)$ and $a_{v_m}(\tau_i)$ represent the speed and acceleration parameters at the ith interval for the mth VU, the model consider that the mth VU mobility parameters depend on the motion and dynamics of the preceding VUs, i.e.,

$$a_{v_m}(\tau_i) = a_{max}\left[1 - \left(\frac{\vec{v}_{v_m}(\tau_i)}{\vec{v}_{max}}\right)^{\delta} - \left(\frac{s^*(\vec{v}_{v_m}, \Delta\vec{v}_{v_m})}{s_{v_m}}\right)^{2}\right] \forall m$$

where a_{max} is the maximum acceleration value, $\vec{v}_{max}$ is the desired velocity required for the steady traffic flow, $\Delta\vec{v}_{v_m} = \vec{v}_{v_m} - \vec{v}_{v,m-1}$ and $s_{v_m} = x_{v,m-1} - x_{v_m} - l_o$ are the relative velocity and inter-vehicular distance between m and $m-1$ with l_o being the VUs length. $\delta \in \{1, 5\}$ is the sensitivity of driver, and s^* is the desired space given as:

$$s^*(\vec{v}_{v_m}, \Delta\vec{v}_{v_m}) = s_{min} + t_r\vec{v}_{v_m} + \frac{\vec{v}_{v_m}\Delta\vec{v}_{v_m}}{2\sqrt{a_{max}b_{max}}} \forall m$$

Here, s_{min} is the desired safe space between consecutive VUs, t_r is the minimum reaction time headway based upon the safe distance, and $b_{max} > 0$ is the comfortable braking deceleration. In this work, the safety distance between VUs is considered as a design parameter similar to the [81]. However, the safety distance between VUs can be based upon several parameters and tradeoffs i.e., traffic flow characteristics, VUs safety demands, communication capabilities, V2V delays, etc. Interested readers can follow [82, 83] for more information. Therefore, at the ith interval, the mth VU speed and position are:

$$\vec{v}_{v_m}(\tau_i) = \vec{v}_{v_m}(\tau_{i-1}) + a_{v_m}(\tau_{i-1})\tau \tag{4.1}$$

$$x_{v_m}(\tau_i) = x_{v_m}(\tau_{i-1}) + \vec{v}_{v_m}(\tau_{i-1})\tau + a_{v_m}(\tau_{i-1})\tau^2 \tag{4.2}$$

The distance in which the mth VU remains under the coverage of nth EN is $D_{m,n}(\tau_i)$ and is given by:

$$D_{m,n}(\tau_i) = \sqrt{d_n^2 - \left(y_n^{EN} - y_m(\tau_i)\right)^2} \pm \left(x_n^{EN} - x_m(\tau_i)\right) \tag{4.3}$$

where $\left(x^{EN}, y^{EN}\right)$ is the location of nth EN, i.e., either an RSU or the MBS. The available sojourn time for the mth VU can be written as:

$$T_{m,n}^{soj}(\tau_i) = \frac{D_{m,n}(\tau_i)}{|\vec{v}_m(\tau_i)|} \qquad \forall i, \quad n = 0, 1, \ldots, N \tag{4.4}$$

4.2.1.2 VU-EN Assignment, Offloading Process and Resource Allocation

We define a binary VU-EN assignment matrix $\mathbf{A}(\tau_i) = \left(a_{m,n}(\tau_i)\right) \in \{0,1\}$ with size $M \times (N+1)$. If mth VU is assigned to nth EN in the interval τ_i then $a_{m,n}(\tau_i) = 1$, and $\sum_{n=0}^{N} \sum_{m=1}^{M} a_{m,n}(\tau_i) = M$, where it is supposed that each VU is able to offload data to only one EN. It should be noted that the first column $(n = 0)$ represents the assignments towards MBS, while the remaining columns, from n equal to 1 to N, are considered for RSUs. The number of VUs requesting services from the nth EN is given by $K_n(\tau_i) = \sum_{m=1}^{M} a_{m,n}(\tau_i)$. With their limited resources, RSUs can provide services to the VUs before task communication and computation costs become unbearable. We consider that K^{max} is the maximum number of VUs that can access to the services of each RSU node. However, with rich resource sets, MBS can provide services to several VUs without such limits.

We assume to perform partial offloading, where tasks can be split and processed remotely while the remaining portion is processed locally [77, 78, 80]; the offloaded portion by the mth VU at τ_i is identified as $\alpha_{\rho_m}(\tau_i) \in \{0, 1\}$. With multiple VUs requesting services from the same EN, during the offloading process, the following constraints need to be taken into account $\forall i, n = 1, \ldots, N$:

$$\begin{cases} K_n(\tau_i) \leq K^{max} & \text{(4.5a)} \\ \sum_{m=1}^{K_n(\tau_i)} c_n^{\rho_m}(\tau_i) \cdot f_n^{\rho_m}(\tau_i) \leq (\mathcal{L}_n \cdot c_n^R \cdot f_n^R) & \text{(4.5b)} \\ \sum_{m=1}^{K_n(\tau_i)} b_n^{\rho_m}(\tau_i) \leq B_n^R & \text{(4.5c)} \end{cases}$$

where $c_n^{\rho_m}(\tau_i) \cdot f_n^{\rho_m}(\tau_i)$ is the processing capacity of nth EN assigned to the mth VUs task, $b_n^{\rho_m}(\tau_i)$ is the communication resources assigned to the VU for communicating with the nth EN. Eqs. (4.5) model an upper bound on the number of users connected, processing capacity, and the communication resources of the RSUs. The constraint (4.5a) refers to a system constraint for limiting the complexity of the system model. The edge infrastructure manager can define a strategy for the scenarios where the number of VUs requesting the services from the same RSU node becomes higher than K^{max}. In the considered vehicular scenarios, VUs are forced to perform the local computation of their whole tasks in case the limit is violated[1]. It is worth to be noticed that the capacity of each link depends on the specific communication technology and it is out of the scope of this work. Also, we consider that MBS has abundant resources and is able to serve a large number of VUs without limitations.

With limited EN communication and computation resources, proper scheduling is required when multiple users access. Here, we use the following model for assigning EN resources to the VUs for computation offloading:

$$c_n^{\rho_m}(\tau_i) \cdot f_n^{\rho_m}(\tau_i) = \begin{cases} c_n^R \cdot f_n^R & \text{if } K_n(\tau_i) \leq \mathcal{L}^n \\ \frac{c_n^R \cdot f_n^R}{\left\lceil \frac{K_n(\tau_i)}{\mathcal{L}^n} \right\rceil} & \text{if } \mathcal{L}^n \leq K_n(\tau_i) \leq K^{max} \end{cases} \tag{4.6}$$

$$b_n^{\rho_m}(\tau_i) = \frac{B_n^R}{K_n(\tau_i)} \tag{4.7}$$

Eqs. (4.6) and (4.7) show the EN resource allocation in terms of computation capacity and bandwidth to the VUs' tasks. According to (4.6), if the number of VUs requesting services from the nth EN are less than $\mathcal{L}_n$, each can have access to the single CPU core with capacity $(c_n^R \cdot f_n^R)$. In case the number of users becomes higher than $\mathcal{L}_n$, multiple VUs share CPU core resources. Here $\lceil x \rceil$ is the ceiling function applied over x for rounding it to the nearest integer value higher than or equal to x. According to (4.7), bandwidth resources will be equally shared among all requesting VUs.

If the mth VU is assigned to the MBS, i.e., $n = 0$, it can have access to the single CPU core, and equally shares bandwidth resources with the other connected VUs. Thus:

$$c_n^{\rho_m}(\tau_i) \cdot f_n^{\rho_m}(\tau_i) = c^{\bar{M}} \cdot f^{\bar{M}}, \quad b_n^{\rho_m}(\tau_i) = \frac{B^{\bar{M}}}{K_n(\tau_i)} \tag{4.8}$$

[1]Note that this is just one possible approach that can be adapted by the RSU nodes. Though it is beyond the scope of this work, these decisions can further be optimized based on specific load-balancing techniques.

In the following, we model the delay and energy requirements of various operations involved during the partial computation offloading enabled vehicular task processing.

Task Computation Model

The generic expression for the time and energy spent for the ρ_mth task computation on any device is given by [84]:

$$T_{c,l}^{\rho_m} = \frac{\Omega_{\rho_m}}{c_l f_l}, \quad E_{c,l}^{\rho_m} = T_{c,l}^{\rho_m} P_{c,l} \tag{4.9}$$

where c_l and f_l are the number of FLOPS per CPU cycle and CPU frequency, respectively, whether l identifies a VU (m), an RSU (n) or the MBS $(\bar{M})$. In (4.9), $P_{c,l}$ is the computation power used by the generic lth device.

Task Communication Model

Since we assume to perform a partial computation offloading, each VU transmits a portion of its task to the assigned EN and receive back the result. Similarly, ENs receive tasks from VUs and send back the results. In general, the transmission time and energy between a generic node k and a generic node l for task ρ_k is given by[2]:

$$T_{tx,kl}^{\rho_k}(\tau_i) = \frac{D_{\rho_k}}{r_{kl}(\tau_i)}, \quad E_{tx,kl}^{\rho_k}(\tau_i) = T_{tx,kl}^{\rho_k}(\tau_i) Pt_k \tag{4.10}$$

where $r_{kl}(\tau_i)$ is the data-rate of the link between the two nodes, while Pt_k is the transmission power of kth node. Similarly, the reception time and energy to receive

[2]In the following we identify with l and k the indexes of any generic node. Hence, l and k can have any index among m, n, and $\bar{M}$.

the task of size $D^r_{\rho_k}$ from lth EN by the kth node are:

$$T^{\rho_k}_{rx,lk}(\tau_i) = \frac{D^r_{\rho_k}}{r_{kl}(\tau_i)}, \qquad E^{\rho_k}_{rx,lk}(\tau_i) = T^{\rho_k}_{rx,lk}(\tau_i) Pr_k \tag{4.11}$$

where Pr_k is the power spent for receiving data.

The channel transmission rate between a generic node k and l at the ith interval can be modeled as [85, 86]:

$$r_{kl}(\tau_i) = b^{\rho_k}_l(\tau_i) \log_2 \left(1 + \frac{Pt_k \cdot h_{k,l}(\tau_i)}{\sigma^2 + I_{kl}(\tau_i)}\right) \quad \forall k, l$$

where Pt_k is the transmission power of node k, $b^{\rho_k}_l(\tau_i)$ is the communication bandwidth, σ^2 is the noise power, and $I_{kl}(\tau_i)$ is the interference due to any transmitting node, except k, towards node l, where the total interference during the uplink communication (i.e., VU to RSU) can be calculated as

$$I_{kl}(\tau_i) = \sum_{\forall k' \in K_l(\tau_i) \setminus \{k\}} (Pt_{k'} \cdot h_{k',l}(\tau_i)).$$

For the downlink, instead, we assume to neglect the interference by assuming an orthogonal frequency assignment among RSUs, as well orthogonal RSU to VU transmissions.

EN Operating Modes

For improving the overall energy efficiency, we assume that ENs can be either in a stand-by or an active state. ENs in a standby state will not be able to serve any VU and effectively will reduce the overall energy consumption. A switching process is assumed for switching ENs from standby to active state with additional switching time and energy. The amount of energy consumed for switching the nth EN from standby to active state is [87]:

$$E_{sw,n} = P_{sw,n} \cdot T_{sw,n} \tag{4.12}$$

where, $P_{sw,n}$ is the consumed switching power and $T_{sw,n}$ is the switching time. The amount of time consumed by nth EN for providing offloading services for VU m is given by[3]:

$$T^{\rho_m}_{en,n}(\tau_i) = \frac{T_{sw,n}}{\alpha_{\rho_m}(\tau_i)} + \left(T^{\rho_m}_{c,n} + T^{\rho_m}_{tx,nm}(\tau_i) + T^{\rho_m}_{rx,mn}(\tau_i)\right) \tag{4.13}$$

where $T^{\rho_m}_{c,n}$, $T^{\rho_m}_{tx,nm}(\tau_i)$ and $T^{\rho_m}_{rx,mn}(\tau_i)$ are the time required for the task computation, transmission and reception between nth EN and mth VU, respectively.

The amount of energy consumed will be based on the operating modes. The nth EN will go into standby mode if no service request from any VU in its coverage area is mapped to it, i.e., $a(m,n) = 0, \forall m$. The total energy consumption of all ENs operating in the standby mode is given by:

$$E^{st}_{en}(\tau_i) = \sum_{n=1}^{N^{st}(\tau_i)} E_{en,n}(\tau_i) \quad \text{with} \quad E^{st}_{en,n}(\tau_i) = \tau_i \cdot P_{sd,n} \tag{4.14}$$

where $N^{st}(\tau_i) = \{n \mid K_n(\tau_i) = 0, \forall n\}$ gives the total number of ENs operating in the standby mode. Also, $E^{st}_{en,n}(\tau_i)$ is the amount of energy consumed by the nth EN, where $P_{sd,n}$ is the power consumed during standby mode that depends upon the computation hardware on the nth EN. Similarly, the amount of energy consumed by the nth EN while serving the mth VU is given by[4]:

$$E^{\rho_m}_{en,n}(\tau_i) =$$

[3]Division by $\alpha_{\rho_m}(\tau_i)$ is merely for equation balancing purposes whose effect will be nullified later in (4.17a).

[4]Division by $\alpha_{\rho_m}(\tau_i)$ is merely for equation balancing purposes whose effect will be nullified later in (4.17b).

$$\frac{\tau_i \cdot P_{0,n} + \frac{E_{sw,n}}{K_n(\tau_i)}}{\alpha_{\rho_m}(\tau_i)} + E^{\rho_m}_{c,n} + E^{\rho_m}_{tx,nm}(\tau_i) + E^{\rho_m}_{rx,mn}(\tau_i) \quad (4.15)$$

where $P_{0,n}$ is the power consumed for the basic circuit operations, and $E_{sw,n}$ is the switching energy required. It should be noted that, as the switching operation occurs only once, if the number of VUs requesting services (i.e., $K_n(\tau_i)$) from a particular EN increases, the switching energy per VU scales down. $E^{\rho_m}_{c,n}$, $E^{\rho_m}_{tx,nm}(\tau_i)$ and $E^{\rho_m}_{rx,mn}(\tau_i)$ are the energy required during task computation, transmission, and reception of data between nth EN and mth VU, respectively.

Task Offloading Process

If mth VU is assigned to nth EN, then the time and energy required to offload the portion of the task with offloading parameter α_{ρ_m} to the selected EN and to get back the result in the ith interval is (from (4.10) and (4.11)),

$$\hat{T}^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) = \alpha_{\rho_m}(\tau_i)\left(T^{\rho_m}_{tx,mn}(\tau_i) + T^{\rho_m}_{rx,nm}(\tau_i)\right) \quad (4.16a)$$

$$\hat{E}^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) = \alpha_{\rho_m}(\tau_i)\left(E^{\rho_m}_{tx,mn}(\tau_i) + E^{\rho_m}_{rx,nm}(\tau_i)\right) \quad (4.16b)$$

Also, the amount of time and energy consumed on the nth EN for providing services to the mth VU is given by (from (4.13) and (4.15)):

$$\hat{T}^{off}_{n}(\alpha_{\rho_m}(\tau_i)) = \alpha_{\rho_m}(\tau_i)\left(T^{\rho_m}_{en,n}(\tau_i)\right) \quad (4.17a)$$

$$\hat{E}^{off}_{n}(\alpha_{\rho_m}(\tau_i)) = \alpha_{\rho_m}(\tau_i)\left(E^{\rho_m}_{en,n}(\tau_i)\right) \quad (4.17b)$$

Thus, the total time and energy cost required for the offloading process is given by:

$$T^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) = \hat{T}^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) + \hat{T}^{off}_{n}(\alpha_{\rho_m}(\tau_i)) \tag{4.18a}$$

$$\begin{aligned} E^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) = \\ w_1 \hat{E}^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) + (1-w_1)\hat{E}^{off}_{n}(\alpha_{\rho_m}(\tau_i)) \end{aligned} \tag{4.18b}$$

where (4.18b) is constituted by two parts (i.e., EN and VUs energy) that can be based upon different energy sources and can have different utility costs. Therefore, for having a properly balanced energy cost over the offloading process, we introduce w_1 as a weighting coefficient in the range between 0 and 1.

Local Computation

From (4.9), the amount of time and energy required for the local computation of the remaining task in the ith interval is:

$$T^{loc}_{m}(\alpha_{\rho_m}(\tau_i)) = \left(1 - \alpha_{\rho_m}(\tau_i)\right) T^{\rho_m}_{c,m} \tag{4.19a}$$

$$E^{loc}_{m}(\alpha_{\rho_m}(\tau_i)) = w_1 \left(1 - \alpha_{\rho_m}(\tau_i)\right) E^{\rho_m}_{c,m} \tag{4.19b}$$

Partial offloading Computation

From (4.18)-(4.19), the delay and the energy consumed during the task processing phases when partial offloading is performed (in the ith interval) can be written as:

$$T^{\rho_m}_{m,n}(\alpha_{\rho_m}(\tau_i)) = \max\left\{T^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)), T^{loc}_{m}(\alpha_{\rho_m}(\tau_i))\right\} \tag{4.20a}$$

$$E^{\rho_m}_{m,n}(\alpha_{\rho_m}(\tau_i)) = E^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) + E^{loc}_{m}(\alpha_{\rho_m}(\tau_i)) \tag{4.20b}$$

where the local and offloading processing are supposed to be performed in parallel. Each vehicle should finish the offloading process and receive the result back within the sojourn time, hence:

$$T_{m,n}^{off}(\alpha_{\rho_m}(\tau_i)) \leq T_{m,n}^{soj}(\tau_i) \quad \forall i \tag{4.21}$$

4.2.1.3 Problem Formulation

The main aim of this work is to optimize the network-wide performance of the VEC-enabled VN. We aim to optimize the performance in terms of overall latency and energy consumed during the offloading process towards edge servers by selecting proper ENs and offloading amounts. The latency and energy requirements of both sides (i.e., VUs and RSU-based edge servers) are considered during the offloading process. The joint latency and energy minimization problem is defined as:

$$\mathbf{P1}: \min_{\mathcal{A},\mathbf{A}} \left\{ \sum_{n=0}^{N} \sum_{m=1}^{M} \left[\gamma_1 T_{m,n}^{\rho_m}\left(\alpha_{\rho_m}(\tau_i)\right) + \gamma_2 E_{m,n}^{\rho_m}(\alpha_{\rho_m}(\tau_i))\right] \right. \\ \left. + \gamma_2(1-w_1)E_{en}^{st}(\tau_i) \right\} \forall i \tag{4.22}$$

s.t.

$$\mathbf{C1}: \sum_{n=1}^{N} a_{m,n}(\tau_i) = 1, \quad \forall m \in M \tag{4.23}$$

$$\mathbf{C2}: \text{Eqs. (4.5a), (4.5b) and (4.5c)} \tag{4.24}$$

$$\mathbf{C3}: T_{m,n}^{\rho_m}\left(\alpha_{\rho_m}(\tau_i)\right) \leq T_{\rho_m} \quad \forall \mathcal{V}, \forall i \tag{4.25}$$

$$\mathbf{C4}: \text{Eq. (4.21)} \tag{4.26}$$

$$\mathbf{C5}: E_{m,n}^{off}(\alpha_{\rho_m}(\tau_i)) < w_1 E_{c,m}^{\rho_m} \tag{4.27}$$

$$\mathbf{C6}: 0 \leq \gamma_1, \gamma_2, w_1 \leq 1; \ \gamma_1 + \gamma_2 = 1 \tag{4.28}$$

where $\mathcal{A} = \{\alpha_{\rho_m}\}^M$ is the computation offloading matrix, $\mathbf{A}$ is the VU-EN assignment matrix defined previously, and γ_1, and γ_2 are weighting coefficients for balancing latency and energy consumption. The objective function in **P1** includes the overall latency, VU, and the RSU side energy costs including both active and standby modes costs. **C1** stands that each VU can select at most one RSU for the computation offloading. **C2** provides the limits over the number of user requests, processing capacity, and bandwidth resource blocks requested by VUs towards ENs, while **C3** puts a limit on the maximum processing time as one of the task requirements. According to **C4**, for avoiding handover phenomena and related latency, each VU should complete the offloading process before it passes through the selected RSUs coverage. In order to have a valid offloading process, according to **C5**, the weighted energy consumed on VU for processing a complete task should be lower than the total weighted energy required to compute a complete task locally. **C6** stands that the two weighting coefficients (γ_1, γ_2) should be between 0 and 1 with a sum equal to 1. Additionally, the energy coefficient w_1 can take a value between 0 and 1.

4.2.2 MDP Formation

When solving the problem in (4.22), we aim to minimize the overall latency and energy consumed by finding the combination of proper EN and the amount of data to be offloaded by each VU in the MBS service area. In this work, we consider the MDP-based RL approach to solve the problem at hand. The basic elements of the MDP model include the state-space, action-space, reward function, and environment dynamics. However, modeling environment dynamics (i.e., state transition probabilities of MDP states) over a highly uncertain vehicular environment can be a challenging task. Figure 4.2, provides an overview of different elements discussed in the following parts. In the following, we first model several possible vehicular

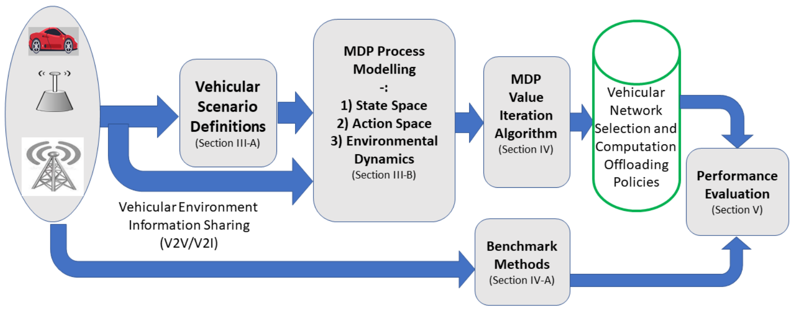

Vehicular Scenario Definitions (Section III-A)
MDP Process Modelling -: 1) State Space 2) Action Space 3) Environmental Dynamics (Section III-B)
MDP Value Iteration Algorithm (Section IV)
Vehicular Network Selection and Computation Offloading Policies
Performance Evaluation (Section V)
Vehicular Environment Information Sharing (V2V/V2I)
Benchmark Methods (Section IV-A)

Figure 4.2: Proposed MDP Model.

scenarios in which a reference VU can find itself over its course. This scenario set can be used to form a proper MDP model aimed at reducing uncertainty over the environment. For avoiding any possible mistakes during the network selection and computation offloading process, each VU scenario needs to be treated separately. After that, we present the main MDP elements (i.e., state-space, action-space, reward function, and environment dynamics) for the considered problem. After a detailed analysis of the state transition probability matrix, we propose generic time-dependent expressions for finding the state transition probability values in different scenarios based on the VUs state and the action performed.

4.2.2.1 VU Scenarios Defintion

Different VU scenarios are formed, based upon VUs physical locations, number of ENs available for offloading, and the number of nearby competing VUs, aiming at creating a more reliable MDP model with reduced uncertainty. VUs can use V2X communication technologies for acquiring useful information about the number of nearby competing VUs and available EN servers. As shown in Fig. 4.1, we have used a grid-based approach for limiting the number of possible scenarios that depend upon the actual VUs position. In the considered grid-based approach, a section of the road is divided into G segments of length l_g, within which each VU is placed, considering its location parameters. Thus, each VU can have associated a specific section number given by $g_m^{id}(\tau_i) = \{1, 2, \ldots, G\}$. Each VU can exploit a different number of ENs for offloading, where $\mathcal{E}_m(\tau_i) = \left\{EN_n | D_{m,n}(\tau_i) > 0, \forall n\right\}$ is the set of available ENs for the mth VU to perform the offloading operation in the interval τ_i. Also, we define $\bar{V}_m(\tau_i) = \sum_{n=1}^{\mathcal{E}_m(\tau_i)} K_n(\tau_i)$ as the number of nearby competing VUs, ranging between 0 to $\mathcal{NV}_{max}$, requesting offloading services from the ENs in the set $\mathcal{E}_m(\tau_i)$.

In the considered multi-user VN, moving VUs can impact each other's network selection and offloading strategies. Each VU should analyze the surrounding environment by finding the competing VUs and their offloading decisions, selected ENs, etc. Since all VUs are supposed simultaneously generate the task requests (i.e., at each ith interval), it is impossible to have such information in advance. In that case, VUs can make offloading decisions by assuming that no other VU is requesting a service leading to a selfish approach. However, this may lead to incorrect node selection and offloading decisions. Another way to tackle this problem is by defining an MDP process that provides a joint solution for all the participating VUs. The presence of a large number of VUs can quickly lead to unbearable complexity and computation requirement. Thus both of these utmost approaches are not suitable for solving the given problem and some sort of assumption is needed for modeling the VUs surrounding environment for avoiding the incorrect offloading strategy/additional complexity. In the following, we consider four strategies supposed by VUs regarding the surrounding environment.

- **Minimum Distance-based VU-EN Assignment:** In this case, the mth VU considers that all the $\bar{V}_m(\tau_i)$ VUs are offloading their data to the nearest ENs based upon their physical locations. Thus, $\forall VU_{m'} \in \bar{V}_m(\tau_i)$:

$$a_{m',n}(\tau_i) = 1 \Longleftrightarrow n = \underset{EN_{n'} \in \mathcal{E}_m(\tau_i)}{\operatorname{argmin}} \{d_{m',n'}(\tau_i)\} \tag{4.29}$$

- **Maximum Sojourn Time-based VU-EN Assignment:** In this case, the mth VU considers that all $\bar{V}_m(\tau_i)$ VUs are offloading their data to the ENs with maximum available sojourn time. Thus, $\forall VU_{m'} \in \bar{V}_m(\tau_i)$:

$$a_{m',n}(\tau_i) = 1 \Longleftrightarrow n = \underset{EN_{n'} \in \mathcal{E}_m(\tau_i)}{\operatorname{argmax}} \{T^{soj}_{m',n'}(\tau_i)\} \tag{4.30}$$

  It should be noted that this approach only considers the assignment towards RSU nodes (since MBS always have high sojourn time). If VUs are not able to find any nearby RSU nodes, they will be assigned to the MBS.

- **Probabilistic VU-EN assignments:** In this approach $\forall VU_{m'} \in \bar{V}_m(\tau_i)$ we select the $EN_{n'} \in \mathcal{E}_m(\tau_i)$ randomly. The probability of m'th VU selecting n'th EN is given by:

$$Pr\{a_{m',n}(\tau_i) = 1\} = \frac{1}{\mathcal{E}_m(\tau_i)} \tag{4.31}$$

- **Position-based VU-EN Assignments:** In this case, each nearby competing VU is allocated to the ENs based on the available distance before it passes through the ENs coverage range and the distance between VU and EN. Thus $\forall VU_{m'} \in \bar{V}_m(\tau_i)$:

$$a_{m',n}(\tau_i) = 1 \Leftrightarrow \frac{D_{m',n}(\tau_i)}{d_{m',n}(\tau_i)} = \max_{EN_{n'} \in \mathcal{E}_m(\tau_i)} \left\{ \frac{D_{m',n'}(\tau_i)}{d_{m',n'}(\tau_i)} \right\}. \tag{4.32}$$

Based upon the above discussion for the mth VU, a vector $\hat{V}_m(\tau_i)$ corresponding to the number of nearby VUs assigned to each $EN_n \in \mathcal{E}_m(\tau_i)$ is formed as:

$$\hat{V}_m(\tau_i) = \left\{V^n_m(\tau_i)\right\}_{1\times\mathcal{E}_m(\tau_i)},$$
$$\text{with } V^n_m(\tau_i) = \sum_{m'=1}^{\bar{V}_m(\tau_i)} a_{m',n}(\tau_i), \quad \forall n \in \mathcal{E}_m(\tau_i) \tag{4.33}$$

where, VU-EN assignment (i.e., $a_{m',n}(\tau_i)$) is based upon any of the four methods presented above.

In the end, for the mth VU, a scenario vector can be defined as $\mathcal{V}_m(\tau_i) = \left\{g_m^{id}(\tau_i), \mathcal{E}_m(\tau_i), \hat{V}_m(\tau_i)\right\}$. The number of possible scenarios is limited by the parameters G, $\mathcal{E}^{max}$, and $\mathcal{NV}_{max}$. Each scenario needs to be treated separately for finding proper EN and offloading amounts. Every vehicular scenario may have an independent optimal policy that needs to be determined through proper analysis. In the end, $\bar{N}$ is the set of all possible VU scenarios.

In the next part, we define the State Space, Action Space, Environment Dynamics or State Transition Probabilities, and Reward Function, as basic elements of an MDP approach for the problem at hand.

4.2.2.2 MDP Elements

The MDP is a stochastic process that evolves over time and is characterized by the state space ($\mathcal{ST}$), action space ($\mathcal{AS}$), reward function (R), and environment dynamics ($\mathcal{P}$). The MDP model can be defined as a tuple $\langle \mathcal{ST}, \mathcal{AS}, R, \mathcal{P} \rangle$.

State-Space ($\mathcal{ST}$) In a multi-user vehicular environment, the available resources for the computation offloading process change continuously over time and are a function of the offloading and network selection decisions taken by individual vehicles. Therefore, we define a discrete state-space set function of resources available for computation offloading. For each scenario ν, the related state-space is a function of the sojourn time, the required latency, VU resources, and the resources of the available RSUs; thus, each state s_ν at time τ_i is defined as:

$$s_\nu(\tau_i) = f(\alpha_{\rho_m}(\tau_i), T^{soj}_{m,n}(\tau_i), B_n, c^R_n,$$
$$f^R_n, \mathcal{L}^n, D_{\rho_m}, D^r_{\rho_m}, \Omega_{\rho_m}, T_{\rho_m}). \quad (4.34)$$

We suppose to limit the multi-dimensional state space to $\bar{N}$ scenarios, hence, $\nu = 1, \dots, \bar{N}$. Moreover, we assume that the environment states observed by each VU during the joint network selection and computation offloading process can be modeled through proper binary functions. If the mth VU is assigned to the nth EN and performs offloading operation with offloading parameter α_{ρ_m}, the environment can be modeled through three proper binary functions, as:

$$F^1_{\rho_m,n}(\tau_i) = \begin{cases} 0 & T^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) \le T^{soj}_{m,n}(\tau_i) \\ 1 & \text{else} \end{cases} \quad (4.35)$$

$$F^2_{\rho_m,n}(\tau_i) = \begin{cases} 0 & T^{\rho_m}_{m,n}\left(\alpha_{\rho_m}(\tau_i)\right) \le T_{\rho_m} \\ 1 & \text{else} \end{cases} \quad (4.36)$$

$$F^3_{\rho_m,n}(\tau_i) = \begin{cases} 0 & E^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) < w_1 E^{\rho_m}_{c,m} \\ 1 & \text{else} \end{cases} \quad (4.37)$$

where $F^1_{\rho_m,n}(\tau_i)$, $F^2_{\rho_m,n}(\tau_i)$ and $F^3_{\rho_m,n}(\tau_i)$ are the binary functions depending upon the sojourn time constraint (4.21), application latency requirement (4.25) and the energy constraint (4.27), respectively, and $F^3_{\rho_m,n}(\tau_i)$ includes both active and standby mode energy costs of RSU nodes. Thus, at τ_i, the state of mth VU in scenario ν is given by,

$$s^{m,n}_\nu(\tau_i) = \left\{F^1_{\rho_m,n}(\tau_i), F^2_{\rho_m,n}(\tau_i), F^3_{\rho_m,n}(\tau_i)\right\} \in S_\nu$$

where, $S_\nu = \mathbb{Z}^3_2$ is the complete state space for the *scenario* ν containing all possible binary combinations of $F^1_{\rho_m,n}(\tau_i)$, $F^2_{\rho_m,n}(\tau_i)$ and $F^3_{\rho_m,n}(\tau_i)$.

Action-Space ($\mathcal{AS}$) The action space defines all the possible actions available during the learning process. If mth VU belongs to the scenario ν, it can explore the available ENs ($\mathcal{E}_m(\tau_i)$), by properly setting a binary vector $\mathbb{EN}_\nu(\tau_i) = \{0,1\}^{\mathcal{E}_m(\tau_i)}$ mapping the RSUs selection among the $\mathcal{E}_m(\tau_i)$ available in the given scenario. At the same time, the offloaded amount can be selected from a discrete set of values given by $\alpha_{\rho_m}(\tau_i) \in \{0, \Lambda, 2\Lambda, \dots, 1\}$ where $0 < \Lambda < 1$ is a step change of offloading amount.

The generic action a_ν for the νth scenario at time τ_i can be defined as $a_\nu(\tau_i) = \left\{\mathbb{EN}_\nu(\tau_i), \alpha_{\rho_m}(\tau_i)\right\}$ where $\mathbb{EN}_\nu(\tau_i)$ is a binary vector with length ν, where 1 in the nth position corresponds to the selected EN. The complete action space for scenario ν is given by $A_\nu = \{a_\nu(\tau_i)\}$.

Once selected, action $a_\nu(\tau_i)$ can change the state of function $F^1(\tau_i)$, $F^2(\tau_i)$ and $F^3(\tau_i)$ with certain probability[5]. Such probabilistic transitions can be defined through:

$$P^{F^1}_{(\bar{i},\bar{j})}(a_\nu(\tau_i)) = Pr\left\{F^1(\tau_{i+\delta}) = \bar{j} \mid F^1(\tau_i) = \bar{i}, a_\nu(\tau_i)\right\} \qquad \bar{i}, \bar{j} \in \{0,1\} \quad (4.38)$$

where $P^{F^1}_{(\bar{i},\bar{j})}(a_\nu(\tau_i))$ is the transition probability of $F^1(\cdot)$ from state $\bar{i}$ to state $\bar{j}$ at τ_i through the action $a_\nu(\tau_i)$. Here, δ is the time step of the MDP process. Similarly for $F^2(\cdot)$ and $F^3(\cdot)$ the transition probability expressions are given by:

$$P^{F^2}_{(\bar{i},\bar{j})}(a_\nu(\tau_i)) = Pr\left\{F^2(\tau_{i+\delta}) = \bar{j} \mid F^2(\tau_i) = \bar{i}, a_\nu(\tau_i)\right\}$$
$$P^{F^3}_{(\bar{i},\bar{j})}(a_\nu(\tau_i)) = Pr\left\{F^3(\tau_{i+\delta}) = \bar{j} \mid F^3(\tau_i) = \bar{i}, a_\nu(\tau_i)\right\}$$

[5]For the simplicity of notations hereafter we omit, $(\rho_m/m,n)$ from $s^{m,n}_\nu(\tau_i)$, $F^1_{\rho_m,n}(\tau_i)$, $F^2_{\rho_m,n}(\tau_i)$, and $F^3_{\rho_m,n}(\tau_i)$.

In general, $F^1(\cdot)$, $F^2(\cdot)$ and $F^3(\cdot)$ can have different probabilistic transitions for any given action $a_v(\tau_i)$. Here, we introduce three transition matrices by considering all the possible transitions of $F^1(\cdot)$, $F^2(\cdot)$, and $F^3(\cdot)$. For $F^1(\cdot)$, the transition matrix $P^{F^1}(a_v(\tau_i))$ is given by,

$$P^{F^1}(a_v(\tau_i)) = \begin{bmatrix} P^{F^1}_{(0,0)}(a_v(\tau_i)) & P^{F^1}_{(0,1)}(a_v(\tau_i)) \\ P^{F^1}_{(1,0)}(a_v(\tau_i)) & P^{F^1}_{(1,1)}(a_v(\tau_i)) \end{bmatrix}, \forall a_v(\tau_i) \tag{4.39}$$

with, $P^{F^1}_{(0,0)}(a_v(\tau_i))+P^{F^1}_{(0,1)}(a_v(\tau_i)) = 1$ and $P^{F^1}_{(1,0)}(a_v(\tau_i))+P^{F^1}_{(1,1)}(a_v(\tau_i)) = 1$. Similarly, for F^2, the transition matrix $P^{F^2}(a_v(\tau_i))$ is given by,

$$P^{F^2}(a_v(\tau_i)) = \begin{bmatrix} P^{F^2}_{(0,0)}(a_v(\tau_i)) & P^{F^2}_{(0,1)}(a_v(\tau_i)) \\ P^{F^2}_{(1,0)}(a_v(\tau_i)) & P^{F^2}_{(1,1)}(a_v(\tau_i)) \end{bmatrix}, \forall a_v(\tau_i) \tag{4.40}$$

with, $P^{F^2}_{(0,0)}(a_v(\tau_i)) + P^{F^2}_{(0,1)}(a_v(\tau_i)) = 1$ and $P^{F^2}_{(1,0)}(a_v(\tau_i)) + P^{F^2}_{(1,1)}(a_v(\tau_i)) = 1$. Also, for the case of $F^3(\cdot)$, the transition matrix $P^{F^3}(a_v(\tau_i))$ is given by,

$$P^{F^3}(a_v(\tau_i)) = \begin{bmatrix} P^{F^3}_{(0,0)}(a_v(\tau_i)) & P^{F^3}_{(0,1)}(a_v(\tau_i)) \\ P^{F^3}_{(1,0)}(a_v(\tau_i)) & P^{F^3}_{(1,1)}(a_v(\tau_i)) \end{bmatrix}, \forall a_v(\tau_i) \tag{4.41}$$

with $P^{F^3}_{(0,0)}(a_v(\tau_i)) + P^{F^3}_{(0,1)}(a_v(\tau_i)) = 1$ and $P^{F^3}_{(1,0)}(a_v(\tau_i)) + P^{F^3}_{(1,1)}(a_v(\tau_i)) = 1$.

Reward Function $(R(s_v(\tau_i), a_v(\tau_i))$ The reward function $(R(s_v(\tau_i), a_v(\tau_i))$ is defined as the joint objective function of time and energy consumed for complete task processing (4.22). At the ith interval, if the mth VU is in state $s_v(\tau_i)$, and decides to take an action $a_v(\tau_i)$ by selecting the nth RSU and $\alpha_{\rho_m}(\tau_i)$ as an offloading amount, the instant reward received by it is given by,

$$R(s_v(\tau_i), a_v(\tau_i)) = \left[\gamma_1 T^{\rho_m}_{m,n}\left(\alpha_{\rho_m}(\tau_i)\right) + \gamma_2 E^{\rho_m}_{m,n}(\alpha_{\rho_m}(\tau_i))\right] \tag{4.42}$$

State Transition Matrix ($\mathcal{P}$) For the MDP process, the state transition matrix characterizes the environment dynamics through the probabilistic transitions between the present states to the next state. Thus, for scenario ν, the state transition probability at τ_i is given by

$$\begin{aligned} Pr\left\{s_\nu(\tau_{i+\delta})|s_\nu(\tau_i), a_\nu(\tau_i)\right\} = \\ Pr\Big\{ \left\{F^1(\tau_{i+\delta}), F^2(\tau_{i+\delta}), F^3(\tau_{i+\delta})\right\} \mid \\ \Big(\left\{F^1(\tau_i), F^2(\tau_i), F^3(\tau_i)\right\}, a_\nu(\tau_i)\Big)\Big\} \end{aligned} \tag{4.43}$$

where $\left\{F^1(\tau_i), F^2(\tau_i), F^3(\tau_i)\right\}$ is the current state of VU at τ_i that takes action $a_\nu(\tau_i)$.

We assume that the state transition probability expression based on $F^1(\tau_i)$, $F^2(\tau_i)$, and $F^3(\tau_i)$ can be considered as independent events, hence (4.43) can be rewritten as:

$$\begin{aligned} Pr\{s_\nu(\tau_{i+\delta}) \mid s_\nu(\tau_i), a_\nu(\tau_i)\} = \\ Pr\left\{F^1(\tau_{i+\delta}) \mid F^1(\tau_i), a_\nu(\tau_i)\right\} \\ \cdot Pr\left\{F^2(\tau_{i+\delta}) \mid F^2(\tau_i), a_\nu(\tau_i)\right\} \\ \cdot Pr\left\{F^3(\tau_{i+\delta}) \mid F^3(\tau_i), a_\nu(\tau_i)\right\} \end{aligned} \tag{4.44}$$

where each term is based upon (4.39)-(4.41). For example if $F^1(\tau_i) = 0$ and $F^1(\tau_{i+\delta}) = 1$, then $Pr\left\{F^1(\tau_{i+\delta}) \mid F^1(\tau_i), a_\nu(\tau_i)\right\} = P^{F^1}_{(0,1)}(a_\nu(\tau_i))$. Detailed analysis of this probability values is given below.

In (4.39) the four probabilistic transitions for the binary-valued function $F^1(\cdot)$ are set. As shown in (4.35), $F^1(\cdot)$ becomes 1, if the computation offloading process fails to follow the sojourn time constraint; on the other hand, it becomes 0, if the process follows the constraint. Two probability values $P^{F^1}_{(0,1)}(a_\nu(\tau_i))$ and $P^{F^1}_{(1,0)}(a_\nu(\tau_i))$

model the behavior of $F^1(\cdot)$ based upon the action taken. These transitions can depend upon several factors, including the number of VUs assigned to the selected ENs, the available sojourn time value, which differs for different ENs, the offloading amount, etc. Modeling the exact nature of these transitions can be hard; we resort to exponential distribution functions for modeling the behavior of $F^1(\cdot)$.

In case the mth VU in scenario v selects the nth EN, through the action $a_v(\tau_i)$ we define:

$$P^{F^1}_{(0,1)}(a_v(\tau_i)) = \begin{cases} 0 & \text{if } \alpha_{\rho_m}(\tau_i) = 0 \\ 1 - \exp(-\lambda_1(\tau_i)) & \text{else} \end{cases} \tag{4.45}$$

where $\lambda_1(\tau_i) = K_{11} \cdot \alpha_{\rho_m}(\tau_i) + K_{12} \cdot V^n_m(\tau_i) + K_{13}/T^{soj}_{m,n}(\tau_i)$ is a parameter modeling the slope of the exponential function and is determined from the action $a_v(\tau_i)$. According to (4.45), if the selected action is characterized by $\alpha_{\rho_m}(\tau_i) = 0$, the possibility of the failure of offloading constraint becomes zero. The value of $\lambda_1(\tau_i)$ depends upon several factors. In particular, if the action performed by the mth VU is characterized by high $\alpha_{\rho_m}(\tau_i)$, if VU selects the EN having a higher number of VUs requesting services (i.e., large $V^n_m(\tau_i)$), or VU-EN pair is characterized by the low sojourn time, the value of $\lambda_1(\tau_i)$ can increase. As a result, the probability that $F^1(\cdot)$ changes its state from 0 to 1 (i.e., failure of offloading constraint) becomes high, which can be emphasized in (4.45). K_{11}, K_{12} and K_{13} are weighting coefficients assigning proper weights to each of these parameters.

$P^{F^1}_{(1,0)}(a_v(\tau_i))$ models the case where, with the selected action $a_v(\tau_i)$, the VU is able to satisfy the offloading time constraint where:

$$P^{F^1}_{(1,0)}(a_v(\tau_i)) = \begin{cases} 1 & \text{if } \alpha_{\rho_m}(\tau_i) = 0 \\ \exp(-\lambda_2(\tau_i)) & \text{else} \end{cases} \tag{4.46}$$

where $\lambda_2(\tau_i) = K_{21} \cdot \alpha_{\rho_m}(\tau_i) + K_{22} \cdot V_m^n(\tau_i) + K_{23}/T_{m,n}^{soj}(\tau_i)$ is a parameter modeling the slope of the exponential function and is determined from the action $a_v(\tau_i)$. This corresponds to say that, if the selected action is characterized by $\alpha_{\rho_m}(\tau_i) = 0$, VUs offloading time becomes zero and, as a result, it will satisfy the sojourn time constraint. Also from the expression of $\lambda_2(\tau_i)$ and (4.46), it can be seen that, when increasing $\alpha_{\rho_m}(\tau_i)$ and $V_m^n(\tau_i)$, the probability that the mth VU respects the sojourn time constraint is reduced. The reduced value of $T_{m,n}^{soj}(\tau_i)$ between the VU-EN pair can also reduce the chances that VU respects the sojourn time constraint. Here, K_{21}, K_{22} and K_{23} are weighting coefficients.

The second function $F^2(\cdot)$ models the VUs behavior with respect to the task latency constraint, where each VU needs to perform the task processing within the task latency requirements. In this case, $P_{(0,1)}^{F^2}(a_v(\tau_i))$ defines the probability that VU fails to satisfy the task latency constraint for a selected action $a_v(\tau_i)$ and is given by:

$$P_{(0,1)}^{F^2}(a_v(\tau_i)) = \begin{cases} 1 & \text{if } \alpha_{\rho_m}(\tau_i) = 0 \\ \exp(-\lambda_3(\tau_i)) & \text{else} \end{cases} \tag{4.47}$$

where $\lambda_3(\tau_i) = K_{31} \cdot T_{\rho_m} + \frac{1}{K_{32}+\alpha_{\rho_m}(\tau_i)(1-\alpha_{\rho_m}(\tau_i))} + \frac{K_{33}}{V_m^n(\tau_i)}$ is a parameter modeling the slope of the exponential function and is determined from the action $a_v(\tau_i)$. This corresponds to say that, with its limited resources, if a VU performs the task processing by itself without offloading any data towards ENs, it always fails to satisfy the task latency requirements. In addition, if we have a strict task latency requirement (T_{ρ_m}), and the selected EN has already a large number of VUs ($V_m^n(\tau_i)$) requesting services, this results in increasing the failure probability of the task latency constraint.

If any VU offloads a very small percentage of data towards an EN, the local computation time required for processing the remaining task can be high. On the other hand, if any VU offloads a larger amount of data toward an EN, it is possible to have a higher offloading time mainly because of unreliable channel conditions, limited EN resources, and other competing VUs. The behavior of $P^{F^2}_{(0,1)}(a_v(\tau_i))$ concerning the offloading parameter $\alpha_{\rho_m}(\tau_i)$ is modeled as a square function for accommodating these facts. In the end, K_{31} and K_{33} define the weights assigned to latency and competing vehicles parameters, while K_{32} avoid having infinite in the second term.

$P^{F^2}_{(1,0)}(a_v(\tau_i))$ models the VUs' chances of satisfying the task latency requirements and is defined as,

$$P^{F^2}_{(1,0)}(a_v(\tau_i)) = \begin{cases} 0 & \text{if } \alpha_{\rho_m}(\tau_i) = 0 \\ 1 - \exp(-\lambda_4(\tau_i)) & \text{else} \end{cases} \tag{4.48}$$

where $\lambda_4(\tau_i) = K_{41} \cdot T_{\rho_m} + \frac{1}{K_{42}+\alpha_{\rho_m}(\tau_i)(1-\alpha_{\rho_m}(\tau_i))} + \frac{K_{43}}{V^n_m(\tau_i)}$ is a parameter modeling the slope of the exponential function and is determined from the action $a_v(\tau_i)$. In case VU does not offload any data, it is not able to satisfy the task latency requirements. On the other hand, the behavior of $P^{F^2}_{(1,0)}(a_v(\tau_i))$ will be based upon the offloading parameter, number of competing VUs, and the task latency requirements. K_{41} and K_{43} are weighting coefficients, while K_{42} avoid to have infinite in the second term.

The third function, $F^3(\cdot)$, models the VU behavior in terms of energy constraint. If the overall offloading process energy becomes higher than the energy required to compute the complete task locally, the offloading process becomes inefficient. $P^{F^3}_{(0,1)}(a_v(\tau_i))$ gives the probability that the VU fails to satisfy the energy constraint for a selected action $a_v(\tau_i)$ and is defined as,

$$P^{F^3}_{(0,1)}(a_v(\tau_i)) = \begin{cases} 0 & \text{if } \alpha_{\rho_m}(\tau_i) = 0 \\ 1 - \exp(-\lambda_5(\tau_i)) & \text{else} \end{cases} \tag{4.49}$$

where, $\lambda_5(\tau_i) = K_{51} \cdot \alpha_{\rho_m}(\tau_i) + K_{52} V_m^n(\tau_i)$ is a parameter modeling the slope of the exponential function and is determined from the action $a_v(\tau_i)$. If the mth VU offloads a large amount of data towards the EN with more $V_m^n(\tau_i)$, with selected action $a_v(\tau_i)$, there is a high chance that the offloading process energy becomes higher than the local computation energy. However, if VU does not offload any data towards EN, it always follows the energy constraint. Here, K_{51} and K_{52} are weighting coefficients.

$P^{F^3}_{(1,0)}(a_v(\tau_i))$ models the chances that VU is satisfying the energy constraint based upon the selected action $a_v(\tau_i)$:

$$P^{F^3}_{(1,0)}(a_v(\tau_i)) = \begin{cases} 1 & \text{if } \alpha_{\rho_m}(\tau_i) = 0 \\ \exp(-\lambda_6(\tau_i)) & \text{else} \end{cases} \tag{4.50}$$

where $\lambda_6(\tau_i) = K_{61} \cdot \alpha_{\rho_m}(\tau_i) + K_{62} V_m^n(\tau_i)$ is a parameter modeling the slope of the exponential function and is determined from the action $a_v(\tau_i)$. The chances that VU satisfies the energy constraint reduce with the increasing of $\alpha_{\rho_m}(\tau_i)$ and $V_m^n(\tau_i)$. K_{61} and K_{62} are weighting coefficients.

By using (4.45)-(4.50), the transition probability matrices for $F^1(\cdot)$, $F^2(\cdot)$, and $F^3(\cdot)$ can be determined. In the following Section, we define a value iteration algorithm for solving the MDP.

4.2.3 MDP-Based Joint Network Selection and Computation Offloading

In the previous section, the elements of the MDP model are presented. By solving the proposed MDP model, VUs can find a proper EN and the offloading amount able to minimize the overall latency and the energy consumed during the task processing operations. The solutions set can be defined as a policy function $\pi_v = \{\pi_v(s_v(\tau_i+\delta)), \forall\delta\}$ that maps every state $s_v \in \mathcal{ST}$ to action $a_v \in \mathcal{AS}$. Selecting different actions can result in different policy functions, where the aim is to find an optimal policy that corresponds to the minimum delay and energy cost during vehicular task processing. For every policy π_v, a value function $V_{\pi_v}(s_v(\tau_i))$, corresponding to a state $s_v(\tau_i)$ can be defined for analyzing its performance. In general, $V_{\pi_v}(s_v(\tau_i))$ corresponds to an expected value of a discounted sum of total reward received by following the policy π_v from state $s_v(\tau_i)$, and can be defined as:

$$V_{\pi_v}(s_v(\tau_i)) = \mathbb{E}\left\{\sum_{\delta=0}^{\Delta} \gamma^{\delta} R\left(s_v\left(\tau_i+\delta\right), \pi_v\left(s_v(\tau_i+\delta)\right)\right)\right\}$$

where $\gamma \in [0,1]$ is the discount factor, $R\left(s_v\left(\tau_i+\delta\right), \pi_v\left(s_v\left(\tau_i+\delta\right)\right)\right)$ is the immediate reward received for following the policy π_v at time $\tau_i+\delta$ from the state $s_v(\tau_i+\delta)$, Δ is the maximum number of steps considered during the MDP evaluation, i.e., episode length, and $\mathbb{E}\{\cdot\}$ corresponds to the expected value. Thus, the value function analyzes the particular policy function by assigning a numeric value to each state, and can be utilized to compare the performance of the different policies. In the end, the following optimization problem can be formulated in order to be able to find the best possible policy function associated with state $s_v(\tau_i)$:

$$V(s_v(\tau_i)) = \min_{\pi_v \in \Pi_v} V_{\pi_v}(s_v(\tau_i)) \tag{4.51}$$

where, Π_v corresponds to the set of policy functions that can be explored.

As shown by many works (e.g., [88, 89]), the problem defined in (4.51), can converge into a Bellman optimality equation given by:

$$V(s_\nu(\tau_i)) = \min_{a_\nu(\tau_i)\in A_\nu(\tau_i)} \Bigg\{ R(s_\nu(\tau_i), a_\nu(\tau_i)) + \gamma \sum_{s_\nu(\tau_i+\delta)\in \mathcal{ST}} Pr\left\{s_\nu(\tau_i+\delta) \mid s_\nu(\tau_i), a_\nu(\tau_i)\right\} V(s_\nu(\tau_i+\delta)) \Bigg\} \tag{4.52}$$

Different approaches can be used to solve the problem in (4.52); however, the value iteration approach is widely known for its fast convergence and easy implementation [90]. Therefore, below we present a value iteration approach aimed at solving the MDP designed in the previous section for finding an optimal policy that corresponds to the minimization of a task processing time and energy during offloading process over VNs.

The value iteration method allows finding an optimal policy and value function for the MDP models. The Algorithm 2 describes the steps involved during the value iteration process. For every scenario ν, the process begins by initializing the values of each state to ∞ and iteration count (it) to 0 (Line 2). For each state-action pair, the state value is determined by using (4.53) (Line 5). In the end state value and a corresponding optimal policy $(\pi^*_\nu(s_\nu(\tau_i)))$ associated with state s_ν is determined by using (4.54) and (4.55) (Lines 7-8). The iterative process continues till the change in all states values becomes less than the predefined convergence parameter ϵ (Lines 10-14). In the end, the algorithm returns the set of optimal policy functions $\left\{\pi^*_\nu\right\}$ associated with all possible scenarios in which VUs can find themselves over the road (Line 16).

Algorithm 2 MDP Value Iteration

Input: $\epsilon, \gamma, \bar{N}, S_v, A_v, Pr$

Output: $\{\pi_v^*\}$

```
1: for $v \in \bar{N}$ do
2:     Initialize $it = 0$, $V^0(s_v(\tau_i)) = \infty, \forall s_v(\tau_i)$
3:     for $s_v(\tau_i) \in S_v$ do
4:         for $a_v(\tau_i) \in A_v$ do
5:
```

$$V^{it+1}(s_v(\tau_i), a_v(\tau_i)) \leftarrow R(s_v(\tau_i), a_v(\tau_i)) + \gamma \sum_{s_v(\tau_i+\delta) \in S_v} Pr(s_v(\tau_i + \delta) \mid s_v(\tau_i), a_v(\tau_i)) v^{it}(s_v(\tau_i + \delta)) \quad (4.53)$$

```
6:         end for
7:
```

$$V^{it+1}(s_v(\tau_i)) = \min_{a_v(\tau_i)} V^{it+1}(s_v(\tau_i), a_v(\tau_i)) \quad (4.54)$$

```
8:
```

$$\pi_v^*(s_v(\tau_i)) = \operatorname*{argmin}_{a_v(\tau_i)} V^{it+1}(s_v(\tau_i), a_v(\tau_i)) \quad (4.55)$$

```
9:     end for
10:    if any $|v^{it+1}(s_v(\tau_i) - v^{it}(s_v(\tau_i)| > \epsilon$ then
11:        $it = it + 1$
12:    else
13:        return $\pi_v^* = \{\pi_v^*(s_v(\tau_i))\}$
14:    end if
15: end for
16: return $\{\pi_v^*\}$
```

The time complexity of the traditional value iteration process can be estimated to be equal to $O(\Delta|\mathcal{ST}| \cdot |\mathcal{AS}|)$ with Δ being the maximum number of time steps considered, $|\mathcal{ST}|$ state space dimension, and $|\mathcal{AS}|$ representing the action space. With the involvement of $\bar{N}$ scenarios, the time complexity expression becomes $O(\bar{N} \cdot \Delta|\mathcal{ST}| \cdot |\mathcal{AS}|)$. The considered scenario-based modeling can reduce the state and action space dimensions significantly by limiting the number of VUs per scenario compared to the one-shot approaches where all VUs are considered altogether. Especially for the case of VNs, such an approach can be beneficial given the importance of VUs' local environments in the decision-making process (i.e., nearby VUs can influence the VUs' decision-making compared with the other VUs that are located far away from it). Additionally, time-dependent state transition probabilities can reduce the overall uncertainty in the MDP process. It should be noticed that $\bar{N}$, i.e., the considered number of VUs scenarios, can impact the performance of the MDP process. On one side, a smaller $\bar{N}$, corresponding to a limited set of parameters, can impact the MDP model performance due to additional uncertainties. On the other side, with a bigger $\bar{N}$, the computational complexity can be higher with improved performance.

4.2.4 Benchmark Approaches

For comparing the proposed MDP model performance, the following benchmark methods are considered:

- **Minimum Distance VU-EN Assignment Based Approach (MDA):** In this approach, VUs are always assigned to the EN, which is at a minimum distance from them. Also, VUs prefer to offload a complete task towards selected EN. Thus, $\forall m$,

$$a_{m,n}(\tau_i) = 1 \Longleftrightarrow n = \underset{n \in N}{\operatorname{argmin}}\{d_{m,n}(\tau_i)\} \tag{4.56}$$

Though this approach can potentially reduce the overall task communication delay and energy, high handovers requirements and questionable energy performance can reduce the offloading performance.

- **Maximum Sojourn Time Based VU-RSU Assignment Approach (MSA):** In this method, VUs prefer to offload their task towards RSUs having the highest sojourn time, hence:

$$a_{m,n}(\tau_i) = 1 \Longleftrightarrow n = \underset{n \in N}{\operatorname{argmax}}\{D_{m,n}(\tau_i)\} \tag{4.57}$$

This approach can reduce the number of handover requirements however, the computation/communication delay and energy performance might not be optimal.

- **MDP-based Network Selection with Static Offloading Policy (MDP-NS):** To show the impact of a joint network selection and offloading optimization, here we consider an MDP-based network selection decision optimization with static offloading process. In particular, the MDP process (i.e., action space) is adapted to the network selection only while considering a static offloading policy with $\alpha_{\rho_m}(\tau_i) = 0.5, \forall i, m$.

- **MDP-based Offloading with Static Network Selection Policy (MDP-Off):** In this case, the offloading decisions are optimized, while a static network selection is considered. In particular, VUs are considered to select the nearest EN while offloading with an optimal policy generated through the MDP process of Algorithm 2.

In the following, MDP-MD, MDP-PA, MDP-SA, and MDP-PsA stand for the MDP with minimum distance assignments, probabilistic assignments, sojourn time based assignments and the position-based assignments of nearby VUs, respectively.

4.2.5 Numerical Results

The proposed MDP model and corresponding value iteration algorithm is evaluated over a Python-based simulator, using ML-related libraries such as NumPy, Pandas, and Matplotlib. The main simulation parameters are listed in Table 4.1. In this work, we have considered that 80 RSUs with h_n^R=3 m are located alongside the road network in the MBS coverage area. The number of VUs is between 200 to 1800 with p_a=0.2. Each VU travels with a variable speed based upon the intelligent mobility model, with parameters $\vec{v}_{max}$=15 m/s, s_{min}=2 m, a_{max}=0.7 m/s^2, b_{max}=1.5 m/s^2, t_r=2 s. The background noise power $\sigma = -110$ dBm is considered.

Also, each RSU can serve up to $K^{\max}$=12 VUs. Additionally, the communication channel parameters are β_0=−25 dB, and θ=2.5. The RSU switching parameters include switching time $T_{sw,n}$=25 ms, and switching power $P_{sw,n}$=0.2 W. Also, when the nth EN is operating in the standby mode, the standby power is $P_{sd,n}$=0.42 W. The power consumed for the basic circuit operations is $P_{0,n}$=0.5 W. The VUs scenarios are based upon l_g=3.3 m, $\mathcal{E}^{max}$=4, and $\mathcal{NV}_{max}$=36. Other MDP parameters include the set of weighing coefficients given by, $[K_{11}, K_{12}, K_{13}, K_{21}, K_{22}, K_{23}] = [0.5, 0.07, 0.4, 0.5, 0.07, 0.4]$, $[K_{31}, K_{32}, K_{33}, K_{41}, K_{42}, K_{43}] = [0.08, 0.6, 0.5, 0.08, 0.6, 0.5]$,

Table 4.1: Simulation parameters

MBS Coverage (d^M)	500 m
RSU Coverage (d_n)	25 m
Task Size (D_{ρ_m})	5 MB
Task Results ($D^r_{\rho_m}$)	($D_{\rho_m}/5$) MB
Required Task Latency (T_{ρ_m})	2 s
VU Computation Cap. ($c_m \cdot f_m$)	18 GFLOPS
RSU Computation Cap. ($c^R_n \cdot f^R_n$)	45 GFLOPS
MBS Computation Cap. ($c^{\bar{M}} \cdot f^{\bar{M}}$)	150 GFLOPS
RSU Height (h^R_n)	4 m
MBS Height (h^M)	10 m
CPU Cores ($\mathcal{L}_n$)	4
Task Proc. Requirements Ω_{ρ_m}	8 GFLOPS per MB
RSU Bandwidth (B^R_n)	80 MHz
BS Bandwidth (B^M)	1 GHz
VU Energy ($P_{c,m}, Pt_m, Pr_m$)	(0.9, 1.3, 1.1) W
RSU Energy ($P_{c,n}, Pt_n, Pr_n$)	(1.2, 1.3, 1.1) W
Weighting Coefficients (w_1, γ_1, γ_2)	(0.6, 0.5, 0.5)

and $[K_{51}, K_{52}, K_{61}, K_{62}] = [0.5, 0.07, 0.5, 0.07]$. During the value iteration process γ=0.9, ϵ=0.01, Λ=0.1 and episode length Δ=100 are used.

Avg. Latency and Energy Cost with Varying VUs

In Fig. 4.3, we present the average cost value in terms of the total latency and energy requirements of VUs task processing. By varying the number of VUs, we obtain the performance of different MDP schemes defined before and analyze their performance by comparing the results with the benchmark methods. It can be seen that proposed MDP schemes perform better compared with the benchmark

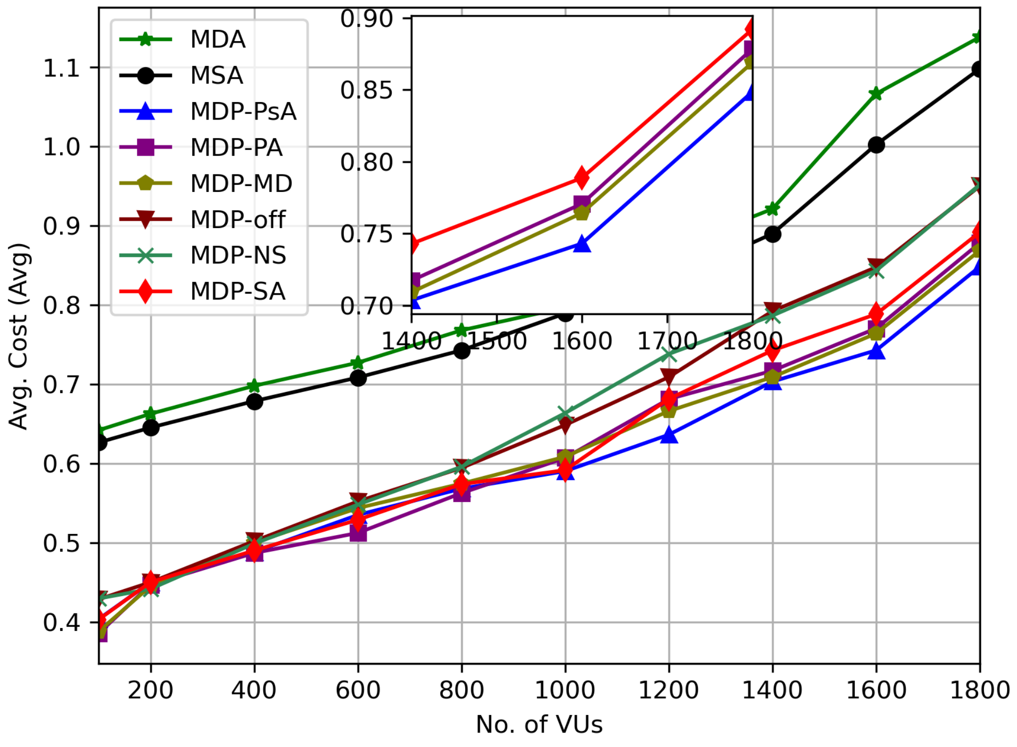


FIGURE 4.3: Cost Function

approaches. By analyzing the surrounding environment, different MDP schemes are able to find proper EN and the amount to be offloaded. In particular, with a high number of VUs, the MDP-PsA approach having a better knowledge of the surrounding environments in terms of various distance measures (i.e., the distance between VU and ENs and the distance before it passes through the EN coverage range), performs better than the other schemes. The superiority of the MDP-PsA approach can be visualized through the zoomed version of the plot. The two benchmark MDP methods (MDP-Off and MDP-NS) have worse performance compared to the joint optimization-based approaches, mainly due to the static policies. This highlights the importance of simultaneously selecting the proper ENs and offloading the proper amount.

Number of RSU handover required during computation offloading

If VU fails to perform the offloading operation (which includes the transmission of VUs data towards selected EN, EN processing, and receiving back the results from EN), before going out from the coverage of the selected EN, an additional handover process/cost is required. In Fig. 4.4, we present such handover requirements posed by a different set of VUs in terms of the average number of VUs which fail to complete the offloading operation within time limits. It can be verified from this figure that the proposed MDP schemes (in particular MDP-PsA) are performing better compared to the other benchmark methods in terms of a reduction in the overall handover requirements. Thus by avoiding the number of handover requirements, MDP schemes can reduce the service provisioning costs over vehicular environments. The benchmark MDP methods, in particular MDP-Off, suffer from higher handover requests due to the imperfect offloading decisions compared to the other MDP methods.

Number of service time constraint failures

In general, VUs application latency requirements need to be respected during task processing operations, failure of which can reduce the overall QoS. In Fig. 4.5, we provide the percentage of VUs that fails to satisfy the application latency requirement constraint in (4.25). The proposed MDP approaches are able to reduce such failures effectively and can be vital for enabling latency-critical services over VN. Similar to the previous cases, the MDP-PsA approach outperforms the other MDP schemes and can be seen through the zoomed version of the plot. The MDP-NS and MDP-Off methods induce higher latency costs, and their performance suffers with more service latency failures than the other MDP approaches.

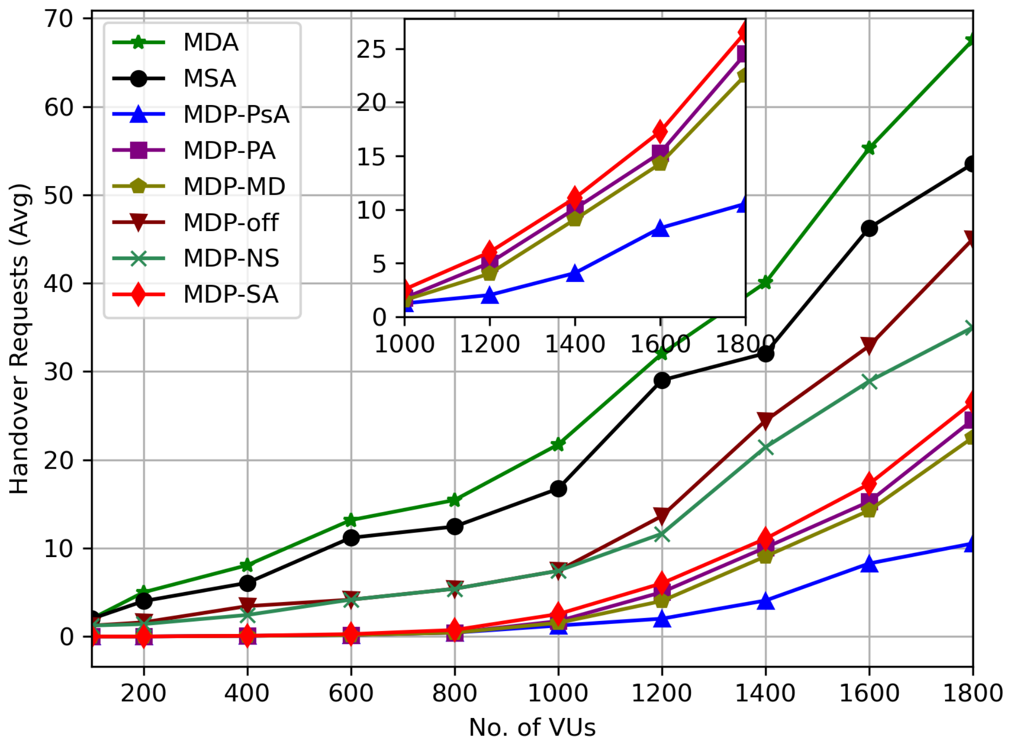


FIGURE 4.4: Percentage of VUs with Handover Requirements

Task Completion Latency

To have a better understanding of overall latency requirements, in Fig. 4.6, we present the performance of different schemes in terms of average latency requirements during the task processing operations. This figure shows the overall reduction of latency cost for VUs task processing operations. Through a proper understanding of the nearby environment parameters (e.g., competing VUs, available RSUs, mobility characteristics), the MDP schemes, in particular MDP-PsA approach, can determine the proper EN and the offloading amount for having better performance. Optimizing only the network selection or offloading decisions through the MDP-NS and MDP-Off methods cannot guarantee optimal performances and suffers from higher latency requirements.

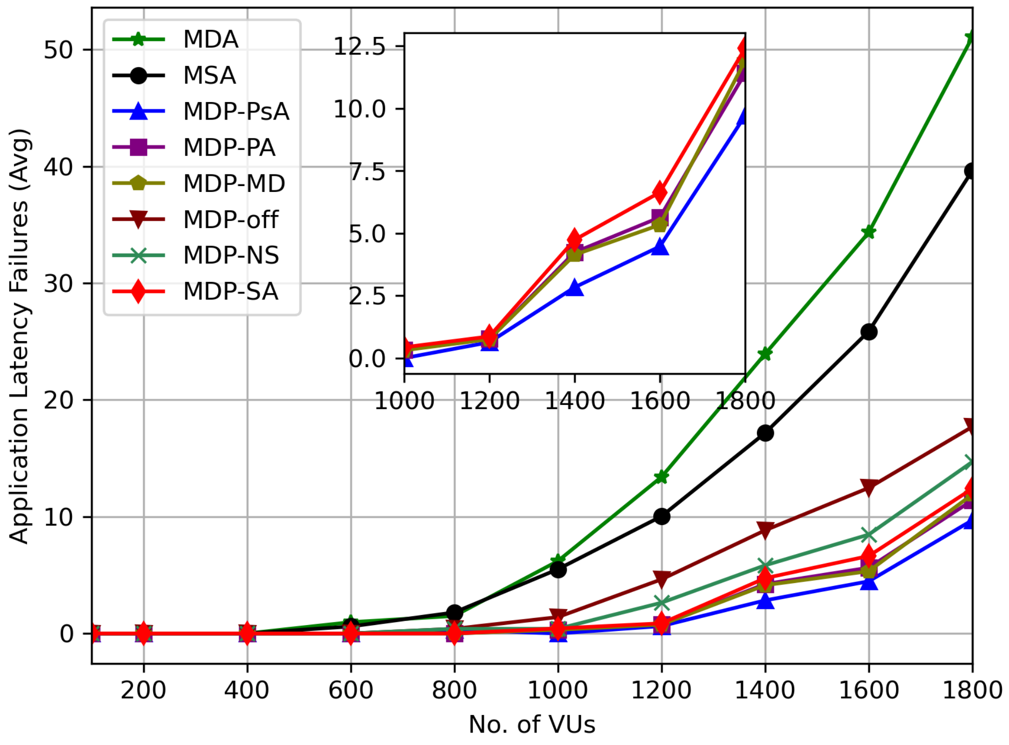


FIGURE 4.5: Percentage of VUs with service time constraint violation.

Average Energy Consumption

In the following Figs. 4.7 and 4.8, we present the performance of different schemes in terms of average energy requirements. Fig. 4.7 presents the average amount of energy cost over VUs, which includes the local computation, data transmission, and reception costs. The benchmark approaches do not perform any local computation, due to which they have slightly better performance in terms of VUs energy consumption. However, as shown in Fig. 4.8, both benchmark methods add large energy costs over ENs. On the other hand, with proper EN selection, and proper offloading decisions, all MDP schemes are having better energy performances over EN. Also, as shown in Fig. 4.3, the overall performance of the MDP process in terms of joint latency and energy cost is better compared with the benchmark approaches.

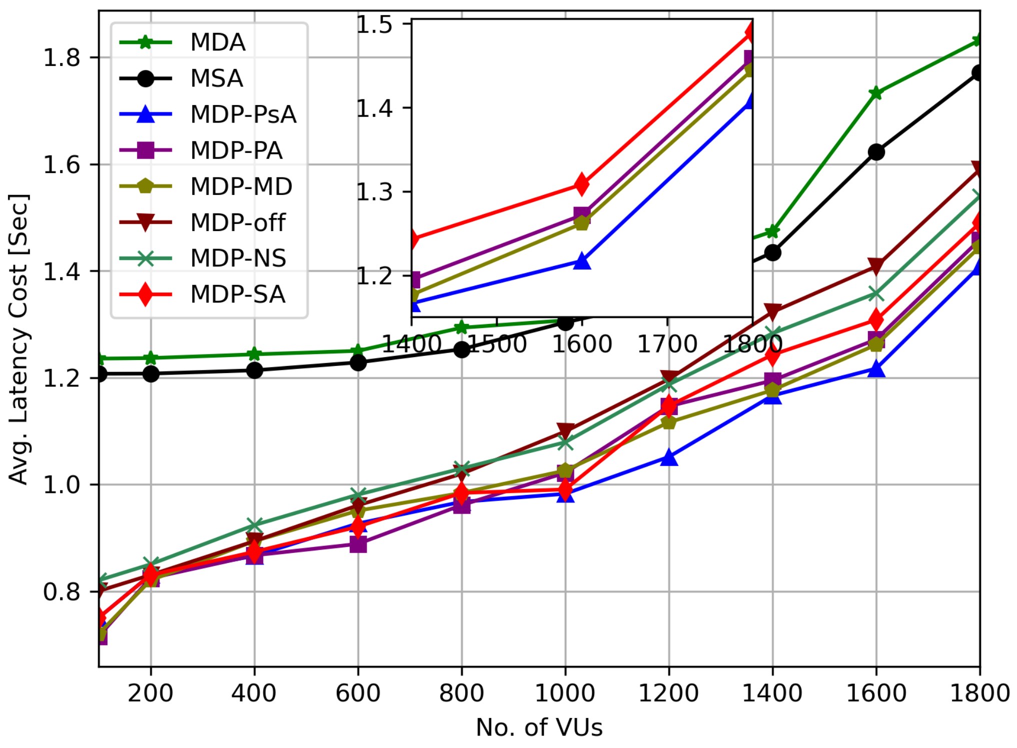


FIGURE 4.6: Avg. Latency Cost

The joint energy performance of the MDP-Off and MDP-NS methods suffers from imperfect decision makings and impacts the overall costs shown in Fig. 4.3.

Average Number of Active ENs

In Fig. 4.9, we have presented the average number of active ENs for varying numbers of VUs. In the beginning with a limited number of VU density, only a limited number of ENs are active. With most of the ENs being inactive, the overall energy cost can be reduced compared with the traditional approaches with all ENs being active. As VU density increases, the active ENs increase for satisfying all VUs service requests. This allows reducing the total number of service failures. With this and previous results, it can be validated that the proposed methods are able to adapt the ENs

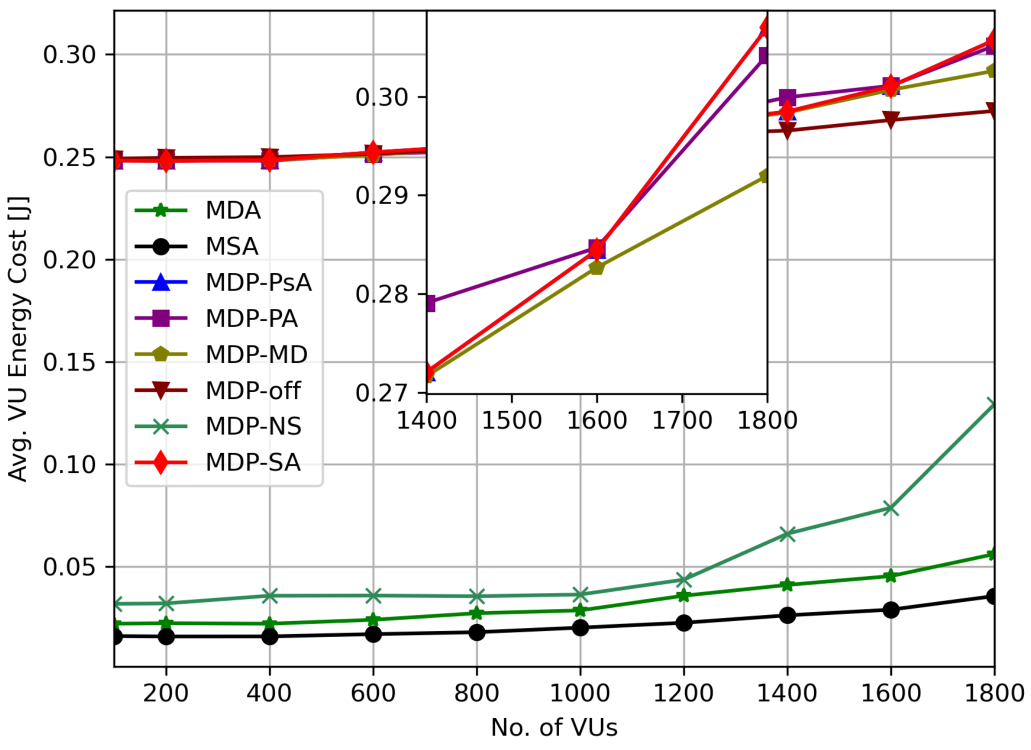


FIGURE 4.7: Avg. VU Energy Cost

energy resources according to the VUs demands limiting the EN energy costs along with the potential service failures.

4.2.6 Conclusion

In this work, we considered the joint optimization of network selection and task offloading through a proper minimization of delay and energy for a VEC offloading system. For solving such a complex problem over a highly uncertain vehicular environment, we have proposed a MDP approach by analyzing different vehicular scenarios. The proposed MDP model considers the changing vehicular environment while making the decisions of EN selection and offloading portion. A value iteration-based method is used for solving the proposed MDP model by finding the optimal

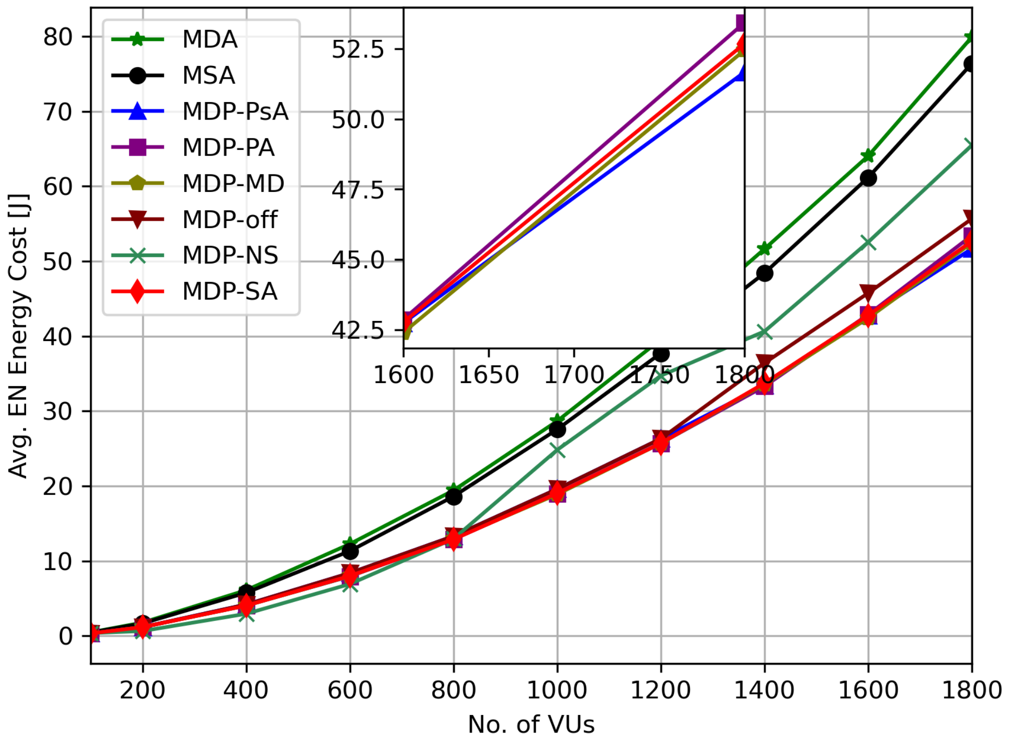


FIGURE 4.8: Avg. EN Energy Cost

policy to be followed by each VU in the different scenarios. The simulation results show the superiority of the proposed scheme over various benchmark methods. One of the most prominent contributions is that of having considered the joint network selection and computation offloading problem while jointly minimizing latency and energy costs with additional energy-saving mechanisms at the edge infrastructure. Such studies were not present in the current literature and thus can motivate future readers to investigate it further. However, with additional granularities and joint decision-making processes, the problem becomes extremely complex to be solved through the traditional approaches. For this, we have proposed a novel MDP model with time-dependent state transition probabilities reducing the overall instability. However, since the MDP approach could become very complex in the case of a large parameter set, in this work, we have exploited the local vehicular communication

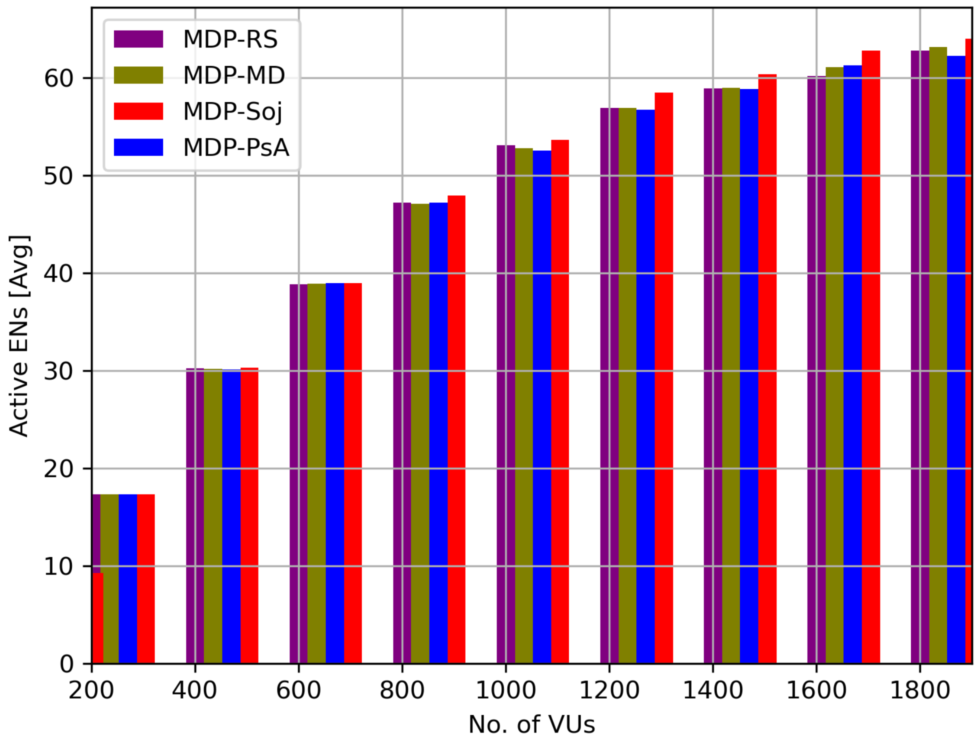


FIGURE 4.9: Avg. Number of Active ENs

modes in the MDP process to improve the overall performance. This can motivate future readers to investigate the proposed solution methods in these directions.

4.3 Joint Network Selection and Offloading: Multi-service Case

Differently from other works, we aim at jointly optimizing the network selection and the computation offloading to minimize network-wide latency and energy consumption with multiple services. To this end, the problem is modeled as a cost function and solved through an RL approach. Differently from other RL-based solutions in VEC scenarios [91], we propose here a new collaborative approach. By gaining from different vehicular communication paradigms, i.e., V2V and V2I, allowing information exchange among nodes [62], we aim at a better understanding of the local environment to be used for the development of two RL solutions.

In the first approach, a V2I collaborative Q-learning method is considered, in which VNs participate in the training process of centralized Q-agents. In the second approach, each VN is made aware of the nearby environment, and the potential offloading neighbors through the V2V links, leading to better decisions. A more advanced deep learning-based approach is also considered to estimate the action value functions for both the Q learning approaches allowing the possible extension towards high dimensional scenarios. During the learning phase, several scenarios are considered based on the VNs local environment. In the end, the numerical results show that the proposed solutions provide better latency and energy performance for end-users with respect to other benchmark methods.

In the following parts, we describe the system model, problem formulation, proposed solutions, and numerical results in detail.

4.3.1 System Model and Problem Formulation

The IoV scenario under consideration is composed of a set $\mathcal{V} = \{VN_1, \dots, VN_m, \dots, VN_M\}$ of M VNs, and a set $\mathcal{R} = \{RSU_1, \dots, RSU_n, \dots, RSU_N\}$ of N RSUs, creating the urban vehicular service. In addition, one MBS able to cover the whole area is supposed. Both RSUs and MBS act as edge nodes providing EC services to the VNs, enabling computation-intensive applications and services at the edge. The IoV system is modeled in a time-discrete manner, and the network parameters are constant over each time interval τ, where τ_i identifies the ith time interval, i.e., $\tau_i = \{\forall t | t \in [i\tau, (i+1)\tau]\}$.

Each VN is equipped with communication, computing, and storage elements, where it is supposed that it can communicate with a maximum bandwidth b_m and can process with a maximum computational capability η_m. In addition, by focusing on the ith time interval, it is supposed to have a battery capacity E_m^c and a battery level $E_m(\tau_i)$. The mth VN is supposed to be located in the position $\{x_m(\tau_i), y_m(\tau_i)\}$, while it moves at a speed $\vec{v}_m(\tau_i)$ along the road paths, where:

$$\vec{v}_m(\tau_i) = \frac{(x_m(\tau_i), y_m(\tau_i)) - (x_m(\tau_{i-1}), y_m(\tau_{i-1}))}{\tau}, \; i \geq 1.$$

Each RSU can be identified through a set of parameters, where the nth RSU, located at the position $\{x_n^R, y_n^R\}$, can provide communication with a maximum bandwidth B_n, and can process with a maximum computational capability H_n. Similarly, the MBS can be identified through its position $\{x^M, y^M\}$, maximum bandwidth B^M, and maximum computational capability H^M.

We consider that RSUs and MBS compose a multi-service network able to provide multiple services to the IoV environment. By assuming that $\mathcal{S} = \{S_1, \ldots, S_s, \ldots, S_S\}$ is the set of all the possible services that can be provided, due to the limited available resources, each RSU can provide only a subset of services. Thus, for the RSU_n we can identify $\mathcal{S}_n \subseteq \mathcal{S}$ as the set of services provided by it. Since the MBS has more resources, it is supposed to offer the whole service set $\mathcal{S}$. Finally, the nth RSU is supposed to have a limited coverage range d_n, whose value depends on the communication technology and radio-propagation environment, and it is supposed to provide VEC services to the vehicles within the coverage area. Similarly, for the MBS, the coverage range d^M stands. Each $VN_m \in \mathcal{V}$ is supposed to be active in each time interval with a probability p_a within which it generates a computation task request $\rho_m(\tau_i)$ identified through the tuple $\langle D_{\rho_m}, D^r_{\rho_m}, \Omega_{\rho_m}, T_{\rho_m}, S_{\rho_m} \rangle$ corresponding to a task with size D_{ρ_m} Byte, expected to give in output a result with size $D^r_{\rho_m}$ Byte, requesting Ω_{ρ_m} CPU execution cycles, with a maximum execution latency T_{ρ_m} and requesting service S_{ρ_m}.

4.3.1.1 RSU Selection

We define a binary VN-RSU assignment matrix $\mathbf{A}(\tau_i) = \left\{a_{m,n}(\tau_i)\right\}_{M\times N} \in \{0,1\}$ with size $M \times N$, where $a_{m,n}(\tau_i) = 1$ if VN_m is assigned to RSU_n in the interval τ_i, and $\sum_{n=1}^{N}\sum_{m=1}^{M} a_{m,n}(\tau_i) = M$, imposing that each VN is able to offload data to only one RSU[6]. Moreover, $a_{m,n}(\tau_i) = 1 \iff S_{\rho_m} \in \mathcal{S}_n$ that is to say the assignment can occur only if the requested service has been deployed on the nth RSU.

We assume to perform partial offloading, that is to say, each task generated by the VNs can be split, and a portion remotely processed while the remaining is processed locally [80]; the portion offloaded by VN_m at τ_i is identified as $\alpha_{\rho_m}(\tau_i) \in [0,1]$. Here, $\alpha_{\rho_m}(\tau_i) = 0$ corresponds to the complete local processing of the task, while $\alpha_{\rho_m}(\tau_i) = 1$ to the complete offloading of the task to the selected VEC node. During the partial offloading process, the task processing operations, performed locally by VNs and remotely at the RSU-based edge servers, are supposed to be executed in parallel to reduce the overall processing time [92]. Each RSU is supposed to have a limited amount of computation and communication resources; hence, it can only serve a limited number of users. Therefore:

$$\begin{cases} \displaystyle\sum_{m=1}^{M_n(\tau_i)} \alpha_{\rho_m}(\tau_i) \cdot \Omega_{\rho_m} \le H_n^R \cdot \tau \quad \forall i & \text{(4.58a)} \\ \displaystyle\sum_{m=1}^{M_n(\tau_i)} b_m(\alpha_{\rho_m}(\tau_i), D_{\rho_m}) \le B_n^R \quad \forall RSU_n \in \mathcal{R}, i & \text{(4.58b)} \end{cases}$$

[6]Given the complex nature of the considered problem, especially over a dynamically changing vehicular environment, we have assumed that each vehicle can select only one edge node for offloading its data.

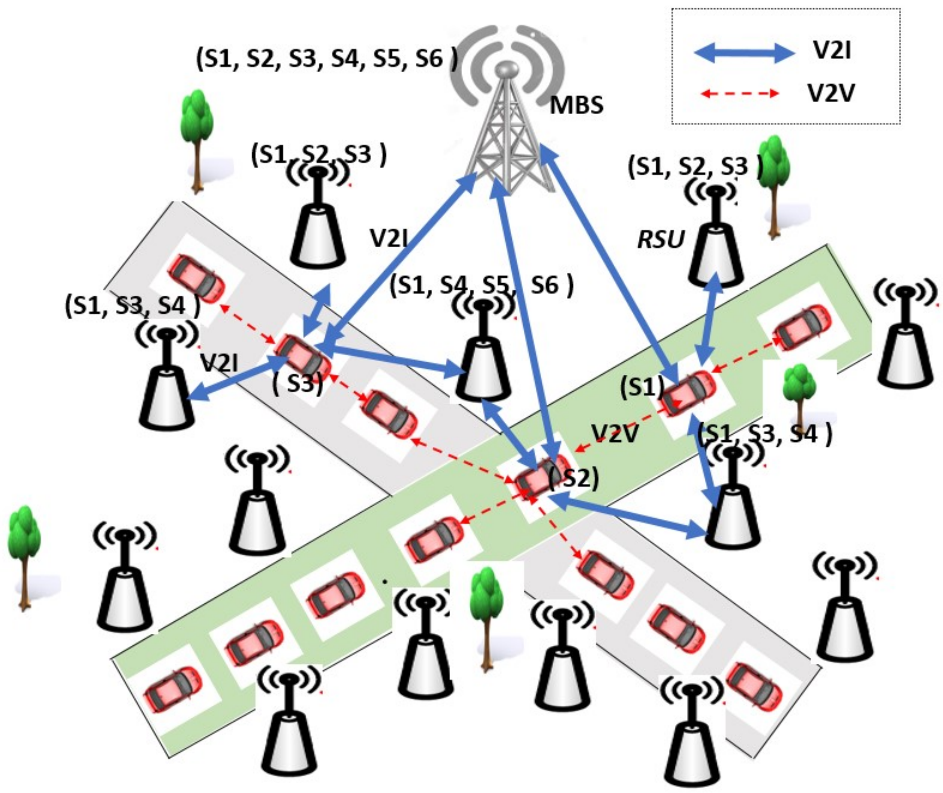


FIGURE 4.10: The multi-service IoV system architecture.

where, $M_n(\tau_i) = \sum_{m=1}^{M} a_{m,n}(\tau_i)$ is the number of vehicles assigned to the RSU n in the ith interval. While (4.58a) models an upper bound on the processing capacity of the RSU, (4.58b) introduces a transmission capacity upper bound for the VNs connected to any RSU. Here, $b_m(\alpha_{\rho_m}(\tau_i), D_{\rho_m})$ corresponds to the communication resources available for the transmission of vehicular tasks depending upon its task size and the offloading parameter. It is worth to be noticed that the capacity of each link depends on the specific communication technology and it is out of the scope of this work.

In Fig. 4.10, a possible IoV scenario is depicted, where different VNs request different services, and are covered by RSUs hosting different service types. In addition, VNs are supposed to be in the coverage area of the MBS.

4.3.1.2 Task Processing

The processing time and energy needed for computing a task depend on the offloading policy and the selected node processing characteristics. The generic expression for the time and energy spent for the ρ_mth task computation on any device is given by [84]:

$$T_{c_l}^{\rho_m} = \frac{\Omega_{\rho_m}}{o_l f_l} \tag{4.59a}$$

$$E_{c_l}^{\rho_m} = k_l \frac{\Omega_{\rho_m}}{o_l} f_l^2 \tag{4.59b}$$

where o_l and f_l are the number of Floating-point Operation Per Second (FLOPS) per CPU-cycle and CPU-frequency, respectively, whether l is a generic index identifying one of the possible processing nodes among VNs (m), RSUs (n) and MBS. In (4.59b), k_l is a constant coefficient representing the chip architecture of the generic lth device.

Since we assume to perform a partial computation offloading, each VN transmits a portion of its task to the assigned RSU and receives back the result. In general, the transmission time and energy between VN_m and RSU_n for task ρ_m is given by:

$$T_{tx,mn}^{\rho_m} = \frac{D_{\rho_m}}{r_{mn}} \tag{4.60a}$$

$$E_{tx,mn}^{\rho_m} = T_{tx,mn}^{\rho_m} Pt_m \tag{4.60b}$$

where r_{mn} is data-rate of the link between the two nodes, while $P_{tx,m}$ is the transmission power of VN_m. Similarly, the reception time and energy to receive back the processing result having size $D_{\rho_m}^r$ from RSU_n by VN_m are, respectively:

$$T_{rx,nm}^{\rho_m} = \frac{D_{\rho_m}^r}{r_{mn}} \tag{4.61a}$$

$$E_{rx,nm}^{\rho_m} = T_{rx,nm}^{\rho_m} P_{rx,m} \tag{4.61b}$$

where $P_{rx,m}$ is the power spent for receiving at the RSU_m side. A symmetric channel is considered between VN_m and RSU_n. The expression for the channel transmission rate is based on the Shannon capacity formula and can be written as:

$$r_{mn} = b_m \log_2 \left(1 + \frac{Pt_m}{L(d_{mn})N_0}\right)$$

where $P_{tx,m}$ is the transmission power of a device m, $L(d_{mn})$ is the path loss at a distance d_{mn}, and $N_0 = N_T b_m$ is the thermal noise power.

Task Offloading Process If VN_m is assigned to RSU_n, then the time and energy required to offload the task to the selected RSU and to get back the result in the ith interval is:

$$T_{m,n}^{off}(\alpha_{\rho_m}(\tau_i)) = \alpha_{\rho_m}(\tau_i)\left(T_{tx,mn}^{\rho_m} + T_{c_n}^{\rho_m} + T_{rx,nm}^{\rho_m}\right) \tag{4.62a}$$

$$E_{m,n}^{off}(\alpha_{\rho_m}(\tau_i)) = \alpha_{\rho_m}(\tau_i)\left(E_{tx,mn}^{\rho_m} + E_{rx,nm}^{\rho_m}\right) \tag{4.62b}$$

where $T_{tx,mn}^{\rho_m}$, $T_{c_n}^{\rho_m}$, and $T_{rx,mn}^{\rho_m}$ are, respectively, the transmission time, computation time on nth RSU, and the receiving time for the task ρ_m generated by VN_m during offloading phase, and $E_{tx,mn}^{\rho_m}$ and $E_{rx,mn}^{\rho_m}$ are, respectively, the energy consumed during the task transmission and result collection phases on device. Since RSU nodes are supposed to be connected to the electrical grid, we do not consider their energy consumption in the energy analysis.

Local Computation The amount of time and energy required for the local computation in the ith interval is:

$$T_m^{loc}(\alpha_{\rho_m}(\tau_i)) = \left(1 - \alpha_{\rho_m}(\tau_i)\right) T_{c_m}^{\rho_m} \tag{4.63a}$$

$$E_m^{loc}(\alpha_{\rho_m}(\tau_i)) = \left(1 - \alpha_{\rho_m}(\tau_i)\right) E_{c_m}^{\rho_m} \tag{4.63b}$$

where $T_{c_m}^{\rho_m}$ and $E_{c_m}^{\rho_m}$ are the time and energy spent for the whole task ρ_m local processing, while $\alpha_{\rho_m}(\tau_i)$ is the portion of the task locally processed at the time interval τ_i.

Partial Offloading Computation The delay and the energy consumed during the task processing phases, when partial offloading is performed in the ith interval, can be written as:

$$T_m^{\rho_m}(\alpha_{\rho_m}(\tau_i)) = \max\left\{T_{m,n}^{off}(\alpha_{\rho_m}(\tau_i)), T_m^{loc}(\alpha_{\rho_m}(\tau_i))\right\}$$
$$E_m^{\rho_m}(\alpha_{\rho_m}(\tau_i)) = E_{m,n}^{off}(\alpha_{\rho_m}(\tau_i)) + E_m^{loc}(\alpha_{\rho_m}(\tau_i)).$$

Since the local computation and offloading processes are executed in parallel, the total task processing latency is the maximum of the two.

4.3.1.3 Vehicle Mobility and Sojourn Time

The VN mobility poses some constraints to the computation offloading decisions. Due to the VNs mobility, each offloading operation should be completed by the VN sojourn time, corresponding to the amount of time it remains under the coverage of the selected RSU [80], otherwise the system may be affected by additional latency due to, e.g., vehicle handover, service migration, additional signaling for managing vehicles and services mobility [13]. The remaining distance in which the mth VN remains in the coverage of nth RSU is:

$$D_{m,n}(\tau_i) = \sqrt{d_n^2 - (y_n - y_m(\tau_i))^2} \pm (x_n - x_m(\tau_i)) \tag{4.64}$$

where $\{x_m(\tau_i), y_m(\tau_i)\}$ and $\{x_n, y_n\}$ are, respectively, the position of the mth VN and the nth RSU at time interval τ_i and R_n is the coverage radius of the nth RSU. Hence, the sojourn time for the mth VN can be written as:

$$T^{soj}_{m,n}(\tau_i) = \frac{D_{m,n}(\tau_i)}{|\vec{v}_m(\tau_i)|} \quad \forall i \tag{4.65}$$

Each vehicle should finish the offloading process and receive the results back within the sojourn time, hence:

$$T^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) \leq T^{soj}_{m,n}(\tau_i) \quad \forall i \tag{4.66}$$

4.3.1.4 Problem Formulation

The main aim of this work is to optimize the network-wide performance of the VEC-enabled IoV network. We aim to optimize the performance in terms of overall latency and energy consumed during the offloading process towards edge servers by selecting proper RSU nodes and offloading amounts. For this, we formulate the joint latency and energy minimization problem as:

P1 :

$$\min_{\mathcal{A},\mathbf{A}} \left\{ \sum_{n=1}^{N} \sum_{m=1}^{M} \left[\gamma_1 T^{\rho_m}_m \left(\alpha_{\rho_m}(\tau_i)\right) + \gamma_2 E^{\rho_m}_m \left(\alpha_{\rho_m}(\tau_i)\right) \right] \right\} \forall i \tag{4.67}$$

s.t.

$$\mathbf{C1} : \sum_{n=1}^{N} \sum_{m=1}^{M} a_{m,n}(\tau_i) = M \tag{4.68}$$

$$\mathbf{C2} : a_{m,n}(\tau_i) = 1 \iff S_{\rho_m} \in \mathcal{S}_n \quad \forall i \tag{4.69}$$

$$\mathbf{C3} : \text{Eqs. (4.58a) and (4.58b)} \tag{4.70}$$

$$\mathbf{C4}: T_m^{\rho_m}\left(\alpha_{\rho_m}(\tau_i)\right) \leq T_{\rho_m} \quad \forall VN_m \in \mathcal{V},\ \forall i \tag{4.71}$$

$$\mathbf{C5}: \text{Eq. } (4.66) \tag{4.72}$$

$$\mathbf{C6}: E_m^{\rho_m}(\alpha_{\rho_m}(\tau_i)) \leq E_{c_m}^{\rho_m} \tag{4.73}$$

$$\mathbf{C7}: 0 \leq \gamma_1, \gamma_2 \leq 1;\ \gamma_1 + \gamma_2 = 1 \tag{4.74}$$

where $\mathcal{A} = \{\alpha_{\rho_m}\}^M$ is the computation offloading matrix, and γ_1, γ_2 are two weighting coefficients for balancing latency and energy consumption. **C1** stands that each VN can select at most one RSU for the computation offloading. According to **C2**, the selected edge node must be able to provide the service requested by the VNs. **C3** sets the bounds in terms of processing capacity and resource blocks requested by VNs towards edge nodes, while **C4** puts a limit on the maximum processing time as one of the task requirements. According to **C5**, for avoiding handover phenomena and related latency, each VN should complete the offloading process before it passes through the selected RSUs coverage. According to **C6**, the total energy required for the task processing during the partial computation offloading process should be bounded by the energy needed to process the complete task locally. **C7** stands that the two weighting coefficients (γ_1, γ_2) should be between 0 and 1 with their sum equal to 1.

4.3.2 Q-Learning Based Joint Network Selection and Computation Offloading

The decision on the EN to be selected for computation offloading and the amount of data to be offloaded can depend upon several factors, such as VNs position and speed, nearby VNs, the available number of RSUs for offloading, availability of the requested service, etc. If a VN is under the coverage of multiple ENs, the proper EN can be selected by sequentially testing ENs one after another. Also, VNs can make sequential decisions for finding a proper amount of data to be offloaded towards the EN. Every decision taken by VNs can alter the surrounding environment's state, and can be mapped with a reward (i.e, an increase or decrease in the task processing time and energy). Therefore, finding the proper EN and the corresponding data to be offloaded in the dynamic vehicular environment can be considered a sequential decision-making problem that can effectively be solved through the RL approach.

RL and multi-agent RL (MARL)-based methods have found applications in vehicular scenarios. For avoiding the unbearable training costs and for adequate training performance, various RL/MARL training architectures are considered in the past. A centralized MARL training process having an exponentially scalable set of actions and observation spaces, is often neglected in highly complicated vehicular environments. In the case of a fully decentralized approach, an independent set of agents tries to optimize a common reward function over its local environment. Issues like spurious rewards, mainly due to partial observability, prevent their use in the considered problem. Due to the involvement of high-speed VNs and a dynamically changing environment, a large set of training agents are required for solving highly complex problems such as the one considered here. This scales up the issues of exponential scalability and spurious rewards in these two traditional approaches.

In recent times, some other MARL architectures have been proposed, mainly for solving the previous challenges. One such example is [93], where the authors have proposed a value-decomposition network-based cooperative MARL architecture. A deep neural network-based back-propagation approach is used to decompose a common value function into a set of agent-based value functions. However, the complexity of the considered decomposition network, the limited use of state information during training, and applicability towards a reduced class of centralized value functions are bottlenecks. Also in [94], another value function-based approach is considered by estimating a joint action-value function from a set of local observation-based action individual agents values through neural networks. The issues of complexity, scalability, etc., prevent the use of this MARL-based method for solving the given problem.

With the availability of novel communication technologies such as V2V, V2I, etc, VNs can share important environmental parameters. Such information can be integrated into the training process of a collaborative RL strategy to solve the given vehicular problem effectively and with a reduced complexity with respect to a MARL approach. Therefore, below, we have proposed efficient vehicular communication-based cooperative RL strategies for solving the problem at hand. In RL, at any given time τ, an intelligent agent in a particular state $St(\tau)$ interacts with the dynamic environment En, through the selected action $ak(\tau)$, and, in return, receives observations in terms of a state change $St^*(\tau)$ and rewards R. The agent tries to maximize the future reward value over consecutive discrete time steps by taking the actions from the current state in the environment based on the received observations [95]. Therefore, state-space, action-space, and the reward function are the main elements of the RL process.

In the considered RL-based framework, multiple agents collaboratively perform the training operations for finding the optimal policy aimed at minimizing the joint latency and energy cost with reliable network selection and computation offloading operations. We have considered a centralized training architecture assisted by vehicular communication data for improving the training performance. MBS, as a centralized entity, can perform the training process for individual training scenarios (i.e., RL agents) for finding the optimal policies. It collects the information from VNs and uses it during the learning process of the RL agents by implementing a collaborative learning process. More description about the learning scenarios is provided below in subsections 4.3.2.1 and 4.3.2.1.

The State-space, Action-space, and Reward function can be identified as follows.

State-Space ($\mathcal{ST}$) In a multi-service multi-user vehicular environment the available resources for the computation offloading process are changing continuously over time and are function of the offloading and network selection decisions taken by vehicles. Therefore, we have defined a state-space composed of a discrete set of states as a function of resources available for computation offloading. The state-space is a discrete set of states identifying the RSUs to be selected and their resources available for the computation offloading. Since each VN_m can exploit a different number of RSUs for offloading, we define $\mathcal{R}_m = \left\{RSU_n | D_{m,n}(\tau_i) > 0 \wedge S_{\rho_m} \in \mathcal{S}_n, \forall \mathcal{R}\right\}$ as the set of RSUs available for the mth VN for the offloading operation; we consider a multi-dimensional state-space representation where the νth state-space corresponds to the *scenario* with ν available RSUs. For each scenario the related state-space is function of sojourn time, required latency, VN resources, resources of the available RSUs; thus, each state St_ν at time τ_i is defined as:

$$St_\nu(\tau_i) = f(\alpha_{\rho_m}(\tau_i), T^{soj}_{m,n}(\tau_i), B_n, H_n, D_{\rho_m}, D^r_{\rho_m}, \Omega_{\rho_m}, T_{\rho_m}, S_{\rho_m}). \tag{4.75}$$

We suppose to limit the multi-dimensional state space to $\bar{N}$ scenarios, hence, $\nu = 1, \ldots, \bar{N}$.

Action-Space ($\mathcal{AS}$) The action space defines all the possible actions available during the learning process of an RL-agent. We consider to have $\bar{N}$ agents collecting information for each possible scenario, composed by a different number of available

RSUs. At each iteration, each agent explores the available RSUs, by properly setting a binary vector $\mathbb{R}_\nu(\tau_i) = \{0,1\}^\nu$ mapping the RSUs selection among the ν available in the given scenario. At the same time, the offloaded amount is selected, by either increasing, decreasing, or keeping the same amount as the previous iteration. Thus, for the τ_ith instance, we have $\alpha_{\rho_m}(\tau_i) \in \left\{\alpha_{\rho_m}(\tau_i - 1), \alpha_{\rho_m}(\tau_i - 1) \pm \Lambda\right\}$ where Λ is a step increase or decrease of the offloading amount. The generic action ak_ν for the νth scenario at time τ_i can be defined as $ak_\nu(\tau_i) = \left\{\mathbb{R}_\nu(\tau_i), \alpha_{\rho_m}(\tau_i)\right\}$ where $\mathbb{R}(\tau_i)$ is a binary vector with length ν, where 1 in the nth position corresponds to the selected RSU.

Reward Function (R) The reward function (R) is defined as the joint objective function of time and energy consumed for the complete task processing (4.67). In addition, three penalty terms are also considered modeling when the agent fails to satisfy the latency and energy constraints, as defined in (4.71), (4.66) and (4.73). Thus, the expression for the reward function is given by:

$$\begin{aligned} R(St_\nu(\tau_i), ak_\nu(\tau_i)) = \gamma_1 T_m^{\rho_m}\left(\alpha_{\rho_m}(\tau_i)\right) + \gamma_2 E_m^{\rho_m}\left(\alpha_{\rho_m}(\tau_i)\right) + \Upsilon_1 \cdot \max(0, C_1(St_\nu(\tau_i), ak_\nu(\tau_i))) \\ + \Upsilon_2 \cdot \max(0, C_2(St_\nu(\tau_i), ak_\nu(\tau_i))) + \Upsilon_3 \cdot \max(0, C_3(St_\nu(\tau_i), ak_\nu(\tau_i))) \end{aligned} \tag{4.76}$$

where Υ_1, Υ_2 and Υ_3 are the weighting coefficients for the penalty values, and:

$$C_1(St_\nu(\tau_i), ak_\nu(\tau_i)) = T_m^{\rho_m}\left(\alpha_{\rho_m}(\tau_i)\right) - T_{\rho_m} \tag{4.77a}$$

$$C_2(St_\nu(\tau_i), ak_\nu(\tau_i)) = T_{m,n}^{off}(\alpha_{\rho_m}(\tau_i)) - T_{m,n}^{soj}(\tau_i) \tag{4.77b}$$

$$C_3(St_\nu(\tau_i), ak_\nu(\tau_i)) = E_m^{\rho_m}(\alpha_{\rho_m}(\tau_i)) - E_{c_m}^{\rho_m}. \tag{4.77c}$$

(4.77a) is the additional penalty value when VN fails to perform the task processing operation within the maximum execution latency bound, (4.77b) is the additional penalty when VNs fails to satisfy the sojourn time bounds during the offloading process, and (4.77c) is the additional cost when VN fails to follow the energy constraint in (4.73).

4.3.2.1 Collaborative Q-Learning Solutions for Joint Network Selection and Offloading

Q-learning is one of the most-known techniques used to solve RL-based problems [96]. In Q-learning, each state-action pair (St, ak) has a Q value, defined as $Q(St, ak)$, which provides the expected future reward for an agent from state St if he decides to take action a. Each agent receives the input values as set of environment state $(\mathcal{ST})$ including the possible terminal state St_{opt}, reward function R and action set $(\mathcal{AS})$. Other input parameter includes the learning rate $\gamma_{lr} \in \{0, 1\}$, discount factor $\Delta \in \{0, 1\}$, and a number of learning episodes E. In every episode, the agent selects the initial state St_{in}, then it takes a random action ak over the environment, receives a reward related to that action and a new state value. The widely known Epsilon-Greedy algorithm (EGA) is used for selecting the future action [95]. In EGA, each Q-agent picks the action based upon the Exploration vs Exploitation dilemma, with the exploration probability e. For every iteration, the Q-values $(Q(St_t, ak_t))$ for the state-action pair (St_t, ak_t) are updated based upon the temporal difference expression defined as:

$$Q^{new}(St^t_v, ak^t_v) \leftarrow Q(St^t_v, ak^t_v) + \gamma_{lr}\big(R_{St^t_v, ak^t_v} + \Delta\left(\max_A\left\{Q(St^{t+1}_v, ak^{t+1}_v)\right\} - Q(St^t_v, ak^t_v)\right)\big) \tag{4.78}$$

Two novel Q-learning based solutions for the joint selection of network and computation offloading problem are considered here. In the first method, a V2I collaborative

Q-learning-based solution is considered, in which several randomly distributed VNs participate in the training process of a Q-agent. In the beginning, a centralized server collects data from VNs by exploiting the V2I communication links. Next, vehicles are classified based upon the local environment scenarios. In the end, every group trains the Q-agent based on its local environment. In the other approach, each VN explores the V2V communication links for collecting information about the nearby VNs. This allows to make a better decisions in terms of network selection and computation offloading.

V2I Assisted Collaborative Q-Learning (V2I-AC) In the V2I collaborative approach, multiple randomly located VNs participates to the training process towards a centralized set of agents through the V2I links. Since the cardinality of $\mathcal{R}_m$ depends on VN_m and RSUs positions as well the deployed services, we define multiple training scenarios, each one characterized by the number of available RSUs, i.e., $|\mathcal{R}_m|$. The scenario vector $\mathcal{SV}_{rsu}$ includes all possible available RSUs for computation offloading, i.e., $\mathcal{SV}_{rsu} = \{2, \cdots, RSU_{vmax}\}$. Here, RSU_{vmax} is the maximum number of RSU nodes available for computation offloading. In the Q-learning process, we generate $\bar{N}$ Q-tables where, $Q_v(\mathcal{ST}, \mathcal{AS})$ is the table of the vth scenario.

By referring to the pseudocode in Algorithm 3, in the V2I collaborative Q-learning approach, the VNs are first classified based on their scenario (lines 1-3). For each VN_m, the cardinality of available RSUs are evaluated, i.e., $|\mathcal{R}_m|$. In order to limit the number of possible scenarios to $\bar{N}$, the mth VN is managed by the vth agent, where $v = \min\{|\mathcal{R}_m|, \bar{N}\}$. Through this, we can classify all VNs into different groups, with vth scenario having $\bar{m}_v$ VNs for the collaborative training process, with $\bar{m}_v = \{|VN_m| : |\mathcal{R}_m| = \min(v, \bar{N}), \forall \mathcal{V}\}$. After the VNs classification, the training process for each scenario is performed (lines 4-16). The number of training episodes for each

Algorithm 3 V2I Assisted Collaborative Q-Learning (V2I-AC)

```
Input: Set of VNs $\mathcal{V}$, $\bar{N}$, $e, \gamma_{lr}, \Delta, \mathcal{ST}, \mathcal{AS}, R, \mathcal{I}$
Output: $\{Q(\mathcal{ST}, \mathcal{AS})\}$
 1: for all $VM_m \in \mathcal{V}$ do
 2:     Find $\mathcal{R}_m$ and set $\nu = \min\{|\mathcal{R}_m|, \bar{N}\}$, $\bar{m}_\nu \leftarrow \bar{m}_\nu + 1$
 3: end for
 4: for all $\nu = 1, \dots, \bar{N}$ do
 5:     for all $VN_m$ such that $|\mathcal{R}_m| = \min(\nu, \bar{N})$ do
 6:         Select a random initial state $St_\nu^0 \in \mathcal{ST} \wedge St_\nu^0 \neq \bar{S}t_\nu$
 7:         $St_\nu \leftarrow St_\nu^0, \quad it = 0$
 8:         while $St_\nu^t \neq \bar{S}t_\nu || it = \mathcal{I}$ do
 9:             $it = it + 1$
10:             Select action $ak_\nu \in \mathcal{AS}$ with probability $e$
11:             Determine next state $(St_\nu^{t+1})$ and reward received
12:             Use Eq. (4.78) to find the TD and update $Q$ table.
13:             $St_\nu \leftarrow St_\nu^{t+1}$
14:         end while
15:     end for
16:     return $Q_\nu(\mathcal{ST}, \mathcal{AS})$
17: end for
```

scenario is equal to the number of VNs in that particular scenario group, i.e., $\bar{m}_\nu$. In each episode, the agent selects a non-optimal random initial state St_ν^0, then it takes a random action ak_ν over the environment, receives a reward and a new state value. For every iteration, the Q-values $(Q(St_\nu^t, ak_\nu^t))$ for the state-action pair (St_ν^t, ak_ν^t) are updated following the the Temporal Difference (TD) expression (4.78). Once the final state $\bar{S}t_\nu$ or $it = \mathcal{I}$ has been reached, with $\mathcal{I}$ being a maximum number of iterations, the agent starts the new episode of learning. This process is repeated till a predefined optimal state is reached. In the end, we receive a Q-table associated with that particular scenario $(Q_{SV_{rsu}})$.

In the V2I collaborative Q-learning method, we aim to find the joint solution for the network selection and computation offloading problem. The solution is composed of a network selection vector $\mathbb{R}_\nu$, depending on the scenario in which any given VN are classified, and the offloading amount α_{ρ_m}. In case none of the surrounding RSUs have the service requested by VN_m, i.e., $\mathcal{R}_m = \emptyset$, VN_m can offload the data towards the MBS, which contains all the services requested by VNs.

V2V Assisted Collaborative Q-Learning (V2V-AC) In this approach, we suppose that the VNs exploit V2V communication links for acquiring knowledge of the potential competing VNs around them before offloading for the training phase. Since the VNs offloading strategy is unknown by each VN we assume that a predefined VN-RSU selection method where each competing VN selects the nearest RSU node for complete task offloading is assumed for resource allocation purposes. Both communication and computation resources of RSU nodes are equally shared among the assigned VNs. Similarly to the V2I approach, we consider different agents acting on different scenarios, where, in this case, we extend also to the number of nearby VNs. Different learning scenarios are considered based upon the number of available RSUs for offloading and the nearby VNs. Therefore, while the number of available RSUs is determined in the same way, we extend the considered scenarios up to $\bar{N} \cdot \bar{M}$ where $\bar{M}$ is the maximum number of nearby VNs to be considered.

By defining $\mathcal{V}_m = \left\{RSU_{m'} | d_{m,m'}(\tau_i) \leq d_{V2V}, \forall \mathcal{R}\right\}$ as the set of VNs within a certain d_{V2V} coverage distance, we can set as $\mu = \min\{|\mathcal{V}_m|, \bar{M}\}$ as the scenario identifying all the vehicles with μ surrounding VNs to be exploited. Given this, the (ν, μ)th agent will exploit as training nodes all the VNs having ν available RSUs and μ competing VNs.

The number of available RSUs could be in the range of 2 to RSU_{vmax}. Similarly, the number of VNs around each test VN is ranged between 0 to VN_{vmax}. Thus, the two scenario vectors are $\mathcal{SV}_{rsu} = [2, \cdots, RSU_{vmax}]$ and $\mathcal{SV}_{vn} = [0, \cdots, VN_{vmax}]$. Therefore, each $SV(i,j) = \{\mathcal{SV}_{rsu}(i), \mathcal{SV}_{vn}(j)\}$ represents the training scenario considered.

Algorithm 4 V2V Assisted Collaborative Q-Learning (V2V-AC)

```
Input: Set of VNs $\mathcal{V}$, $\bar{M}, \bar{N}, e, \gamma_{lr}, \Delta, \mathcal{ST}.\mathcal{AS}, R, \mathcal{I}$
Output: $\{Q(\mathcal{ST}, \mathcal{AS})\}$
 1: for all $VN_m \in \mathcal{V}$ do
 2:     Find $\mathcal{R}_m$ and $\mathcal{V}_m$
 3:     Set $\nu = \min\{|\mathcal{R}_m|, \bar{N}\}$ and $\mu = \min\{|\mathcal{V}_m|, \bar{M}\}$
 4:     $\bar{m}_{(\nu,\mu)} \leftarrow \bar{m}_{(\nu,\mu)} + 1$
 5: end for
 6: for all $\nu = 1, \ldots, \bar{N}$ do
 7:     for all $\mu = 1, \ldots, \bar{M}$ do
 8:         Follow steps from 5 to 14 of Algorithm 3
 9:         return $Q_{(\nu,\mu)}(\mathcal{ST}, \mathcal{AS})$
10:     end for
11: end for
```

The Algorithm 4 describes the steps used during the training phase of each scenario. In lines 1-5, each VN scenario is determined, by estimating the number of available RSUs for data offloading $\mathcal{R}_m$ and the number of VNs around it ($\mathcal{V}_m$). Based on their training scenario, each VN will be classified into different groups. At the end of the training sessions, each scenario will have a separate Q-table. Similar to the previous approach, the solution is composed of network selection vector $\mathbb{R}_\nu$ and the offloading amount α_{ρ_m}. In case any of the surrounding RSUs have the requested service by the VN_m, i.e., $\mathcal{R}_m = \emptyset$, VN_m can offload the data towards the MBS, which contains all the services requested by VNs.

4.3.2.2 Deep Learning Based Solutions

The proposed collaborative learning-based methods use Q-learning with Q-table as a baseline solution method. Such techniques are widely used for solving RL problems with simple settings, however, they are not targeted with problems with the curse of high dimensionality, i.e., larger state or action spaces. In such scenarios, more advanced techniques are suitable. For the case of VNs, this is often the case. Therefore a deep learning-based approach is also discussed, where a Deep Q Network

(DQN) is used for approximating the Q-function, i.e., the action value function. In this case, the DQN can replace the Q-learning process (lines 5-14 in Algorithm 3) for both V2I-AC and V2V-AC methods when estimating the Q-values.

As shown in Fig. 4.11, the considered DQN is based on the presence of both primary and target networks both with L_d layers with n_l $(l \in L_d)$ neurons for estimating the Q-values. Considering an approach similar to [97], the primary network is used for estimating the real/primary Q-value while the target Q-values are estimated through the target network. The learning agent utilizes the backpropagation and gradient descent processes with Mean Square Error (MSE) based loss function for reducing the gap between the primary and the target Q-values. For the scenario v, the loss function is defined as,

$$L(w, v) = R_{St_v^t, ak_v^t} + \Delta \max_A \left\{ Q(St_v^{t+1}, ak_v^{t+1}; w_v^{'}) \right\} - Q(St_v^t, ak_v^t; w_v) \tag{4.79}$$

The primary Q-value, given as $Q(St_v^t, ak_v^t; w_v)$, is a result of primary network with parameters w_v, while the target Q-value $R_{St_v^t, ak_v^t} + \Delta \max_A \left\{ Q(St_v^{t+1}, ak_v^{t+1}; w_v^{'}) \right\}$ is based upon the results of the target network with parameters $w_v^{'}$. After collecting the Q-values, the EGA is used for selecting the future action. The agent's learning experiences are stored in the forms of tuple $(St_v^t, ak_v^t, R_{St_v^t, ak_v^t}, St_v^{t+1})$ constituted by current state, action, reward received and the next state in the replay memory of size r_b and used during the training process. Randomly sampled experiences forming a batches of size b helps to improve the training performance. The V2I-AC and V2V-AC techniques built by using the DQN-based approaches are named V2I-DAC and V2V-DAC in the following parts.

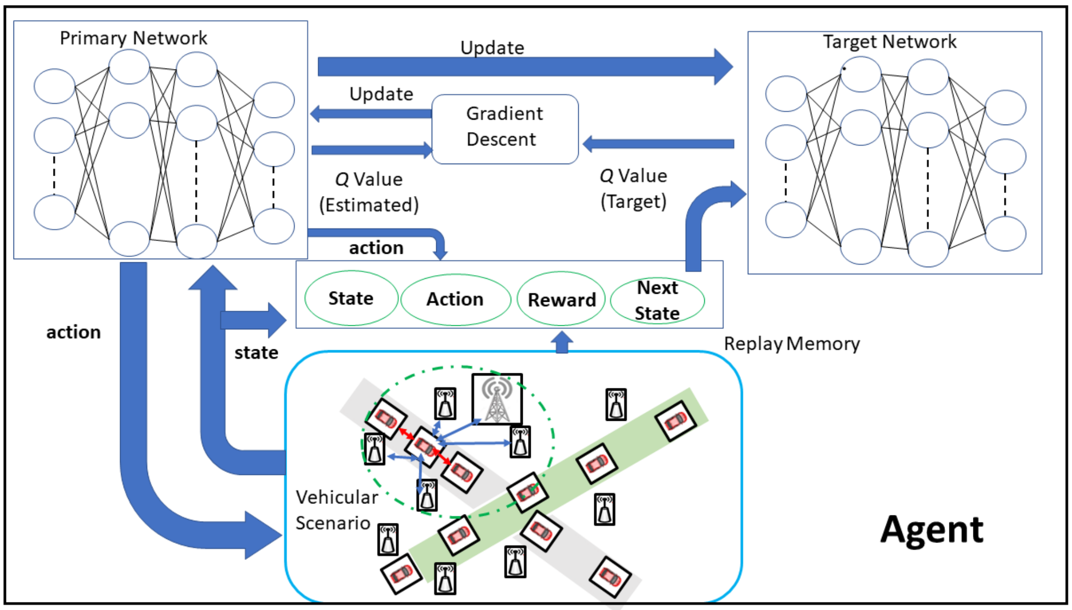


Figure 4.11: DQN Architecture.

4.3.3 Limited Search-space based Heuristic Approach

With the involvement of a huge number of VUs, the problem in (4.67) is highly complex to be solve through the traditional heuristic approaches mainly due to large solution space $\mathcal{SP}$. For comparison purposes, a two-step limited search space-based heuristic is also considered, able to reduce the computation complexity through design parameters bounding the size of the overall search space. Algorithm 5, provides the step-by-step process to be implemented. In the first step (lines 1-13), all VUs are assigned to the available edge nodes based on the requested services and the distance measures. The ratio between the sojourn time distance and the distance between VU and edge node, i.e., $\frac{D_{m,n}(\tau_i)}{d_{m,n}(\tau_i)}$, allows selecting the edge node with the smallest communication cost and highest sojourn time (line 11). Thus, each VU greedily selects the edge node best suited for their operations without considering the presence of other competing users.

Next, a solution space $\mathcal{SP}_n\left(A^{''}(n), \Lambda_{hu}\right)$ is formed at the nth RSU node based upon the set of VUs requesting the services ($A^{''}(n)$) and step parameter Λ_{hu}. Each solution point contains a vector of offloading parameters given as,

$$\hat{\mathcal{A}}(n)_{(1\times A''(n))} = \left\{\alpha_{\rho_1}(\tau_i), \cdots, \alpha_{\rho_{A''(n)}}(\tau_i)\right\}$$

The step value taken by the offloading parameter (i.e., Λ_{hu}) is considered as a design parameter limiting the size of search space. Thus, the overall solution space of the nth RSU node is based upon the total number of VUs requesting the services and the design parameter Λ_{hu}. Next for each solution point, the objective function in (4.67) is analyzed along with the constraint set for finding the best possible solution (lines 17-25).

Algorithm 5 Limited Search-space based Heuristic Approach (LS-HA)

```
Input: Set of VNs $\mathcal{V}$, $R$, $\{\mathcal{R}_m\}, \Lambda_{hu}$
Output: $\{\mathcal{A}, \mathbf{A}\}$
 1: function FIND($A^{'}, A^{''}$})
 2:     for all $RSU_n = 1, \ldots, R$ do
 3:         for all $VM_m \in \mathcal{V}$ do
 4:             $A^{'}(m,n) = 1 \iff RSU_n \in \mathcal{R}_m$
 5:         end for
 6:         $A^{''}(n) = \sum_{m=1}^{M}(A^{'}(m,n))$
 7:     end for
 8: end function
 9: function ASSIGN($\mathbf{A} = \{a(m,n)\}$})
10:     for all $VM_m \in \mathcal{V}$ do
11:         $a_{m,n}(\tau_i) = 1 \iff \frac{D_{m,n}(\tau_i)}{d_{m,n}(\tau_i)} = \max_{n' \in \mathcal{R}_m}\left\{\frac{D_{m,n'}(\tau_i)}{d_{m,n'}(\tau_i)}\right\}$
12:     end for
13: end function
14: function OFFLOAD($\mathcal{A} = \{\alpha_{\rho_m}\}^{A^{''}(n)}, \forall n$)
15:     for all $RSU_n = 1, \ldots, R$ do
16:         Create $\mathcal{SP}_n(A^{''}(n), \Lambda_{hu}) = \{\hat{\mathcal{A}}(n)_{(1\times A''(n))}\}$ with all possible solution points to be searched in the
    reduced-size solution space.
17:         $C_h = \infty$
18:         for all $\hat{\mathcal{A}}(n) \in \mathcal{SP}$ do
19:             Use (4.67) to evaluate cost $C(\hat{\mathcal{A}}(n), \mathbf{A})$.
20:             Determine all constraint functions values.
21:             if ($C(\hat{\mathcal{A}}(n), \mathbf{A}) \leq C_h$ and all constraints are satisfied) then
22:                 $C_h = C(\hat{\mathcal{A}}(n), \mathbf{A})$ and $\{\alpha_{\rho_m}\}^{A^{''}(n)} = \hat{\mathcal{A}}(n)$
23:             end if
24:         end for
25:     end for
26: end function
27: return $\{\mathcal{A}, \mathbf{A}\}$
```

4.3.4 Numerical Results

The performance evaluation of the proposed solutions has been carried out through a Python-based simulator, using ML-related libraries such as NumPy, Pandas, Matplotlib. The scenario contains a multi-service VN composed of one MBS, and several randomly distributed RSUs and VNs. The main parameters used in computer simulations are in Table 4.2. We have considered a maximum of 350 RSUs, distributed in the coverage area, and a variable number of VNs from 100 to 1800. Each RSU provides a random number of services $|\mathcal{S}|$ between 1 and 6. We consider that VNs are not always active, while, at each time instant, they request a random service with a probability equal to 0.1, while with probability 0.9 they are inactive. The two weighting coefficients (γ_1, γ_2) in (4.67) have been set to 0.5.

In the Q-learning simulation, we have generated different scenarios, based on the available number of RSUs and the nearby competing VNs during offloading cases. In the training phase of each Q-agent, we have used 500 randomly distributed VNs in the MBS coverage area, considered our operational area. The maximum number of RSUs that can be exploited by each VN is $\bar{N} = 4$, while the maximum number of nearby VNs that can be exploited for the V2V approach is $\bar{M} = 15$. In the reward function, $\Upsilon_1 = 1.5$, $\Upsilon_2 = 0.8$, and $\Upsilon_3 = 0.8$ are used. The weighting coefficients determine the influence of the constraint failure penalty terms compared with the other parts (i.e., joint latency and energy costs) in the reward signal. Their values are set for generating a uniform reward signal without any bias toward cost or penalty terms. Failure of service time latency constraint can be catastrophic, resulting in service failure. However, the failure of the other two constraints (i.e., sojourn time and energy constraints) can lead to additional costs without ensuring the service failure. Given that different constraints can have different influences, the values are set accordingly. For example, given the importance of service time constraint, Υ_1

TABLE 4.2: Simulation parameters

BS Coverage (d_n^M)	500 m
RSU Coverage (d_n)	10 m-40 m
Task Size ($D_{\rho m}$)	3 MB
Task Results ($D_{\rho m}^r$)	($D_{\rho m}$/5) MB
Required Task Latency (T_ρ)	2 s
VN Computation Cap. (η_m)	14 GFLOPS
RSU Computation Cap. (H_n)	25 GFLOPS
Task Proc. Requirements $\Omega_{\rho m}$	1250 FLOPS per bit
RSU Bandwidth (B_n)	35 MHz
VN Speed ($\vec{v}_m(\tau_i)$)	6 m/s-18 m/s
No. of Services ($\mathcal{S}$)	6
VN power (Pt_m, Pr_m)	(1.6 W, 1.4 W)

has a higher value than the other two coefficients (i.e., Υ_2 and Υ_3). These values are set empirically and can be optimized for having more precise results in the future. The learning rate (γ_{lr}), discount factor (Δ), and the epsilon value (e) for the epsilon greedy algorithm part of the Q learning simulation are 0.99, 0.7, and 0.5, respectively, while Λ is equal to 0.01 and $\mathcal{I} = 10^4$ is used. The primary and target neural networks have $L_d = 5$ fully connected layers with ReLU and linear activation functions (i.e., discrete actions). Other learning parameters include batch size $b = 32$ samples, and, a replay buffer of $r_b = 50000$ samples. For the heuristic approach $\lambda_{hu} = 0.2$ is considered.

In the following, the performance of the proposed approaches has been compared with a complete offloading procedure and with a local processing approach. Since utility minimization/maximization-based node selection strategies are often considered for computation offloading purposes [92, 98], in the following two benchmark methods are also considered, based on the communication distance between edge nodes and vehicles, and the available sojourn time. In addition a Single Agent approach has been considered for comparison.

- **Minimum Distance Based Offloading (MDBO)** In this method, each VN offloads its task to the nearest RSU [98], hence:

$$a_{m,n}(\tau_i) = 1 \Longleftrightarrow n = \underset{n' \in \mathcal{R}_m}{\operatorname{argmin}}\{d_{m,n'}(\tau_i)\}, \quad \forall m$$

  Though MDBO reduces the communication latency of the offloading process, it cannot guarantee optimal performance in terms of overall latency and energy consumption.

- **Maximum Sojourn Time Based Offloading (MSTBO)** We have considered a utility maximization-based benchmark with the sojourn time as the utility function while selecting the edge node [92]. In this method, VN selects the RSU node with maximum sojourn time for offloading it's data, hence:

$$a_{m,n}(\tau_i) = 1 \Longleftrightarrow n = \underset{n' \in \mathcal{R}_m}{\operatorname{argmax}}\{T^{soj}_{m,n'}(\tau_i)\} \quad \forall m$$

  In this method, VNs can reduce the overall handover requirements; however, RSU selection process does not consider the nearby VNs and corresponding RSU resource demands, and, as a result, optimal performance cannot be guaranteed.

- **Single Agent Based Q Learning Approaches (Q-SA and Q-DSA)** To compare the performance of the proposed Q-learning based methods, we have also considered a single agent-based Q-learning approach. Both traditional tabular Q-learning (Q-SA) and DQN-learning (Q-DSA) based methods are considered. In this case, a single RL agent without considering the surrounding environment parameters attempts to find the optimal policy for the network selection and computation offloading operations jointly. Since the agent is unaware of the total number of edge nodes able to provide the requested service,

Table 4.3: Complexity Analysis

Solution Approach	Computational Complexity
Q-SA	$\mathcal{O}(\mathcal{ST} \cdot \mathcal{AS} \cdot \mathcal{I})$ [99]
V2I-AC	$\mathcal{O}(\bar{N} \cdot \mathcal{ST} \cdot \mathcal{AS} \cdot \mathcal{I})$
V2V-AC	$\mathcal{O}(\bar{N} \cdot \bar{M} \cdot \mathcal{ST} \cdot \mathcal{AS} \cdot \mathcal{I})$
Q-DSA	$\mathcal{O}(\mathcal{I} \cdot b \cdot (n_0 n_l + \sum_{l=1}^{L_d-1} n_l n_{l+1})/\bar{\mathcal{I}})$
V2I-DAC	$\mathcal{O}(\bar{N} \cdot \mathcal{I} \cdot b \cdot (n_0 n_l + \sum_{l=1}^{L_d-1} n_l n_{l+1})/\bar{\mathcal{I}})$
V2V-DAC	$\mathcal{O}(\bar{N} \cdot \bar{M} \cdot \mathcal{I} \cdot b \cdot (n_0 n_l + \sum_{l=1}^{L_d-1} n_l n_{l+1})/\bar{\mathcal{I}})$

it uses the static network selection policy by selecting the nearest edge node. Also, the offloading process is performed without knowing the surrounding competing VNs.

Table 4.3 shows the computation complexity of the considered Q-learning based solutions. By following the analysis considered in [99], where the computational complexity for some basic Q-learning based algorithms is given, we extended the analysis to the proposed methods, where the computational complexity of proposed V2V and V2I-based approaches can be based on the total number of vehicular scenarios considered. For the V2I-AC approach, the complexity evaluation include the minimum number of edge node scenarios considered, i.e., $\bar{N}$, in addition to the state space $\mathcal{ST}$, action space $\mathcal{AS}$ and the maximum number of iterations during each learning round $\mathcal{I}$. In addition to this, in the V2V-AC approach, the complexity also depends upon the nearby VN scenarios, i.e., $\bar{M}$, and thus requires a higher number of computations compared with the V2I-AC approach. For the DQN-based approaches, the computation complexity is function of the number of layers (L_d), the number of neurons n_l, with $l \in L_d$, computation requirements for parameter updates, batch size, number of training episodes $\mathcal{I}$, total number of steps considered while updating the target DQN model weights $\bar{\mathcal{I}}$ [100, 101]. For the Q-DSA, V2I-DAC and V2V-DAC approaches, the computation complexity expressions are given in Table 4.3, where $n_0 = |\mathcal{ST}|$ represents the state space dimension.

Joint Latency and Energy Cost for task processing In Fig. 4.12, we present the average cost in terms of latency and energy requirements for VNs task processing. It is possible to notice that the proposed Q-learning-based approaches have reduced costs compared with the other benchmark methods including LS-HU. Both MDBO and MSTBO approaches perform the computation offloading operation without considering the surrounding environment parameters and the resource limitations of the edge servers. By selecting the nearest RSU, MDBO can reduce communication latency and energy costs. Though the MDBO approach can keep overall cost under control mainly due to the selection of the nearest edge node (i.e., reduced communication cost), as shown in Figs. 4.13 and 4.14, it suffers from a large number of failures due to the improper node selection and offloading. Additionally, with a growing number of VNs, MDBO performance weakens compared to the Q-learning based approaches. On the other hand, though the MSTBO approach selects the edge node with the highest sojourn time, the overall cost is higher mainly due to the reduced flexibility of the overall partial offloading process. Single agent-based Q-learning approaches (i.e., Q-SA and Q-DSA) without a proper knowledge of the VNs local environment also have a limited performance with growing VN density.

With better knowledge of the surrounding environment, Q-learning based approaches can jointly select the proper edge node and the amount to be offloaded. Though the V2I-AC approach is able to adapt the Q-learning policies (and perform better compared with benchmark methods) according to the varying number of edge nodes, without having the proper knowledge of the surrounding competing VUs, its performance is slightly worse compared with the V2V-AC approach. By considering both the number of edge nodes and the surrounding competing VNs information, the V2V-AC approach is able to perform the computation offloading operation with superior performance. In particular, for the case of 1400 VNs, a 30.3% performance

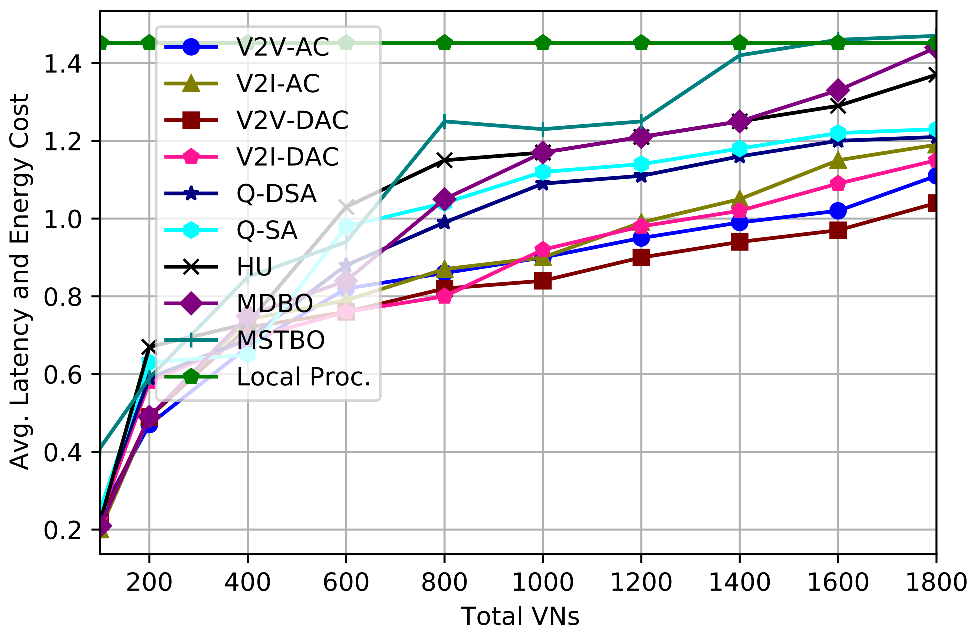


FIGURE 4.12: Average Joint Latency and Energy Cost for variable number of VNs.

gain in terms of reduced cost can be observed for the V2V-AC approach compared with the MSTBO method. The DQN approaches, especially V2V-DAC, allows to achieve a performance gain and is able to handle more complex scenarios. Therefore, in the considered multi-service based vehicular scenario, the proposed schemes can serve VNs with innovative services having better latency and energy performance; it is in particular important to stress that the collaborative approach by itself is able to increase the performance, even when implemented with a tabular Q-learning approach.

RSU handover required during computation offloading In a VEC environment, each VN should perform the offloading process before it crosses through the coverage range of the RSU node. The offloading process is composed by the computation data offloading towards the RSU server and the reception back of the results.

In case a VN is not able to finish the offloading process before going out the coverage of the selected RSU an additional cost in terms of handover latency should be considered.

In Fig. 4.13, the amount of handover requests is considered, mapping the amount of times each VN is not able to complete an offloading request by the sojourn time. It is possible to notice that they increase with the increased number of VNs in the service area, and that the performance with respect to the benchmark is better. Though both MDBO and MSTBO approaches select the edge nodes to maximize the particular utility (performance in terms of communication latency/available sojourn time), they fail to adapt according to the varying VNs demands. Without properly distributing the VNs requests and improper offloading amounts they fail to adapt the sojourn time bounds of RSU nodes resulting in higher failures. The LS-HU approach with a more flexible node selection and the offloading process can have a better performance compared to the other benchmark approaches. On the other hand, by properly utilizing the parameters of the surrounding environment through V2I and V2V, proposed Q learning-based approaches are able to reduce the overall number of failures. It is also possible to see that the Q-SA approach has a higher number of failures compared with both V2I-AC and V2V-AC approaches highlighting the importance of the proposed vehicular communication-based learning framework. Moreover, it is clear the advantage of the V2V approaches with respect to the V2I information sharing, while the DQN solutions allows to slightly increase the performance with respect to the tabular Q-learning approaches.

Number of VNs failing to satisfy the service time constraint Each VN should complete the task processing operation within a requested service latency

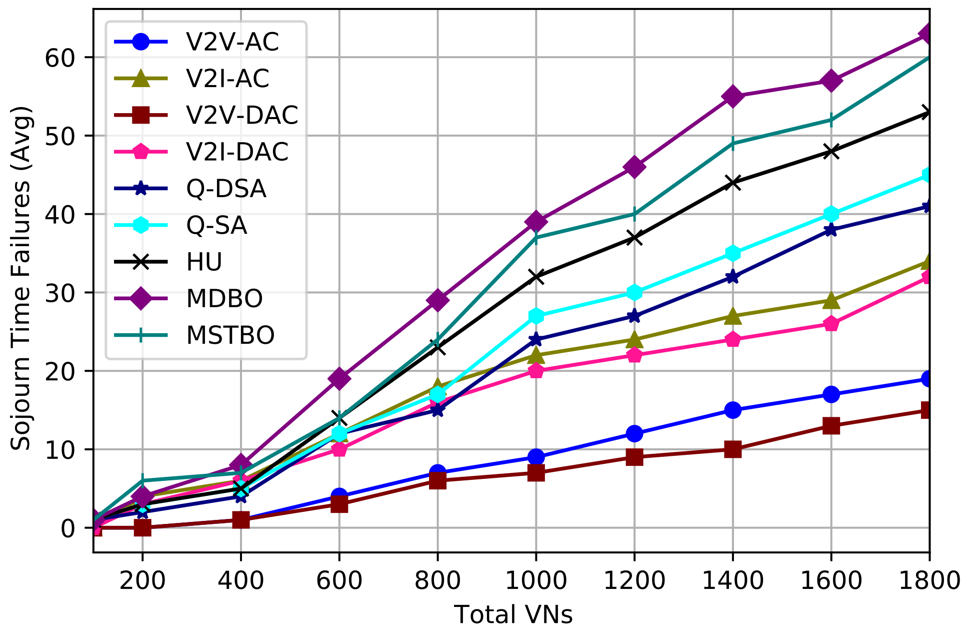


Figure 4.13: Average Number of RSUs Handover Requests for variable number of VNs.

bound failure which can reduce the QoS. In Fig. 4.14, we present the average number of VNs failing to satisfy the service latency bound. Both MDBO and MSTBO approaches are failing to perform the task processing within the service time requirement. This shows that performing computation offloading operation without considering the surrounding environment results in higher failures. On the other hand, the proposed Q-learning-based approaches, especially the V2V-AC method, exploiting various environmental parameters, are able to select the proper edge node and amount to be offloaded jointly, resulting in superior performance compared with the other benchmark schemes. The DQN approaches allows to slightly increase the performance with respect to the tabular Q-learning approaches. Additionally, the proposed VNs scenario-based Q-learning approaches outperform the traditional single agent based approaches i.e., Q-SA and Q-DSA, in terms of a number of failures. Thus, the proposed Q-learning and DQN approaches can be useful for enabling latency-critical services over VNs.

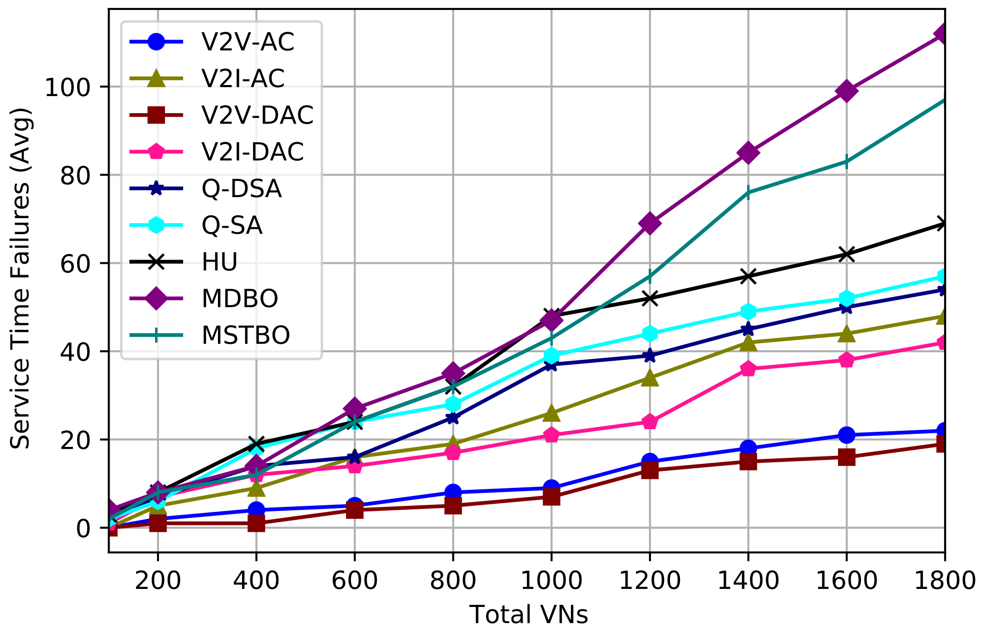


FIGURE 4.14: Average number of Service Time Failures for variable number of VNs.

Task Completion Latency Total task completion latency is a function of computation offloading and the local device computation times. Here, we compare the average task completion latency of different schemes when changing the vehicles density in the service area. Fig. Fig:LC provides the average task computation latency required by each solution method. In a given service area, the total latency required for processing the tasks with Q-learning approaches is much smaller than the benchmarks. Both Q-learning and DQN approaches are able to select proper edge nodes and the amount of data to be offloaded towards them that results in a better performance in terms of overall latency requirements. By analyzing the vehicular environment data through the V2V and V2I technologies, the Q-learning and DQN approaches are able to reduce the overall task processing latency. The traditional single agent-based approaches, i.e., Q-SA and Q-DSA, require higher latency costs, mainly due to the improper node selections and offloading process. On the other hand, the latency performance of the benchmark methods increases rapidly,

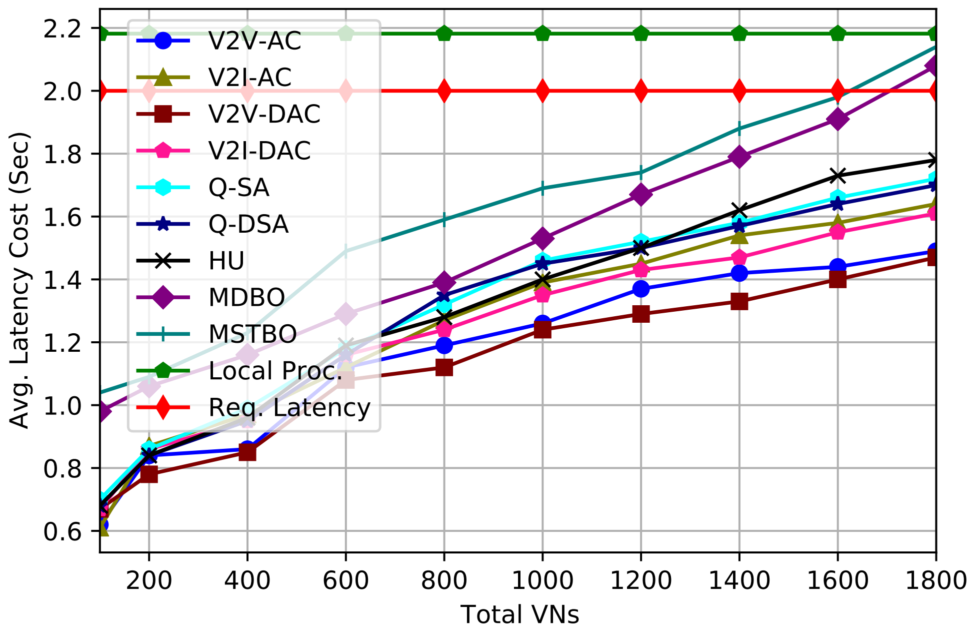


Figure 4.15: Average Task Completion Latency for variable number of VNs.

which can be disastrous for latency-critical services. Though the MDBO approach is using the nearest edge node for offloading its data, without properly assessing the resource availability, the number of edge nodes, and the competing VNs, it fails to adapt to the increasing VNs demands. Similarly, the MSTBO approach also fails to adapt its network selection and computation offloading policies according to the VUs' demands resulting in higher latency costs. The LS-HU approach can reduce the latency cost by selecting the nodes with proper distance measures and offloading adequate data compare to the other benchmark methods. This result shows the high potential of proposed schemes for enabling latency-critical applications and services.

Average Energy Consumption Fig. 4.16 shows the average amount of energy consumed by VNs for the complete task processing. The total energy consumed includes the energy required for the local device computation, transmission, and reception of tasks towards and from RSUs. It has to be noticed that the total

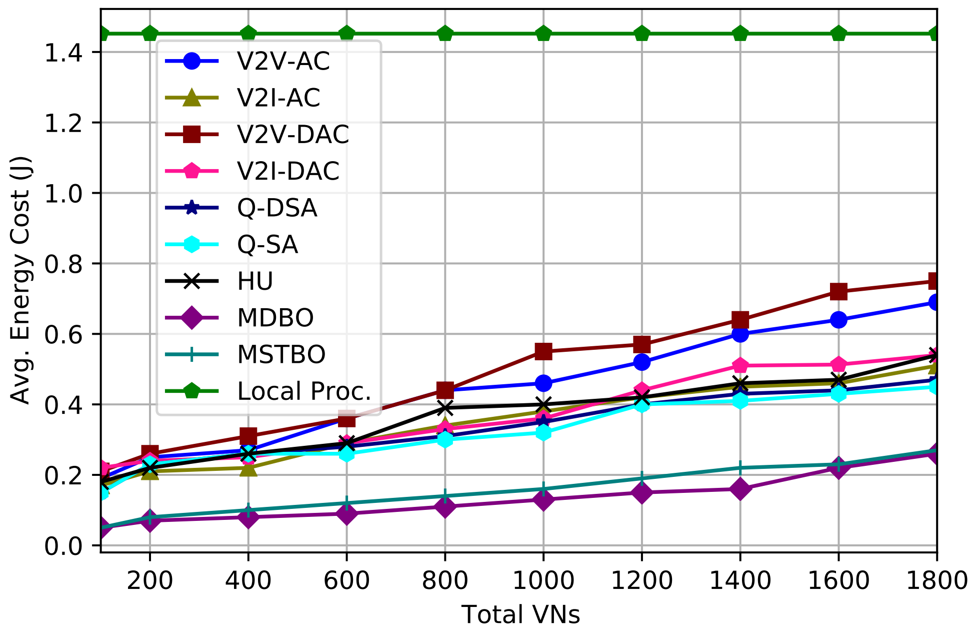


FIGURE 4.16: Average Energy Consumption for Task Completion for variable number of VNs.

energy required for locally processing the vehicular tasks is relatively higher than the offloading process energy. Therefore, with a complete offloading process, the benchmark methods have relatively better energy performance compared with the proposed Q-learning and DQN approaches. On the other hand, Q-learning methods require higher energy mainly because of the local processing energy part, where the DQN approaches consume slightly more than the tabular Q-learning approach. However, the overall performance in terms of joint latency and energy requirements (as shown in Fig. 4.12), considering also the handover and latency failures is much better than the benchmarks.

Average Computation Offloading We analyze then the impact of nearby VNs on the computation offloading amount. In Table 4.4, the average percentage of data offloaded by VNs towards RSUs is shown. Since the available resources at each RSU are limited, if the number of vehicles increases, the V2V assisted collaborative

Table 4.4: Average Percentage of Data Offloading.

# VNs	100	200	400	600	800	1000	1200	1400	1600	1800
V2I-AC	.79	.80	.82	.64	.65	.57	.58	.59	.60	.60
V2V-AC	.80	.78	.77	.60	.59	.50	.48	.47	.45	.45
V2I-DAC	.79	.80	.80	.62	.66	.58	.59	.57	.56	.56
V2V-DAC	.82	.75	.75	.58	.54	.50	.49	.45	.43	.42

Q-learning approach offloads the lowest amount of data towards RSU servers for avoiding handovers. On the other hand, the V2I approach does not take into account the other VNs, and, as a result, a higher offloading percentage and more handover requests are set.

Training Iterations in Terms of Service Failures For analyzing the training performance of the proposed Q learning-based solutions in Fig. 4.17, we show the number of VNs failing to perform the task processing in a limited time. For a given set of VNs with reduced training iterations, Q-learning approaches have a higher number of failures, however, with increasing training iterations Q-learning approaches, especially the V2V-AC approach outperforms the other solutions. It has to be clarified that DQN-based training iterations can last for longer time compared to the tabular approach, despite a reduced number of overall iterations is generally needed. Hence, a comparison with the tabular method is not fair, preventing their representation in figure.

4.3.5 Conclusion

In this work, we have considered a joint RSU selection and computation offloading problem in a multi-service multi-user VN. We have modeled it as a RL-based problem and solved it by using Q-learning methods. Two collaborative learning-based approaches where the Q-agents learn the optimal policy exploiting the V2I and V2V communication paradigms are considered. Along with the traditional tabular

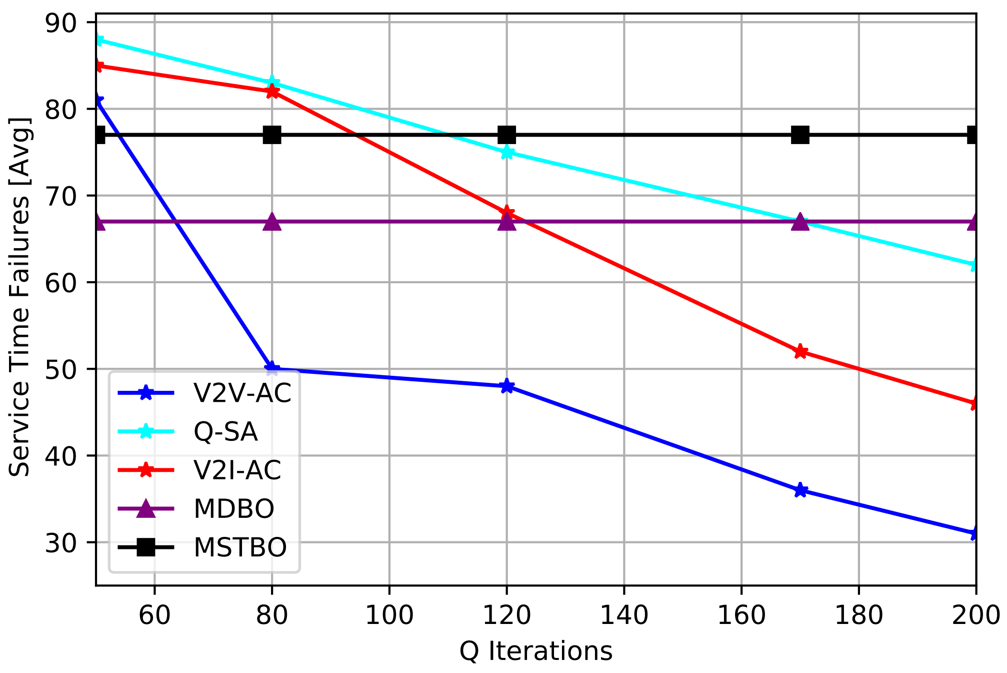


FIGURE 4.17: Q-Learning Training Performance in Terms of a Service Time Failures for a variable number of iterations.

method, a DQN approach is also considered for estimating the Q values, allowing to handle more complex learning scenarios. Compared with other benchmark methods, the proposed schemes provide better network-wide performance in terms of latency and consumed energy. Thus, proposed schemes can have a great potential for enabling latency-critical applications and services over VNs. This work highlights the importance of enabling collaborative RL strategies, exploiting vehicular communication modes, for solving the joint network selection and the offloading problem over a multi-service VN; this is also strengthen by the fact that the collaboration among vehicular nodes has an impact on the performance higher than the implementation of DQN with respect to the tabular Q-learning.

As a future direction, the offloading process costs can further be reduced by allowing VNs to select more than one edge node for computation offloading. However, such an approach can add extra dimensions of complexity, and more rigorous solution methods. It can also be interesting to use a multi-task learning-based approach (i.e. multi task RL/FL) with additional vehicular service parameters. By considering the distributed nature of the VNs and corresponding data, other decentralized learning methods such as FL can also be a potential solution method.

Chapter 5

Distributed Data Processing for IoV -Joint-Terrestrial and Non-Terrestrial Case

Some content of this chapter is based on the following articles [78];

1) " Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Network Selection and Computation Offloading in Non-Terrestrial Network Edge Computing Environments for Vehicular Applications." In 2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC), pp. 1-8. IEEE, 2022.".

2) " Shinde, Swapnil Sadashiv, and Daniele Tarchi. " Multi-Time Scale Markov Decision Process for Joint Service Placement, Network Selection, and Computation Offloading in Aerial IoV Scenarios." Submitted to IEEE Transactions on Network Science and Engineering (2023).".

3) " Shinde, Swapnil Sadashiv, and Daniele Tarchi. " A Multi-level Sequential Decision Making Process for Non-Terrestrial Vehicular Edge Computing Environments." Submitted to IEEE Transactions on Communications (2023).".

5.1 Introduction

Non-terrestrial Networks (NTNs) are considered a key enabler for the upcoming 6G technology, and are expected to play an important role in boosting the capacity and coverage of traditional terrestrial networks [102]. Therefore, in recent times, many new networking platforms are populating the space. Based on their distance from the Earth's surface, these platforms can be classified into aerial and space-based networking technologies. Aerial platforms include Low Altitude Platforms (LAPs), such as unmanned aerial vehicles (UAVs), air taxis, and helicopters, and High Altitude Platforms (HAPs), including airships, balloons, aircraft, etc. On the other side, various satellite constellations can compose the space network, such as Low Earth Orbit (LEO), Medium Earth Orbit (MEO), and Geostationary Orbit (GEO) satellites. Aerial and space onboard computation and communication technologies can be exploited for potential EC services, creating a multi-layer EC environment. Additionally, NTN platforms can also act as relay nodes, to route the VUs computation load toward ground cloud facilities to serve the users having limited or zero connectivity towards terrestrial networks. Therefore, integration of NTN platforms into the current VEC system may be useful to serve Vehicular Users (VUs) and their growing needs for new services [6, 103].

In the case of multi-service vehicular environments, VUs can request different services from EC nodes and offload a portion of their tasks to the selected nodes through partial offloading processes. The above-mentioned EC facilities can have different characteristics in terms of mobility, node density, size, distance from VUs, etc. Therefore selecting a proper EC layer is important. Moreover, the size and energy restrictions often limit the ability of the EC nodes placed on different networking layers. Due to these limitations, each node can provide only a limited set of services that can be exploited by VUs. Therefore, selecting a particular EC node for offloading VUs service data is an important problem to be solved. Furthermore, due to the limited computing capabilities of each node, offloading a proper amount towards the selected Edge Node (EN) can improve performance and service quality [104]. These problems can be defined as a joint network selection and computation offloading problem and solving such complex problems over a multi-EC, multi-service Vehicular Network (VN) can be extremely challenging.

5.2 Joint Network Selection and Offloading with joint T-NTN Vehicular Scenario - Metaheuristic Solution

The traditional terrestrial network-based EC facilities, usually referred to as Vehicular Edge Computing (VEC), enabled through the Road Roadside Units (RSU) deployments, have limited resources, higher deployment costs, limited coverage, and can rapidly become a bottleneck for the VNs performance. On the other hand, various Non-terrestrial Networking (NTN) platforms from air and space networks have gained a lot of attention in 5G and beyond studies and are expected to play a key role in the upcoming days. Integration of NTN-based EC facilities into the current VEC system can be useful for serving VUs with different service types. Therefore, we design a multi-EC-enabled vehicular networking platform for serving VUs (VUs) with a heterogeneous set of services. We model the various latency and energy requirements for processing the VU task requests through partial computation offloading operations. We further aim to minimize the overall latency and energy requirements for processing the VUs data by selecting the proper ENs and the offloading amounts over a multi-EC-enabled VN. We have modeled the problem into the framework of an evolutionary Genetic Algorithm (GA) able to find the proper EN selections and the offloading amounts. The results are compared with some benchmark solutions to show the effectiveness of the proposed approach.

The main contributions of this work are:

- We propose a multi-EC facility-enabled T-NT network for promoting VUs with various services. Ground-based VUs can request different services offered by the RSUs, LAP, HAP, and LEO satellites through the EC facilities onboard.
- A proper mathematical model is done by including the latency and energy requirements of the different steps involved during the VUs task processing operation. In the end, an optimization problem is developed for minimizing the overall latency and energy consumption by selecting a proper EC facility and the amount to be offloaded.
- The joint network selection and computation offloading problem is solved by using the adaptive genetic algorithm (A-GA) that takes into account the available EN resources while selecting the offloading process parameters.
- In addition, a set of benchmark methods are used for analyzing the results, which show the improved latency and the energy performance of a proposed scheme.
- Section 5.2.4 concludes the work with proper remarks.

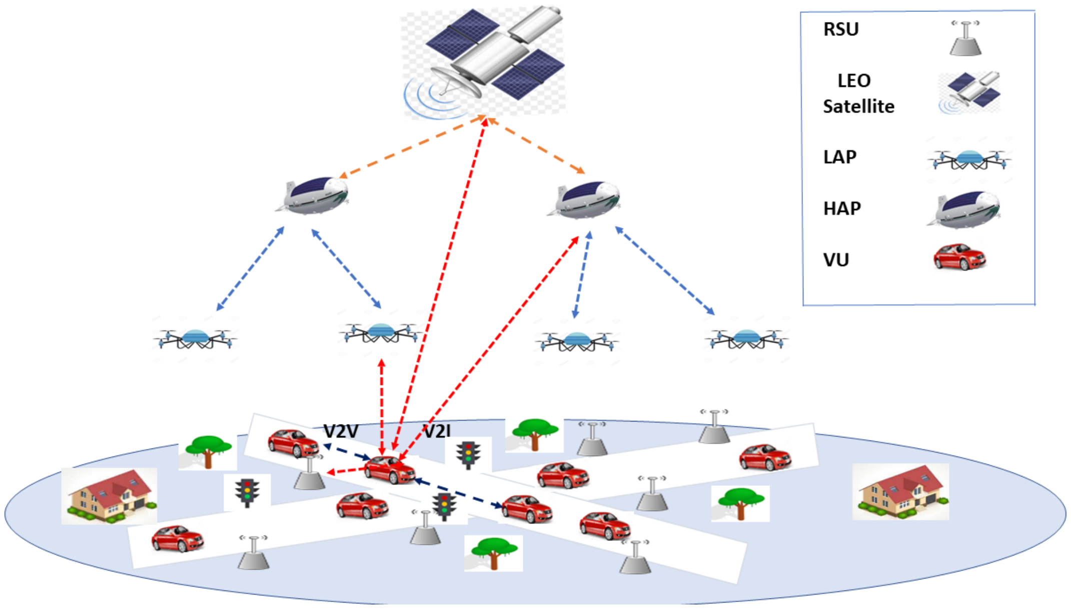


FIGURE 5.1: System Model

5.2.1 System Model and Problem Formulation

We consider an integrated T-NT network composed of a LEO satellite constellation $\mathcal{S} = \{s_1, \ldots, s_k, \ldots, s_K\}$ with K satellites, a set $\mathcal{H} = \{h_1, \ldots, h_p, \ldots, h_P\}$ of P HAPs, a set $\mathcal{L} = \{l_1, \ldots, l_u, \ldots, l_U\}$ of U LAPs, a set $\mathcal{R} = \{r_1, \ldots, r_n, \ldots, r_N\}$ of N RSUs, and a set of $\mathcal{V} = \{v_1, \ldots, v_m, \ldots, v_M\}$ of M VUs, located randomly in the area, supposed to be modeled as a multi-layered EC enabled vehicular networking scenario. LEO satellites, HAPs, UAVs and RSUs are equipped with EC facilities that can be exploited by the VUs for different applications and services. The VN is modeled as a time-discrete system where the network parameters are supposed to be constant in each time interval τ, where τ_i identifies the ith time interval, i.e., $\tau_i = \{\forall t | t \in [i\tau, (i+1)\tau]\}$. In the following we consider to focus on a simplified scenario where at each time instant only one LEO satellite is under visibility, while the whole constellation can be reached through proper Inter-satellite links. Fig. 5.1 shows the various elements of the considered VN scenario composed by the one reference LEO satellite, HAPs, LAPs, RSUs and VUs.

The generic mth VU is characterized by a processing capability equal to $c_{v,m}$ Floating Point Operations per Second (FLOPS) per CPU cycle, while its CPU frequency is $f_{v,m}$. Each VU is supposed to be able to communicate using a bandwidth $B^{\mathrm{e}}_{v,m}$ with a reference EN $e \in \{\mathcal{S} \cup \mathcal{H} \cup \mathcal{L} \cup \mathcal{R}\}$. At each interval, the mth VU is supposed to generate a task request x_m to be processed, where the task x_m is identified through the tuple $\langle D_{x_m}, \Omega_{x_m}, \bar{T}_{x_m}, \bar{\Xi}_{x_m} \rangle$ where D_{x_m} is the task size in Byte, Ω_{x_m} are the requested CPU execution cycles, $\bar{T}_{x_m}$ is the maximum latency of the requested service and $\bar{\Xi}_{x_m}$ is the service type requested by the VU.

The eth EN (i.e., one node among any RSU, LAP, HAP, or LEO)[1] is characterized by a processing capability equal to c_e FLOPS per CPU cycle, with CPU frequency f_e, and communication capabilities, supposed to be identified through a communication technology able to work on a bandwidth B_e and covering an area with radius R_e. The LAP, HAP and satellite nodes are located at height h_l, h_h and h_s from the ground level, respectively. Each EN provides computation offloading services to the VUs within its coverage area. VUs tasks are characterized by the service type, supposing the system able to provide different services, where $\Xi = \{\xi_1, \dots, \xi_z, \dots, \xi_Z\}$ is the set of possible services and Z the maximum number of services. With limited storage capabilities, RSUs, LAPs and HAPs can provide a limited number of services. We consider that the eth EN can provide $\Xi^e = \{\xi_1, \cdots \xi_z, \cdots, \xi_e^{max}\}$. Additionally, the LEO constellation is able to provide all the possible services requested by VUs. Fig. 5.2 shows the multi-service vehicular scenario where RSUs, LAPs and HAPs are providing a subset of services (i.e., 2 and 3, respectively) while the LEO node is able to provide all possible services (i.e., 6) requested by the VUs.

[1]In the following, $e = \{r, n\}$, $e = \{l, u\}$, $e = \{h, p\}$, and $e = \{s, k\}$ stands for the rth RSU, uth LAP, pth HAP and kth LEO satellite.

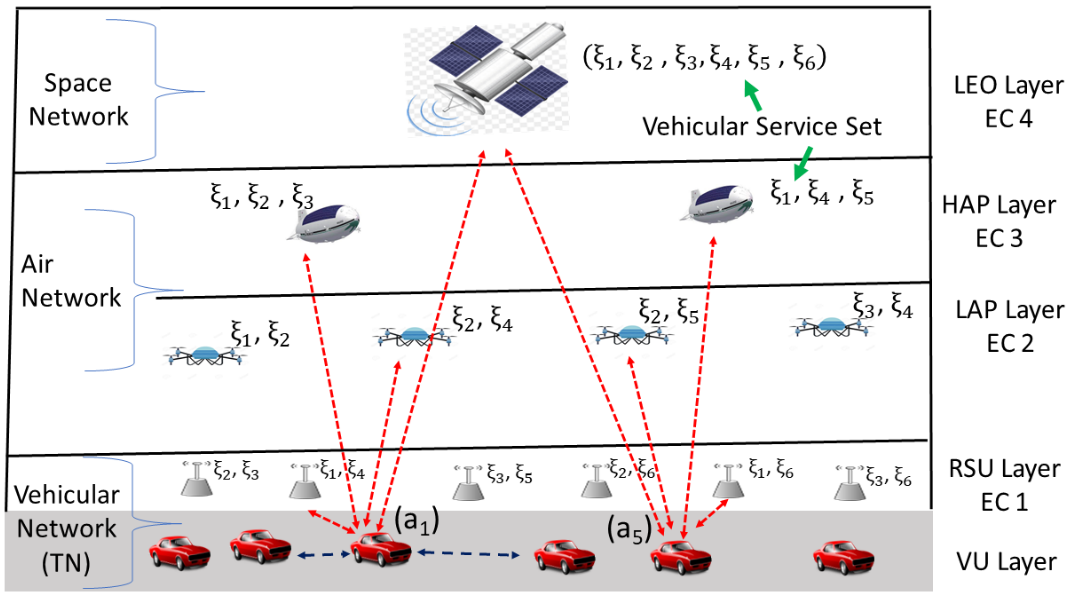


FIGURE 5.2: Multi Service VN with Non-terrestrial EC Layers

5.2.1.1 VUs Mobility and Distance Measures

In this work, we consider that the VUs are moving with a variable speed $\vec{v}_m(\tau_i)$, whose value is bounded within $\vec{v}_{\min}$ and $\vec{v}_{\max}$ [105], while the the mth VUs instantaneous speed is modeled through a truncated normal distribution density function:

$$f\left(\vec{v}_m(\tau_i)\right) = \begin{cases} \dfrac{2e^{\frac{-(\vec{v}_m(\tau_i)-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}\left(\operatorname{erf}\left(\frac{\vec{v}_{\max}-\mu}{\sigma\sqrt{2}}\right)-\operatorname{erf}\left(\frac{\vec{v}_{\min}-\mu}{\sigma\sqrt{2}}\right)\right)}, \\ \qquad\qquad \vec{v}_{\min} \le \vec{v}_m(\tau_i) \le \vec{v}_{\max} \\ 0, \quad \text{else} \end{cases} \tag{5.1}$$

where μ and σ are the mean and standard deviation of the vehicles speed, and $\operatorname{erf}(x)$ is the Gauss error function over x. The path length within which the mth VU remains under the coverage of any terrestrial and aerial eth node (i.e., RSU, UAV, HAP) is $D_{v_m,e}(\tau_i)$ and can be given by:

$$D_{v_m,e}(\tau_i) = \sqrt{d_e^2 - \left(y_e - y_{v_m}(\tau_i)\right)^2} \pm \left(x_e - x_{v_m}(\tau_i)\right) \tag{5.2}$$

where, $\left(x_{v_m}(\tau_i), y_{v_m}(\tau_i)\right)$ is the location of the mth VU at τ_i and (x_e, y_e) is the projection over the ground of a generic eth EC node, which can be RSU, UAV, HAP. The available sojourn time for the mth VU with respect to a generic eth node (RSU, LAP or HAP) can be written as:

$$T^{soj}_{v_m,e}(\tau_i) = \frac{D_{v_m,e}(\tau_i)}{|\vec{v}_m(\tau_i)|} \qquad \forall i, v_m \tag{5.3}$$

5.2.1.2 LEO Satellite Mobility and Distance Measures

In general, VUs can communicate with satellite nodes for a particular time interval, which depends on the locations and the mobility patterns of LEO satellites. LEO satellites can move at a very high speed (a few Kms per second) compared with the VUs, which move at a few meters per second. Here we consider a well-known coverage model for finding the arc length over which ground-based VUs can communicate.

Let us consider a LEO satellite located at height h_s with respect to the ground, moving with a constant speed v_s km/s. Les us consider that at a certain instant τ_i its elevation angle with respect to the mth VU is $\theta_m(\tau_i)$, corresponding to a geocentric angle for the mth VU as [106],

$$\delta_{s,m}(\tau_i) = \arccos\left(\frac{R_e}{R_e + h_s} \cdot \cos\left(\theta_m(\tau_i)\right)\right) - \theta_m(\tau_i)$$

where, R_e is the Earth radius. The total arc length over which VU can communicate with the considered LEO satellite is defined as,

$$L_{s,m}(\tau_i) = 2(R_e + h_s) \cdot \delta_{s,m}(\tau_i)$$

With this the total time for which the mth VU can be in the coverage range of the LEO is given as,

$$T^{soj}_{v_m,s}(\tau_i) = \frac{L_{s,m}(\tau_i)}{v_s} \tag{5.4}$$

The mth VU should complete the offloading process towards the LEO satellite during the limited coverage time, for avoiding the additional costs in terms of LEO satellite handovers. Since the satellite nodes have much higher speed than the VUs, the VUs mobility is neglected while defining the satellite coverage space. Eq. (5.3) and (5.4) defines the mobility constraint for the considered multi-EC vehicular networking system.

5.2.1.3 Network Selection and Task Offloading Process

For each ith interval, the mth VU task is supposed to be managed through a partial computation offloading process allowing a portion of the task to be offloaded towards the selected EN while the rest can be locally processed, hence, allowing to reduce the overall processing time and energy cost [76]. Here, we consider $0 \leq \alpha_{x_m}(\tau_i) \leq 1$ as an offloading index associated with the mth VU representing the portion of the task offloaded towards the selected EN, where $\alpha_{x_m}(\tau_i) = 1$ corresponds to a complete offloading, while $\alpha_{x_m}(\tau_i) = 0$ corresponds to perform only local processing. To avoid additional complexity, we assume that each VU can offload only towards one EN. We define a network selection variable, $a_{(m,e)}(\tau_i)$, indicating that the mth VU has selected at τ_i the eth EN for the task offloading, where

$$\sum\nolimits_e a_{(m,e)}(\tau_i) \leq 1, \quad \forall i \tag{5.5}$$

corresponding to say that the mth VU can select at most one of the possible EN under visibility. The vehicular task processing operation with partial computation offloading involves several steps such as the transmission of a selected task portion towards a selected EN, task computation operation at EN, reception of task back at VU, local computation of remaining task, etc. Here we present the generic task computation and communication models in the vehicular scenario, which are later used to model the overall task computation latency and energy.

Computation Model

The time and energy required for task computation on any lth device for a generic task x_m can be written as:

$$T_{c,l}^{x_m} = \frac{\Omega_{x_m}}{c_l^{x_m} f_l^{x_m}}, \quad E_{c,l}^{x_m} = T_{c,l}^{x_m} P_{c,l} \tag{5.6}$$

where $c_l^{x_m}$ and $f_l^{x_m}$ are the number of FLOPS per CPU-cycle and CPU-frequency allocated to the task x_m, respectively, whether l identifies a VU (v_m), a RSU (r_n), a LAP (l_u), a HAP (h_p) or a LEO satellite (s_k). We have considered that the nodes capacity will be equally shared by all the tasks allocated to it. Also, in (5.6), $P_{c,l}$ is the computation power used by the generic lth device.

Communication Model

The partial computation offloading process consider the possibility to split the tasks into sub-portions and offload only part of the task, while the rest is locally processed at the originating device; this means that both transmission of the VUs data portion towards the selected EN and the reception back of the results from EN should be considered. The total time and energy consumed during the transmission of data from a generic node k to another node l for task x_k are given by[2]:

$$T^{\rho_k}_{tx,kl}(\tau_i) = \frac{D_{x_k}}{r_{kl}(\tau_i)}, \quad E^{x_k}_{tx,kl}(\tau_i) = T^{x_k}_{tx,kl}(\tau_i) Pt_k \tag{5.7}$$

where $r_{kl}(\tau_i)$ is data-rate of the link between the two nodes[3], and Pt_k is the transmission power of kth node. Similarly, the time and energy required to receive the task of size $D^r_{x_k}$ from lth EN to k are:

$$T^{x_k}_{rx,lk}(\tau_i) = \frac{D^r_{x_k}}{r_{kl}(\tau_i)}, \quad E^{x_k}_{rx,lk}(\tau_i) = T^{x_k}_{rx,lk}(\tau_i) Pr_k \tag{5.8}$$

where Pr_k is the power consumed for receiving data. Additionally, a symmetric channel model is assumed between k and l.

Task Offloading Process During the partial computation offloading process, if the mth VU is assigned to the eth EN, the time and energy required to offload the task to e and to get back the result at v_m in the ith interval are:

$$T^{off}_{m,e}(\alpha_{x_m}(\tau_i)) = \alpha_{x_m}(\tau_i) \left(T^{x_m}_{tx,me} + T^{x_m}_{c,e} + T^{x_m}_{rx,em} \right) \tag{5.9a}$$

[2]Here l and k are the indexes of any generic node among v_m, r_n, l_u, h_p, and s_k.

[3]The expression for the channel transmission rate is based on the the Shannon capacity formula, properly adapted to the different channel models (i.e., vehicular, LAP, HAP, and LEO channel models). We avoid to report them here due to space constraints. The interested reader could refer to [76, 106]

$$E_{m,e}^{off}(\alpha_{x_m}(\tau_i)) = \alpha_{x_m}(\tau_i)\left(E_{tx,me}^{x_m} + E_{rx,em}^{x_m}\right) \tag{5.9b}$$

where $T_{tx,me}^{x_m}$, $T_{c,e}^{x_m}$, and $T_{rx,em}^{x_m}$ are the transmission time, computation time on eth EN and the receiving time for the task x_m generated by v_m during offloading phase, and $E_{tx,me}^{x_m}$ and $E_{rx,em}^{x_m}$ are the energy consumed during the task transmission and result collection phases on device. For limiting the complexity, we have neglected the energy consumed by the EN for task processing.

Local Computation The amount of time and energy required for computing the task locally in the ith interval is:

$$T_m^{loc}(\alpha_{x_m}(\tau_i)) = \left(1 - \alpha_{x_m}(\tau_i)\right) T_{c,m}^{x_m} \tag{5.10a}$$

$$E_m^{loc}(\alpha_{x_m}(\tau_i)) = \left(1 - \alpha_{x_m}(\tau_i)\right) E_{c,m}^{x_m} \tag{5.10b}$$

where $T_{c,m}^{x_m}$ and $E_{c,m}^{x_m}$ are the time and energy spent for the whole task x_m local processing, while $\alpha_{x_m}(\tau_i)$ is the portion of the task locally processed at the time interval τ_i.

Partial offloading Computation The delay and the energy consumed during the task processing phases, when partial offloading is performed in the ith interval, can be written as:

$$\begin{aligned} T_m^{x_m}(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)) &= \max\left\{T_{m,e}^{off}(\alpha_{x_m}(\tau_i)), T_m^{loc}(\alpha_{x_m}(\tau_i))\right\} \\ E_m^{x_m}(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)) &= E_{m,e}^{off}(\alpha_{x_m}(\tau_i)) + E_m^{loc}(\alpha_{x_m}(\tau_i)). \end{aligned}$$

In addition, each vehicle should finish the offloading process and receive the results back within the sojourn time, hence:

$$T_{m,e}^{off}(\alpha_{x_m}(\tau_i)) \leq T_{v_m,e}^{soj}(\tau_i) \quad \forall i \tag{5.11}$$

5.2.1.4 Problem Formulation

The main aim of this work is to optimize the network-wide performance of the multi-EC enabled VN. We aim to optimize the performance in terms of overall latency and energy consumed during the offloading process towards edge servers by selecting proper EN and offloading amounts. For this, we formulate the joint latency and energy minimization problem as:

$$\mathbf{P1}: \min_{\mathcal{A},\mathbf{A}} \left\{ \frac{1}{M} \sum_{m=1}^{M} \left[\gamma_1 T_m^{x_m} \left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right) + \gamma_2 E_m^{x_m} \left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right) \right] \right\} \forall i \tag{5.12}$$

s.t.

$$\mathbf{C1}: \text{Eq. } (5.5) \tag{5.13}$$

$$\mathbf{C2}: a_{m,e}(\tau_i) = 1 \iff \Xi_{x_m} \in \Xi_e \quad \forall i \tag{5.14}$$

$$\mathbf{C3}: T_m^{x_m}\left(\alpha_{x_m}(\tau_i)\right) \leq T_{x_m} \quad \forall v_m \in \mathcal{V}, \ \forall i \tag{5.15}$$

$$\mathbf{C4}: \text{Eq. } (5.11) \tag{5.16}$$

$$\mathbf{C5}: E_m^{x_m}(\alpha_{x_m}(\tau_i)) \leq E_{c,m}^{x_m} \tag{5.17}$$

$$\mathbf{C6}: 0 \leq \gamma_1, \gamma_2 \leq 1; \ \gamma_1 + \gamma_2 = 1 \tag{5.18}$$

where $\mathbf{A} = \{\alpha_{x_m}\}^M$ is the computation offloading matrix, and γ_1, γ_2 are two weighting coefficients for balancing latency and energy consumption. **C1** stands that each VU can select at most one EN for the computation offloading. According to **C2**, the selected EN must be able to provide the service requested by the VUs. **C3** puts a limit on the maximum processing time as one of the task requirements. According to **C4**, for avoiding handover phenomena and related latency, each VN should complete the offloading process before it passes through the selected EN coverage. According to **C5**, the total energy required for the task processing during the partial computation offloading process should be bounded by the energy needed to process the complete task locally. **C6** stands that the two weighting coefficients (γ_1, γ_2) should be between 0 and 1 with their sum equal to 1.

5.2.2 Proposed Solutions

We aim to minimize the VUs latency and energy cost during the vehicular task processing operation over the multi-service multi-EC enabled VN. We aim to find the proper EN selection and offload the correct amount of data toward the selected EN to optimize the performance. Given the complex nature of the problem, we have considered an evolutionary search-based metaheuristic approach for solving it. In particular, we propose to use a genetic algorithm (GA) for selecting the ENs and the offloading amount for the VUs tasks. We first introduce the GA by highlighting the main steps involved during the search process. After that, we define the main GA elements for the considered problem.

The GA process is an adaptive search-based optimization technique that is inspired by the theory of natural selection and genetics. It can effectively solve both constrained and non-constrained optimization problems in a complex domain such as VNs. GA process begins with the initial definition of population space $(\mathcal{PS})$ containing a set of possible solutions (i.e., individuals) defined as a chromosome $(\mathcal{C})$.

GA process is an iterative process where at each iteration new $\mathcal{PS}$ having better individuals is formed. Each iteration involves the analysis of the current $\mathcal{PS}$ through a fitness function ($\mathcal{FF}$), the selection of a parent C through a selection function (S_f), then the formation of new individuals by using mutation and crossover as a fundamental GA operators. The mutation process involves the creation of new C from a selected solution form ($\mathcal{PS}$), by altering a set of genes. On the other hand, through the crossover process, two chromosome sets with good fitness function constitute a C for the next generation by combining their genes. Each evaluation creates a better solution set and finally ends by providing a solution point with a higher fitness value. More comprehensive information on GA and evolutionary algorithms can be found in [107], while here we focus on the main elements for the sake of brevity.

Chromosome

For the considered joint *network selection and offloading* problem, the chromosome C is constituted by a set of ENs available for the selection and the offloading amount. A binary sequence of b bits is adapted for defining the offloading amounts allowing a binary chromosome C. For example, with $b = 3$, a set of three bits defines the offloading amount selected. In such a case, $[0, 0, 0]$ indicates the 0% offloading while $[1, 1, 1]$ indicate the possibility of 100% offloading, where $1/(2^b)$ provides the step change for the offloading parameter. Fig. 5.3 shows the example chromosome vector with T ENs and b bits representing the offloading parameter.

Fitness Function The $\mathcal{FF}$ is defined by resorting to the objective function and the constraint failure penalties and is given by,

$$\mathcal{FF}\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right) = \Big[\gamma_1 T_m^{x_m}\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right) + \gamma_2 E_m^{x_m}\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right)\Big]$$

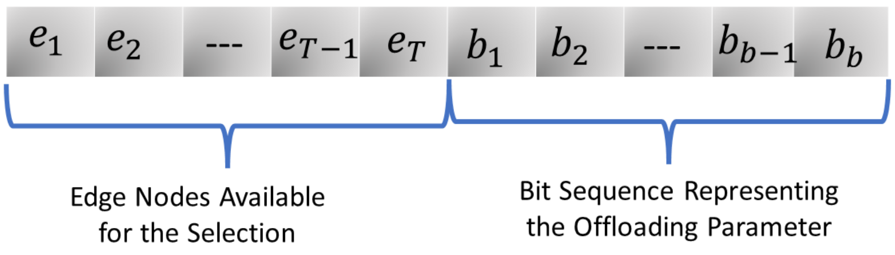

e_1
e_2
---
e_{T-1}
e_T
b_1
b_2
---
b_{b-1}
b_b
Edge Nodes Available for the Selection
Bit Sequence Representing the Offloading Parameter

FIGURE 5.3: Chromosome Example

$$+\Upsilon_1 \cdot \max\left(0, C1\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right)\right) + \Upsilon_2 \cdot \max\left(0, C2\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right)\right) + \Upsilon_3 \cdot \max\left(0, C3\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right)\right) \quad (5.19)$$

where Υ_1, Υ_2 and Υ_3 are the weighting coefficients for the penalty values, and:

$$C1\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right) = T_m^{x_m}\left(\alpha_{x_m}(\tau_i)\right) - T_{x_m}$$

$$C2\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right) = E_m^{x_m}(\alpha_{x_m}(\tau_i)) - E_{c,m}^{x_m}$$

$$C3\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right) = T_{m,e}^{off}(\alpha_{x_m}(\tau_i)) - T_{v_m,e}^{soj}(\tau_i)$$

$C1\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right)$ is the additional fitness penalty for VUs not performing task processing within the service latency requirement, $C2\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right)$ is the penalty for not respecting the energy constraint defined in (5.17), and $C3\left(\alpha_{x_m}(\tau_i), a_{(m,e)}(\tau_i)\right)$ is the supplementary penalty for VUs not performing the offloading process before moving out of T-NT coverage.

Selection The selection function S_f selects the parent solutions and is implemented through the roulette wheel selection technique with the selection probability depending upon the individual solutions fitness score. For a given minimization problem, the parent with the lowest fitness are selected at each round for the reproduction stage.

Crossover

In the crossover operator, new chromosomes $(\mathcal{C}_1^{\text{new}}, \mathcal{C}_2^{\text{new}})$ are generated by alternating genes of the parents $(\mathcal{C}_1^{\text{old}}, \mathcal{C}_2^{\text{old}})$ from a crossover point. Thus, child chromosomes can be written as

$$\mathcal{C}_1^{\text{new}} = \Phi\mathcal{C}_1^{\text{old}} + (1 - \Phi)\mathcal{C}_2^{\text{old}}$$
$$\mathcal{C}_2^{\text{new}} = \Phi\mathcal{C}_2^{\text{old}} + (1 - \Phi)\mathcal{C}_1^{\text{old}}$$

where Φ is the crossover point uniformly distributed in $[\Lambda, (1 + \Lambda)]$, i.e., $\Phi \sim U(-\Lambda, 1 + \Lambda)$

Mutation The mutation operator is based upon a Gaussian function where selected genes (β_m) from a child $\mathcal{C}$ can be altered by adding a random value from a Gaussian distribution, i.e., $\beta_m \longrightarrow \beta_m + \nu$, where, ν is a random variable with a Gaussian distribution, i.e., $\nu \sim \mathcal{N}(\bar{\mu}, \bar{\sigma}^2)$.

5.2.2.1 Adaptive GA Process (A-GA)

Given the dynamic nature of the VN, the number of resources available at each EN can impact the GA results. In the conventional GA process, each $v_m \in \mathcal{V}$ can perform the GA process till some stopping criteria (i.e., maximum GA iterations). With the involvement of many VUs with inherent dynamicity and randomness, finding the exact information about the available EN resources can be challenging. This can impact the GA results since both the EN selection and the offloading parameter values can depend upon the available EN resources. Such improper resource information can lead to the incorrect EN selection and the offloading process, which in result can degrade the performance. An adaptive GA process, lasting for several global iterations, is proposed. In particular, for every global iteration, all VUs perform the local GA operations, and the results is used to update the EN resource allocation. Once updated, the available resource information is then used in the next iteration for refining the GA performance.

Algorithm 6 shows the steps used during the implementation of GA for solving the joint network selection and computation offloading problem. The GA process begins with the random allocation of all VUs to the available ENs (Line 1). After that, an iterative GA process for I_{max} global iterations, is implemented wherein each iteration the VU assignment towards EN is updated and used for the next iteration. In particular, the updated VU-EN assignment information is used for allocating the EN resources (Line 3) by calling the resource allocation procedure detailed in Algorithm 7. After the resource allocation phase, every v_m performs the GA process. The primary GA steps contain the evaluation of $\mathcal{PS}$ (Lines 7-9), selection of better fit individuals as parent $\mathcal{C}$s (Lines 10-12), and generation of new possibly better fit $\mathcal{C}$s for the next generation (Lines 13-16). For each VU, the GA process terminates after a maximum number of local iterations $G_{\max}$ are reached.

Algorithm 7 describes the resource allocation procedure. For each VU, based upon the VU-EN assignment matrix $\mathcal{A}$, the number of EN available for the connection $\mathcal{E}_m(\tau_i)$ is determined (Line 3). Additionally, the number VUs associated with each EN $(U_e(\tau_i), \forall e \in \mathcal{E}_m(\tau_i))$ is also determined (Line 4). This information is then used to perform the uniform resource allocation for all the UVs associated with different ENs. In the case of the adaptive GA process, the resource allocation process is adapting the results generated by the GA in every global iteration. This allows more accurate allocation of scarce computation and communication resources of EN

Algorithm 6 The proposed Adaptive GA-based Approach

```
Input: $\mathcal{FF}, G_{\max}, \Phi, , \mathcal{S}, \mathcal{H}, \mathcal{L}, \mathcal{R}, \mathcal{V}, D_{m,e}, \Xi_e, c_e, f_e, B_e, \forall e$
Output: $\mathcal{A}, \mathbf{A}$

1: Assign all $v_m \in \mathcal{V}$ randomly to the available ENs and find $\mathcal{A}$
2: while $j \leq I_{\max}$ do
3:     $\{b_e^{x_m}(\tau_i)\}, \{c_e^{x_m}(\tau_i) \cdot f_e^{x_m}(\tau_i)\} =$
       Resource Allocation($\mathcal{V}, \mathcal{S}, \mathcal{H}, \mathcal{L}, \mathcal{R}, \mathcal{A}, D_{m,e}, \Xi_e, c_e, f_e, B_e, \forall e$)
4:     for $v_m \in \mathcal{V}$ do
5:         Generate the initial population space
6:         while $i \leq G_{\max}$ do
7:             function Evaluate($\mathcal{PS}$)
8:                 Find $\mathcal{FF}(C), \ \forall C \in \mathcal{PS}$.
9:             end function
10:            function Search($\mathcal{PS}$)
11:                Select better fit individuals using $S_f$
12:            end function
13:            function Create($\mathcal{PS}$)
14:                Generate new $C$s through Crossover and Mutation (using $\Phi, \nu$).
15:                Integrate $C$s with current $\mathcal{PS}$; sort them using fitness scores $\mathcal{FF}(C)$
16:            end function
17:            Replace current $\mathcal{PS}$ with new best set of $C$s.
18:            $i = i + 1$
19:        end while
20:        return $a_{m,e}, \alpha_{x_m}$
21:    end for
22:    Find the number of VUs per EN based upon GA results
23: end while
24: return $\mathcal{A}, \mathbf{A}$
```

Algorithm 7 Resource Allocation Function

```
Input: $\mathcal{V}, \mathcal{S}, \mathcal{H}, \mathcal{L}, \mathcal{R}, \mathcal{A}, D_{m,e}, \Xi_e, c_e, f_e, B_e, \forall e$
Output: $\{b_e^{x_m}(\tau_i)\}, \{c_e^{x_m}(\tau_i) \cdot f_e^{x_m}(\tau_i)\}$

1: function Resource Allocation($\mathcal{V}, \mathcal{S}, \mathcal{H}, \mathcal{L}, \mathcal{R}, \mathcal{A}, D_{m,e}, \Xi_e, c_e, f_e, B_e, \forall e$)
2:     for $v_m \in \mathcal{V}$ do
3:         $\mathcal{E}_m(\tau_i) = \{e | D_{m,e}(\tau_i) > 0, \bar{\Xi}_{x_m} \in \Xi_e, \quad \forall e\}$
4:         $U_e(\tau_i) = \sum_m \sum_e (a_(m,e)(\tau_i))$
5:         for $e \in \mathcal{E}_m(\tau_i)$ do
6:             $B_{v,m}^{\mathrm{e}}(\tau_i) = \frac{B_e}{U_e(\tau_i)}, \quad c_e^{x_m}(\tau_i) \cdot f_e^{x_m}(\tau_i) = \frac{c_e \cdot f_e}{U_e(\tau_i)}$
7:         end for
8:     end for
9: end function
10: return $\{b_e^{x_m}(\tau_i)\}, \{c_e^{x_m}(\tau_i) \cdot f_e^{x_m}(\tau_i)\}$
```

to VU and as result increases the performance (as shown later in the simulation results section)

5.2.2.2 Benchmark Solutions

Three different benchmark solutions are considered for comparison purposes.

Probabilistic VU-EN assignment In this approach $\forall v_m \in \mathcal{V}$ we select the node e randomly. The probability of v_m selecting EN e is given by:

$$Pr\{a_{m,e}(\tau_i) = 1\} = \frac{1}{\mathcal{E}_m(\tau_i)} \tag{5.20}$$

Position-based VU-EN Assignments In this case, each nearby competing VU is allocated to the ENs based on the available distance before it passes through the ENs coverage range and the distance between VU and EN. Thus, $\forall v_m \in \mathcal{V}$:

$$a_{m,e}(\tau_i) = 1 \Leftrightarrow \frac{D_{m,e}(\tau_i)}{d_{m,e}(\tau_i)} = \max_{e \in \mathcal{E}_m(\tau_i)} \left\{ \frac{D_{m,e}(\tau_i)}{d_{m,e}(\tau_i)} \right\} \tag{5.21}$$

Local Computation In this case, VUs are performing the task computation by themselves without employing any EN.

5.2.3 Numerical Results

The proposed adaptive GA approach and the other benchmark methods are simulated on a Matlab-based simulator for analyzing the performance. In Table 5.1, the main simulation parameters are provided for the network architecture used during the simulation. We have considered the variable number of VUs from 200 up to 2000 that are randomly distributed over the two-lane road network of length 10 Km. VUs are under the coverage area of a Satellite node, P = 10 HAPs, U = 40 LAPs, and N = 100 RSUs. Each VU is traveling with a variable speed based upon the truncated normal distribution with mean (μ=10 m/s) and standard deviation σ=1. The probability of VU being active (i.e., having a task to offload) is 0.1 at any time instant. The service set Ξ includes the six services with RSU, LAP, and HAP providing a maximum of 2, 2, and 4 services respectively. As stated before, a

TABLE 5.1: Simulation parameters

Coverage $(R_{r,n}, R_{l,u}, R_{h,p})$	(50, 200, 1000) m
VU Computation Cap. $(c_{v,m} \cdot f_{v,m})$	8 GFLOPS
RSU Computation Cap. $(c_{r,n} \cdot f_{r,n})$	30 GFLOPS
LAP Computation Cap. $(c_{l,u} \cdot f_{l,u})$	30 GFLOPS
HAP Computation Cap. $(c_{h,p} \cdot f_{h,p})$	50 GFLOPS
LEO Computation Cap. $(c_{s,1} \cdot f_{s,1})$	80 GFLOPS
Altitude (h_l, h_h, h_s)	(1, 100, 2000) Km
Bandwidth $(B_{r,n}, B_{l,u}, B_{h,p}, B_{s,1})$	(20, 20, 50, 100) MHz
VU Speed Range $(\vec{v}_{\min}, \vec{v}_{\max})$.	(8 m/s, 14 m/s)
VU Power $(P_{c,v_m}, Pt_{v_m}, Pr_{v_m})$	(1.1, 1.5, 1.3) W
Task Size (D_{x_m})	3 MB
Task Latency Req. $(\bar{T}_{x_m})$	2 Sec
Task Computation Req. (Ω_{x_m})	$(10^3 \cdot D_{x_m})$ Flops
Elevation angle (θ_m)	$\mathcal{U}(20^0, (\pi/2 + 20)^o)$
Weighting Coefficients (η_1, η_2)	(0.5, 0.5)

satellite node can provide a complete set of services. The GA simulation parameters includes the $\Upsilon_1, \Upsilon_3 = 10$, $\Upsilon_2 = 1$, $\Lambda = 0.1$, $\bar{\mu} = 0.02$, and $\bar{\sigma} = 0.1$. Additionally initial population is constituted by 50 chromosomes with $G_{\max}$=50 and I_{max}=10.

As shown in (5.12), the main objective of this work is to minimize the joint cost of latency and energy during the computation offloading operation over the multi-EC enabled VN. In particular, in Fig. 5.4, we present the average joint latency and energy cost requirements of the solution methods proposed in the previous section for different VU densities. With limited search flexibility and improper EN selections, the benchmark methods are unable to keep the latency and energy costs under control. It can be seen that the proposed A-GA method is able to reduce the cost significantly and outperform the other benchmark methods. By selecting the proper ENs and offloading the correct amount of data, A-GA can use the limited resources of different EC layers effectively. Given the high demand for the different vehicular services in resource-constrained vehicular environments, the proposed scheme can be effectively used to serve VUs with new services at affordable costs.

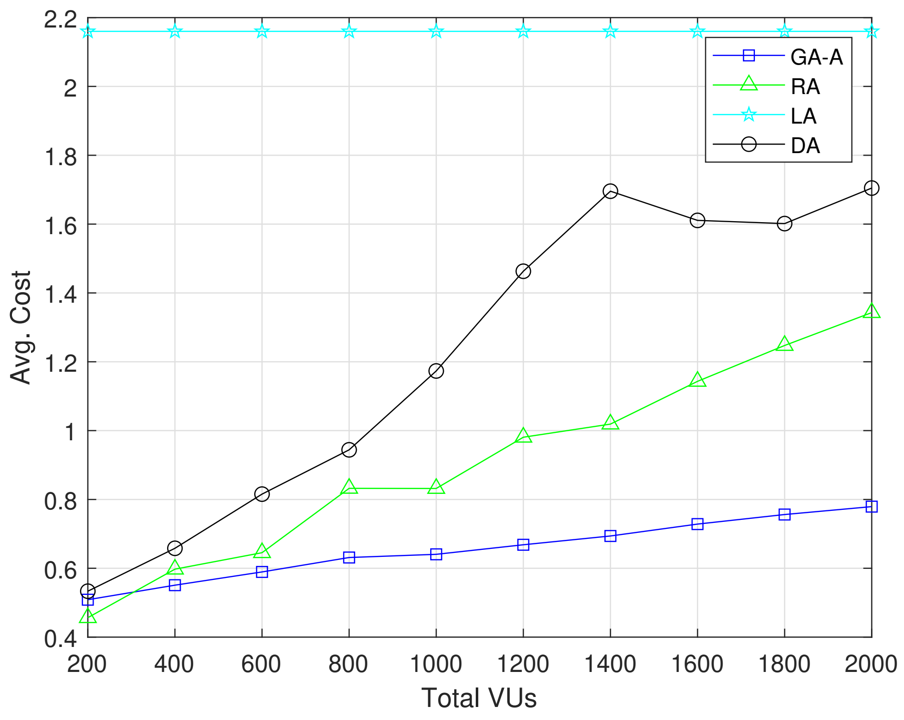


Figure 5.4: Joint Latency and Energy Cost.

Given the latency-critical natures of various vehicular services, it is important to show the latency performance of different methods. In Fig. 5.5, we present the latency requirements for various solution methods considered for solving the joint network selection and computation offloading problem over the multi-EC enabled VN. The A-GA scheme has a much better performance compared to the other methods through the proper offloading process. Given the presence of different EC layers located at varying distances, the A-GA method can select the proper EN resulting in reduced communication latency. It can also limit the computation cost by selecting the proper offloading parameters. Therefore A-GA method can be useful to serve the VUs with latency-critical service.

Day by day, VUs energy requirements are also becoming the critical bottleneck in VN, mainly due to a huge demand for high-quality vehicular services. Therefore it is important to analyze the energy performance during the computation offloading

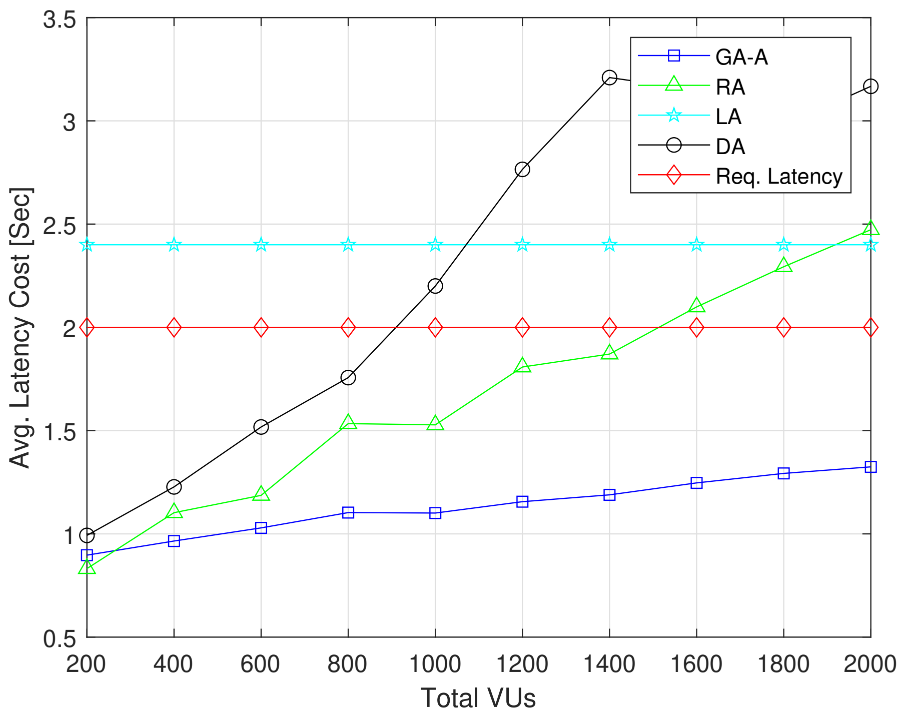


FIGURE 5.5: Latency Cost.

process. In Fig. 5.6, we show the energy requirements of different solution methods. It can be seen that the A-GA method is able to keep the energy cost under control with a proper offloading process and outperforms the LA methods. With the complete offloading process, even the other two benchmark methods, i.e., RA and DA, have better energy performance however, they suffer from huge latency cost as shown in Fig. 5.5.

To have a robust system, decreasing the latency and energy costs can not be sufficient, and is important to measure the number of times the system is not able to respect the different constraints. Here, in Fig. 5.7 we present the average number of VUs that are unable to follow the service latency requirements during the task processing operations with varying VUs. The benchmark methods with improper EN selections and the offloading process are having a large number of failures. On the other hand, the proposed A-GA method with adequate search flexibility offloads

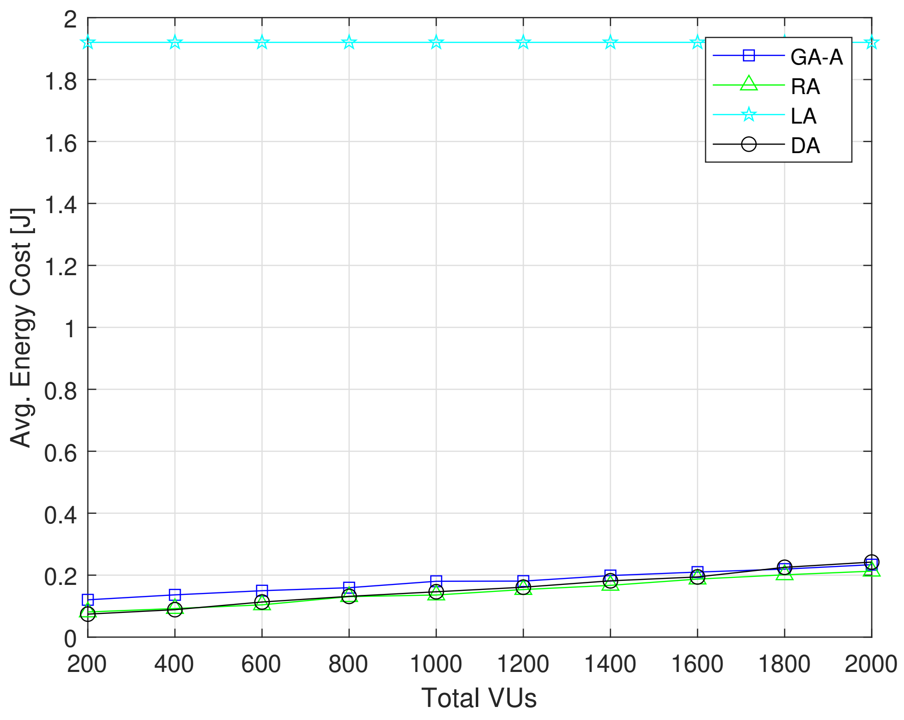


FIGURE 5.6: Energy Cost.

the proper amount of data to the selected EN and as result limits the number of failures. Thus, for critical systems such as VN, the introduced A-GA method can help to provide new services with higher reliability.

For dynamic VNs, the mobility constraint is important to follow to avoid the additional handover costs. Therefore each VU needs to complete the task offloading process that includes the task data transmission, EN-based computation, and receiving back the results in a limited sojourn time i.e., the time available before VU passes through the EN coverage. It can be seen that the proposed A-GA method has less number of failures mainly due to the proper offloading process that includes both EN selection and offloading. On the other hand, the benchmark methods are not able to adopt the proper offloading strategies and fail to perform the offloading process in limited sojourn times, resulting in higher failures. This shows that the proposed A-GA method can be useful over VN having high-speed VUs requesting

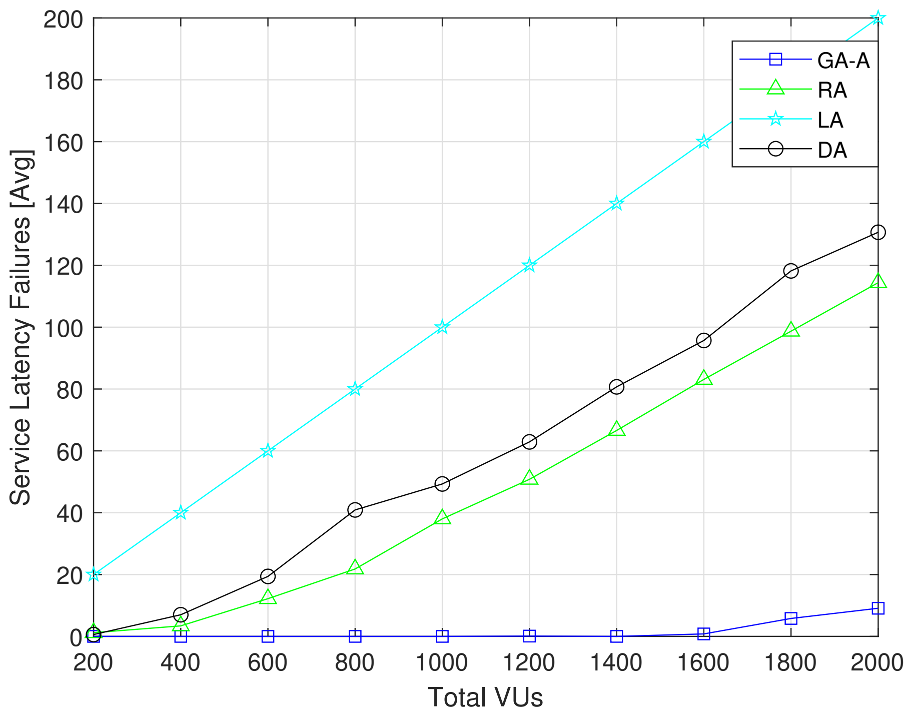


FIGURE 5.7: Service Latency Failures.

various services.

Finally, in Fig. 5.9 we show the impact in terms of node selection at each layer performed by the VUs. For this reason we focus on the proposed A-GA approach, by showing how the different VUs select nodes belonging to the different layers in percentage. It is possible to notice that when the number of VUs is reduced, VUs prefer to explore the EC resources from the upper layers, mainly due to coverage reasons. However, as the VU density increases, more VUs are selecting the nearby RSU nodes despite their limited communication distance. This shows that the proposed system considering multiple layers allows to exploit different layer characteristics.

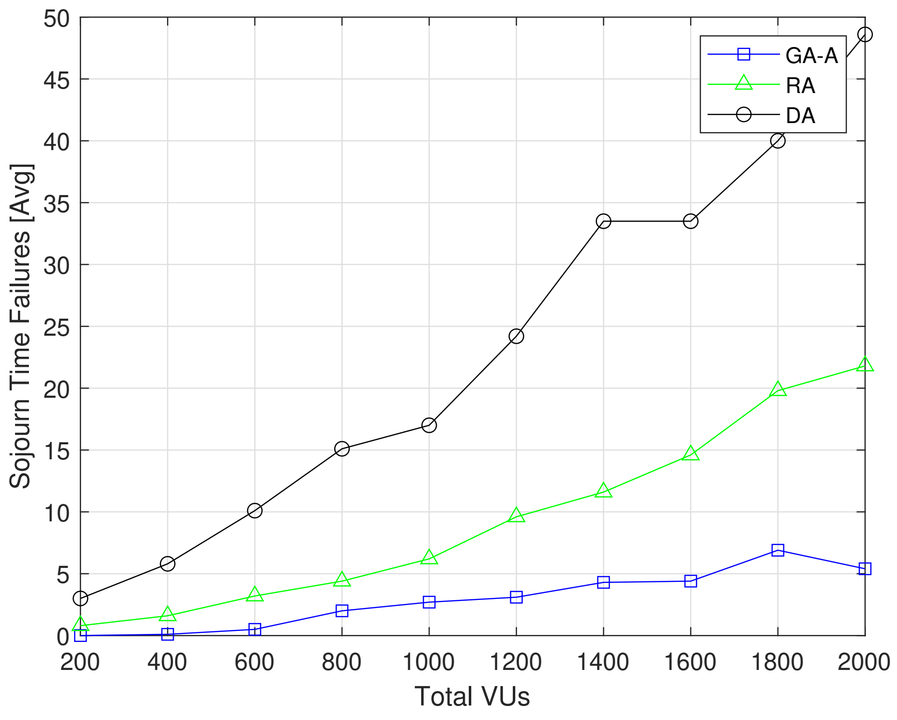


FIGURE 5.8: Sojourn Time Failures.

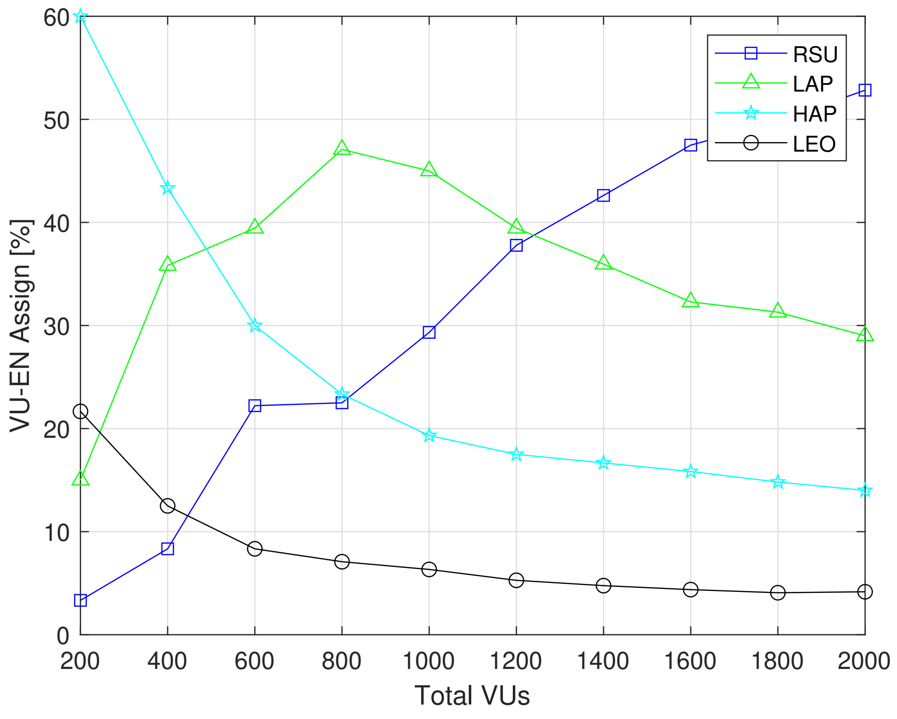


FIGURE 5.9: VU-EN Assignment for A-GA Method.

5.2.4 Conclusions

In this work, we have considered a joint latency and energy minimization for the Multi-EC enabled VN with T-NT layers of edge resources. A partial computation offloading process is considered, where VUs can request different services from edge-based servers with limited capabilities. Additionally, proper mobility models are considered to analyze the mobility of ground-based VUs and the satellite elements in the LEO. The problem is modeled as a constrained optimization problem, and a GA-based evolutionary search algorithm is proposed to solve it. The proposed adaptive GA technique uses multiple GA iterations for finding the proper solution by taking into account the limited resources. The simulation results with varying VUs density shows the effectiveness of the proposed method, over the benchmark approaches in terms of overall performance.

In the future, we plan to explore the impact when heterogeneous service requirements are considered as well as distributed machine learning approaches for solving more complex scenarios (e.g., multi-hop offloading)

5.3 Network Selection and Offloading with joint T-NTN Vehicular Scenario- HRL Solution

With the presence of heterogeneous edge computing facilities and multiple services, solving the network selection and offloading problem can be challenging. The heuristic and meta-heuristic approaches can be applied to some simplified scenarios. However, considering the complex nature of the problems and various decision variables, it is important to consider intelligent ML-based solution approaches. RL has shown great promise to solve such complex problems effectively in the case of terrestrial systems [69], and is a candidate to solve the considered problem. Recently, RL has been further specialized in different methods, e.g., Multi-agent RL [108], hierarchical RL (HRL) [109], and distributed RL [110]. As discussed before, the considered network selection and offloading problem can effectively be solved through a multi-level sequential decision-making process in terms of network layer selection, EC node selection, and offloading portion. Such multi-level hierarchical processes where a decision made at different levels can impact each other's performance can be solved through HRL methods.

The main contributions of this work are:

- **Multi-EC, Multi-service Joint T-NT Network:** We propose a multi-EC facility-enabled T-NT network for promoting VUs with various services. Ground-based VUs can request different services offered by the RSUs, LAPs, HAPs, and LEO satellites through the EC facilities onboard (Section 5.3.1).
- **Joint Latency and Energy Minimization Problem:** A proper mathematical model is proposed by including latency and energy requirements of the different steps involved during the VUs task processing operation. Both VU and EN side latency/energy cost is considered while modeling the offloading process. In the end, an optimization problem is developed for minimizing the overall latency and energy consumption by selecting a proper EC facility and the amount to be offloaded (Section 5.3.1).
- **Multi-level Sequential Decision-Making Process:** The joint network selection and computation offloading problem is modeled as a multi-level sequential decision-making process through MDPs and an advanced DQN approach is considered for finding optimal policies (Section 5.3.2).

- **Performance Analysis:** In addition, a set of benchmark methods are used to analyze the results, showing the improved latency and energy performance of a proposed scheme (Section 5.3.3).

5.3.1 System Model and problem Formulation

The system model includes a multi-tiered EC facility for serving VUs with a set of services. We consider an integrated T-NT network composed of a LEO satellite constellation $\mathcal{S} = \{s_1, \ldots, s_q, \ldots, s_Q\}$ with Q satellites, a set $\mathcal{H} = \{h_1, \ldots, h_p, \ldots, h_P\}$ of P HAPs, a set $\mathcal{L} = \{l_1, \ldots, l_u, \ldots, l_U\}$ of U LAPs, a set $\mathcal{R} = \{r_1, \ldots, r_n, \ldots, r_N\}$ of N RSUs, and a set of $\mathcal{V} = \{v_1, \ldots, v_k, \ldots, v_K\}$ of K VUs, located randomly in the area, which is supposed to be modeled as a multilayered EC-enabled vehicular scenario. Additionally, one cloud facility C is considered. VUs can explore the EC facilities provided by different layers for enabling various applications and services. The VN is modeled as a time-discrete system where the network parameters are supposed to be constant in each time interval τ, where τ_i identifies the ith time interval, i.e., $\tau_i = \{\forall t | t \in [i\tau, (i+1)\tau]\}$. For avoiding the additional complexity we have considered that in each time instance, VUs can access to services on the LEO satellite under visibility, while the whole constellation can be reached through proper inter-satellite links. Also, each EC layer can have access to cloud facility through backhaul links. Fig. 5.10 shows the various elements of the considered VN scenario composed by one reference LEO satellite, HAPs, LAPs, RSUs, cloud computing facility, and VUs.

The generic kth VU is characterized by a processing capability equal to $c_{v,k}$ Floating Point Operations per Second (FLOPS) per CPU cycle, while its CPU frequency is $f_{v,k}$. Each VU is supposed to be able to communicate using a bandwidth $B^{\mathrm{e}}_{v,k}$ with

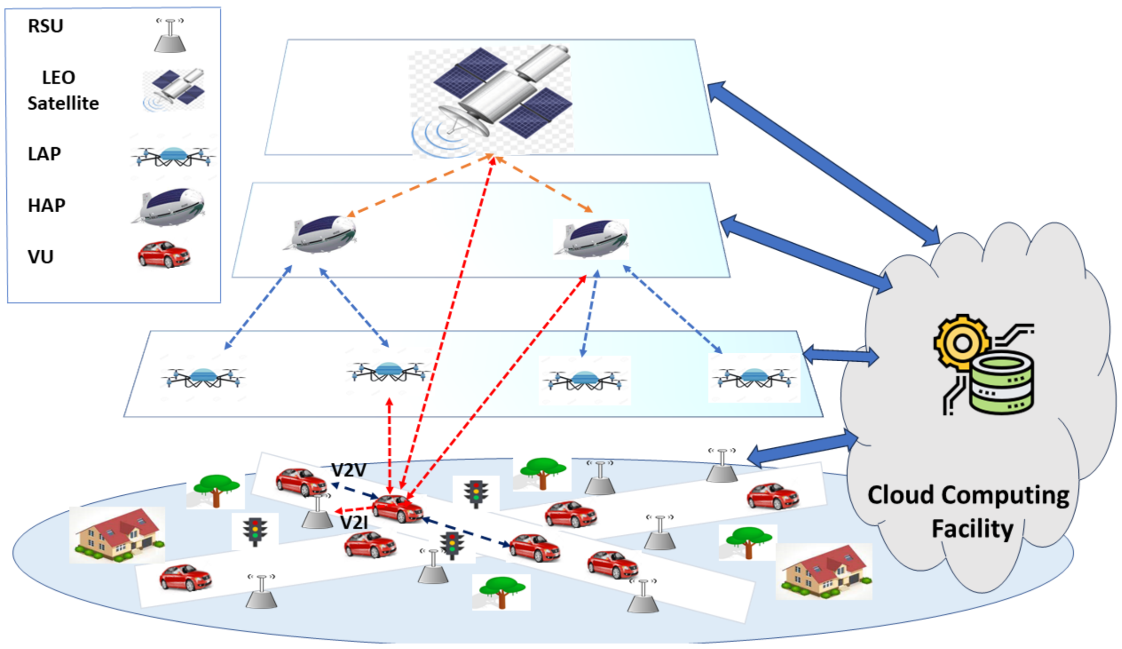


FIGURE 5.10: NTN Edge Computing System Model

a reference EN $e \in \{\mathcal{S} \cup \mathcal{H} \cup \mathcal{L} \cup \mathcal{R} \cup\}$. At each interval, the kth VU is supposed to generate a task request x_k to be processed, where task x_k is identified through the tuple $\langle D_{x_k}, \Omega_{x_k}, \bar{T}_{x_k}, \bar{\Xi}_{x_k} \rangle$ where D_{x_k} is the task size in Byte, Ω_{x_k} are the CPU execution cycles requested, $\bar{T}_{x_k}$ is the maximum latency of the requested service, and $\bar{\Xi}_{x_k}$ is the type of service requested by the VU.

The eth EN (i.e., one node among any RSU, LAP, HAP, LEO or Cloud)[4] is characterized by a processing capability equal to c_e FLOPS per CPU cycle, with CPU frequency f_e, and communication capabilities, supposed to be identified through a communication technology able to work on a bandwidth B_e and covering an area with radius R_e. The LAP, HAP and satellite nodes are located at height h_l, h_h and h_s from the ground level, respectively. Each EN provides computation offloading services to the VUs within its coverage area. VUs tasks are characterized by the service type, supposing the system able to provide different services, where $\Xi = \{\xi_1, \ldots, \xi_z, \ldots, \xi_Z\}$ is the set of possible services and Z the maximum number

[4]In the following, $e = \{r_n\}$, $e = \{l_u\}$, $e = \{h_p\}$, $e = \{s_q\}$ and $e = C$ stands for the rth RSU, uth LAP, pth HAP, qth LEO satellite and a cloud facility.

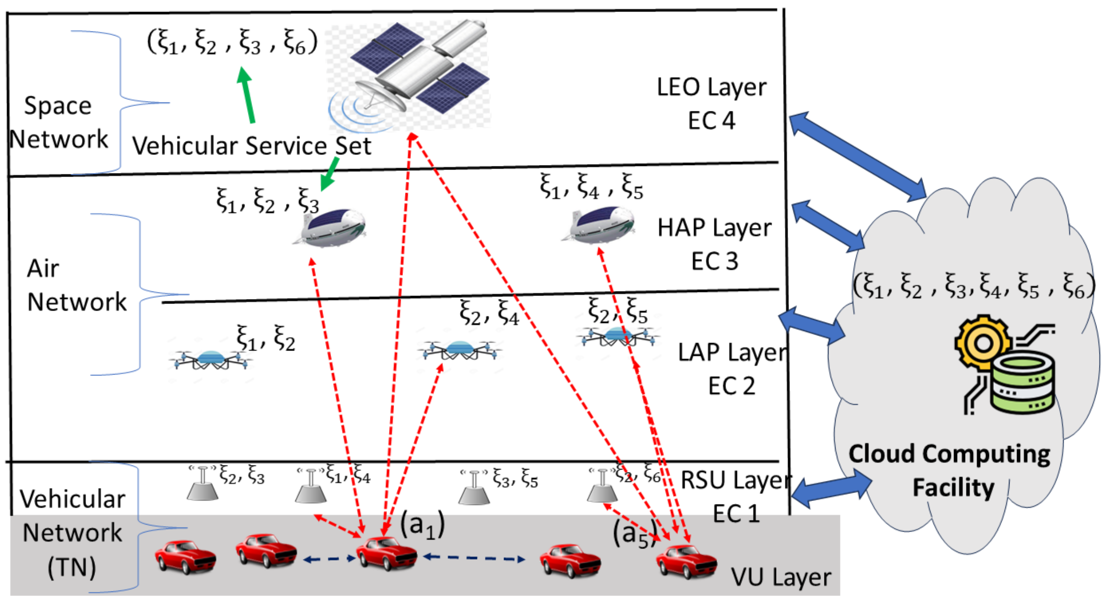


FIGURE 5.11: Multi Service VN with Non-terrestrial EC Layers

of services. With limited storage capabilities, RSUs, LAPs, HAPs and LEO satellites can provide a limited number of services. We consider that the eth EN can provide the service set $\Xi^e = \{\xi_1, \cdots \xi_z, \cdots, \xi_e^{max}\}$. Additionally, the cloud facility C is able to provide all the possible services requested by VUs. Fig. 5.11 shows the multi-service vehicular scenario where RSUs, LAPs HAPs, and LEO satellites are providing a subset of services (i.e., 2, 2, 3, 4, respectively) while the cloud facility is able to provide all possible services (i.e., 6) requested by the VUs.

5.3.1.1 VUs Mobility and Distance Measures

In this work, we consider that VUs are moving with a variable speed $\vec{v}_{v,k}(\tau_i)$, whose value is bounded within $\vec{v}_{\text{min}}$ and $\vec{v}_{\text{max}}$ [105], while the kth VU instantaneous speed is modeled through a truncated normal distribution density function:

$$f\left(\vec{v}_{v,k}(\tau_i)\right) = \begin{cases} \dfrac{2e^{\frac{-\left(\vec{v}_{v,k}(\tau_i)-\mu\right)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}\left(\operatorname{erf}\left(\frac{\vec{v}_{\max}-\mu}{\sigma\sqrt{2}}\right)-\operatorname{erf}\left(\frac{\vec{v}_{\min}-\mu}{\sigma\sqrt{2}}\right)\right)}, \\ \qquad\qquad \vec{v}_{\min} \leq \vec{v}_{v,k}(\tau_i) \leq \vec{v}_{\max} \\ 0, \quad \text{else} \end{cases} \tag{5.22}$$

where μ and σ are the mean and standard deviation of the vehicles speed, and $\operatorname{erf}(x)$ is the Gauss error function over x. The path length within which the kth VU remains under the coverage of any terrestrial and aerial eth node (i.e., RSU, UAV, HAP) is $D_{v_k,e}(\tau_i)$ and can be given by:

$$D_{v_k,e}(\tau_i) = \sqrt{d_e^2 - \left(y_e - y_{v_k}(\tau_i)\right)^2} \pm \left(x_e - x_{v_k}(\tau_i)\right) \tag{5.23}$$

where, $\left(x_{v_k}(\tau_i), y_{v_k}(\tau_i)\right)$ is the location of the kth VU at τ_i and (x_e, y_e) is the projection over the ground of a generic eth EC node, which can be RSU, UAV, HAP. The available sojourn time for the kth VU with respect to a generic eth node (RSU, LAP or HAP) can be written as:

$$T^{\text{soj}}_{v_k,e}(\tau_i) = \frac{D_{v_k,e}(\tau_i)}{|\vec{v}_{v,k}(\tau_i)|} \qquad \forall i, v_k \tag{5.24}$$

5.3.1.2 LEO Satellite Mobility and Distance Measures

In general, VUs can communicate with satellite nodes within a specific time interval, which depends on the locations and the mobility patterns of LEO satellites. LEO satellites can move at a very high speed (i.e., a few km per second) compared with the VUs, which move at a few meters per second. Here we consider a well-known coverage model for finding the arc length over which ground-based VUs can communicate [106].

Let us consider a LEO satellite s_q located at height h_q with respect to the ground, moving with a constant speed $\vec{v}_{s,q}$ km/s. Let us consider that at a certain instant τ_i its elevation angle with respect to the kth VU is $\theta_k(\tau_i)$, corresponding to a geocentric angle for the kth VU as,

$$\delta_{q,k}(\tau_i) = \arccos\left(\frac{R_e}{R_e + h_q} \cdot \cos\left(\theta_k(\tau_i)\right)\right) - \theta_k(\tau_i)$$

where R_e is the Earth radius. The total arc length over which VU can communicate with the considered LEO satellite is defined as,

$$L_{q,k}(\tau_i) = 2(R_e + h_q) \cdot \delta_{q,k}(\tau_i)$$

With this, the total time within which the kth VU can be in the coverage range of the LEO is given as,

$$T^{\text{soj}}_{v_k,q}(\tau_i) = \frac{L_{q,k}(\tau_i)}{\vec{v}_{s,q}} \tag{5.25}$$

The kth VU should complete the offloading process towards the LEO satellite during the limited coverage time, for avoiding additional costs in terms of LEO satellite handovers. Since the satellite nodes have much higher speed than the VUs, the VUs mobility is neglected while defining the satellite coverage space.

Eqs. (5.24) and (5.25) defines the mobility constraint for the considered multi-EC vehicular networking system.

5.3.1.3 Network Selection and Task Offloading Process

For each ith interval, the kth VU task is supposed to be managed through a partial computation offloading process allowing a portion of the task to be offloaded towards the selected EN while the rest can be locally processed, hence, allowing to reduce the overall processing time and energy cost [76]. Here, we consider $0 \leq \alpha_{x_k}(\tau_i) \leq 1$ as an offloading index associated with the kth VU representing the portion of the task offloaded towards the selected EN, where $\alpha_{x_k}(\tau_i) = 1$ corresponds to a complete offloading, while $\alpha_{x_k}(\tau_i) = 0$ corresponds to perform only local processing. To avoid additional complexity, we assume that each VU can offload only towards one EN. We define a network selection variable, $a_{(k,e)}(\tau_i)$, indicating that the kth VU has selected the eth EN at τ_i for the task offloading, where

$$\sum\nolimits_e a_{(k,e)}(\tau_i) \leq 1, \quad \forall i \tag{5.26}$$

corresponding to say that the kth VU can select at most one of the possible EN under visibility. The vehicular task processing operation with partial computation offloading involves several steps such as the transmission of a selected task portion towards a selected EN, task computation operation at EN, reception of task back at VU, local computation of remaining task, etc. Here we present the generic task computation and communication models in the vehicular scenario, which are later used to model the overall task computation latency and energy.

Task Computation Model

The time and energy required for task computation on any $\hat{l}$th device for a generic task $x_{\hat{k}}$ can be written as:

$$T^{x_{\hat{k}}}_{c,\hat{l}} = \frac{\Omega_{x_{\hat{k}}}}{c^{x_{\hat{k}}}_{\hat{l}} f^{x_{\hat{k}}}_{\hat{l}}}, \quad E^{x_{\hat{k}}}_{c,\hat{l}} = T^{x_{\hat{k}}}_{c,\hat{l}} P_{c,\hat{l}} \tag{5.27}$$

where $c^{x_{\hat{k}}}_{\hat{l}}$ and $f^{x_{\hat{k}}}_{\hat{l}}$ are the number of FLOPS per CPU-cycle and CPU-frequency allocated to the task $x_{\hat{k}}$, respectively, whether $\hat{l}$ identifies a VU (v_k), a RSU (r_n), a

LAP (l_u), a HAP (h_p) or a LEO satellite (s_q). We have considered that the nodes capacity will be equally shared by all the tasks allocated to it. Also, in (5.27), $P_{c,\hat{l}}$ is the power used by the generic $\hat{l}$th device for the task computation phase.

Task Communication Model

The partial computation offloading process considers the possibility to split the tasks into sub-portions and offload only part of the task, while the rest is locally processed at the originating device; this means that both transmission of the VUs data portion towards the selected EN and the reception back of the results from EN should be considered. The total time and energy consumed during the transmission of data from a generic node $\hat{k}$ to another node $\hat{l}$ for task $x_{\hat{k}}$ are given by[5]:

$$T^{x_{\hat{k}}}_{tx,\hat{k}\hat{l}}(\tau_i) = \frac{D_{x_{\hat{k}}}}{r_{\hat{k}\hat{l}}(\tau_i)}, \quad E^{x_{\hat{k}}}_{tx,\hat{k}\hat{l}}(\tau_i) = T^{x_{\hat{k}}}_{tx,\hat{k}\hat{l}}(\tau_i) Pt_{\hat{k}} \tag{5.28}$$

where $r_{\hat{k}\hat{l}}(\tau_i)$ is data-rate of the link between the two nodes Similarly, the time and energy required to receive the task of size $D^r_{x_{\hat{k}}}$ from $\hat{l}$th EN to $\hat{k}$ are:

$$T^{x_{\hat{k}}}_{rx,\hat{l}\hat{k}}(\tau_i) = \frac{D^r_{x_{\hat{k}}}}{r_{\hat{k}\hat{l}}(\tau_i)}, \quad E^{x_{\hat{k}}}_{rx,\hat{l}\hat{k}}(\tau_i) = T^{x_{\hat{k}}}_{rx,\hat{l}\hat{k}}(\tau_i) Pr_{\hat{k}} \tag{5.29}$$

where $Pr_{\hat{k}}$ is the power consumed for receiving data. Additionally, a symmetric channel model is assumed between $\hat{k}$ and $\hat{l}$. The channel between $\hat{k}$ and $\hat{l}$ at ith interval is characterized by the link gain, modeled as [111]:

$$h_{\hat{k},\hat{l}}(\tau_i) = \beta_0 \cdot d^{\theta}_{\hat{k},\hat{l}}(\tau_i)$$

[5]Here $\hat{l}$ and $\hat{k}$ are the indexes of any generic node among v_k, r_n, l_u, h_p, and s_q.

where, $d_{\hat{k},\hat{l}}(\tau_i)$ is the distance between node $\hat{k}$ and $\hat{l}$ at ith interval, β_0 is the channel power gain at 1 m reference distance, while θ is the path loss coefficient over different communication links. The expression for the channel transmission rate is based on the Shannon capacity formula and can be written as:

$$r_{\hat{k}\hat{l}}(\tau_i) = b_{\hat{l}}^{x_{\hat{k}}}(\tau_i)\log_2\left(1 + \frac{Pt_{\hat{k}} \cdot h_{\hat{k},\hat{l}}(\tau_i)}{N_0}\right) \quad \forall \hat{k}, \hat{l}$$

where $Pt_{\hat{k}}$ is the transmission power of a node $\hat{k}$, $b_{\hat{l}}^{x_{\hat{k}}}(\tau_i)$ is the communication bandwidth, and $N_0 = N_T b_{\hat{l}}^{x_{\hat{k}}}(\tau_i)$ is the thermal noise power with noise power spectral density N_T.

Next, we model the total time and energy required for the task offloading and the local computation process.

5.3.1.4 Task Offloading Process

Offloading Process During the partial computation offloading process, if the kth VU selects the eth EN, the time and energy required to offload the task to e and to get back the result at v_k in the ith interval are:

$$T_{k,e}^{\text{off}}(\alpha_{x_k}(\tau_i)) = \alpha_{x_k}(\tau_i)\left(T_{tx,ke}^{x_k} + T_{w,e}^{x_k} + (1 - b_{\Xi_{x_k}}^{e})T_{c,e}^{x_k} + T_{rx,ek}^{x_k} + b_{\Xi_{x_k}}^{e}\left(T_{tx,eC}^{x_k} + T_{w,C}^{x_k} + T_{c,C}^{x_k} + T_{rx,Ck}^{x_k}\right)\right) \tag{5.30a}$$

$$E_{k}^{\text{off}}(\alpha_{x_k}(\tau_i)) = \alpha_{x_k}(\tau_i)\left(E_{tx,ke}^{x_k} + E_{rx,ek}^{x_k}\right) \tag{5.30b}$$

$$E_{e}^{\text{off}}(\alpha_{x_k}(\tau_i)) = \alpha_{x_k}(\tau_i)\left(E_{w,e}^{x_k} + (1 - b_{\Xi_{x_k}}^{e})E_{c,e}^{x_k} + b_{\Xi_{x_k}}^{e}\left(E_{tx,eC}^{x_k} + E_{rx,Ce}^{x_k}\right)\right) \tag{5.30c}$$

where $T_{tx,ke}^{x_k}$, $T_{w,e}^{x_k}$ $T_{c,e}^{x_k}$, and $T_{rx,ek}^{x_k}$ are the transmission time, waiting time at e for receiving the task data, computation time at e, and the receiving time for the task

x_k generated by v_k during the offloading phase, and $E^{x_k}_{tx,ke}$, and $E^{x_k}_{rx,ek}$ are the energy consumed by VU during the task transmission, and result collection phases on the device, while $E^{x_k}_{c,e}$ and $E^{x_k}_{w,e}$ are the energy consumed by the EN e during task computation and waiting phases, respectively. A binary variable $b^e_{\bar{\Xi}_{x_k}}$ is also considered, having value 1 if EN e is unable to provide a requested service $\bar{\Xi}_{x_k}$, else 0. In such case, EN e relays the user data towards the cloud facility C for further computation. Therefore, additional latency and energy components are considered accordingly. From a latency perspective, the time required to transmit the data to cloud facilities, waiting time at C, computation time, and the additional time required to receive back the computation results at e are considered. While energy cost is limited to the EN side costs only assuming that the cloud facilities have a stable energy supply from the electric grid. The energy required to transmit and receive back the data from the cloud facility is considered for cloud-based computations.

Local Computation The amount of time and energy required for computing the task locally in the ith interval are:

$$T^{\mathrm{loc}}_k(\alpha_{x_k}(\tau_i)) = \left(1 - \alpha_{x_k}(\tau_i)\right) T^{x_k}_{c,k} \tag{5.31a}$$

$$E^{\mathrm{loc}}_k(\alpha_{x_k}(\tau_i)) = \left(1 - \alpha_{x_k}(\tau_i)\right) E^{x_k}_{c,k} \tag{5.31b}$$

where $T^{x_k}_{c,k}$ and $E^{x_k}_{c,k}$ are the time and energy spent for the whole task x_k local processing, while $\alpha_{x_k}(\tau_i)$ is the portion of the task locally processed at the time interval τ_i.

Partial offloading Computation The delay and the energy consumed during the task processing phases, when partial offloading is performed in the ith interval, can be written as:

$$T^{x_k}(\alpha_{x_k}(\tau_i), a_{(k,e)}(\tau_i)) = \max\left\{T^{\text{off}}_{k,e}(\alpha_{x_k}(\tau_i)), T^{\text{loc}}_{k}(\alpha_{x_k}(\tau_i))\right\}$$
$$E^{x_k}(\alpha_{x_k}(\tau_i), a_{(k,e)}(\tau_i)) = w_k\left(E^{\text{off}}_{k}(\alpha_{x_k}(\tau_i)) + E^{\text{loc}}_{k}(\alpha_{x_k}(\tau_i))\right) + w_e E^{\text{off}}_{e}(\alpha_{x_k}(\tau_i))$$

where w_k and w_e are the weighting coefficients associated with the energy costs of VUs and ENs, respectively. In addition, each vehicle should finish the offloading process and receive the results back within the sojourn time, hence:

$$T^{\text{off}}_{k,e}(\alpha_{x_k}(\tau_i)) \leq T^{\text{soj}}_{v_k,e}(\tau_i) \quad \forall i \tag{5.32}$$

5.3.1.5 Problem Formulation

Main aim of this work is to optimize the network-wide performance of the multi-EC enabled VN. We aim to optimize the performance in terms of overall latency and energy consumed during the offloading process towards edge servers by selecting proper EN and offloading amounts. For this, we formulate the joint latency and energy minimization problem as:

$$\mathbf{P1}: \min_{\mathcal{A},\mathbf{A}}\left\{\frac{1}{K}\sum_{k=1}^{K}\left[\gamma_1 T^{x_k}\left(\alpha_{x_k}(\tau_i), a_{(k,e)}(\tau_i)\right) + \gamma_2 E^{x_k}\left(\alpha_{x_k}(\tau_i), a_{(k,e)}(\tau_i)\right)\right]\right\} \forall i \tag{5.33}$$

s.t.

$$\mathbf{C1}: \text{Eq. } (5.26) \tag{5.34a}$$
$$\mathbf{C2}: T^{x_k}\left(\alpha_{x_k}(\tau_i)\right) \leq T_{x_k} \quad \forall v_k \in \mathcal{V},\ \forall i \tag{5.34b}$$
$$\mathbf{C3}: \text{Eq. } (5.32) \tag{5.34c}$$
$$\mathbf{C4}: E^{x_k}(\alpha_{x_k}(\tau_i), a_{(k,e)}(\tau_i)) \leq w_k E^{x_k}_{c,k} \tag{5.34d}$$

$$\mathbf{C5}: 0 \leq \gamma_1, \gamma_2 \leq 1;\ \gamma_1 + \gamma_2 = 1 \tag{5.34e}$$

where $\mathbf{A} = \{\alpha_{x_k}\}^M$ is the computation offloading matrix, and γ_1, γ_2 are two weighting coefficients for balancing latency and energy consumption. **C1** stands that each VU can select at most one EN for the computation offloading. **C2** puts a limit on the maximum processing time as one of the task requirements. According to **C3**, for avoiding handover phenomena and related latency, each VN should complete the offloading process before it passes through the selected EN coverage. According to **C4**, the total energy cost for the task processing during the partial computation offloading process should be bounded by the cost of the energy needed to process the complete task locally. **C5** stands that the two weighting coefficients (γ_1, γ_2) should be between 0 and 1 with their sum equal to 1.

5.3.2 Hierarchical Reinforcement Learning Solution

In this work, by finding the proper ENs and offloading portions we aim to minimize the overall latency and energy cost during the vehicular task processing operation over the multi-service multi-EC enabled VN. Given the complex nature of the considered problem, traditional heuristic and meta-heuristic approaches can have limited impacts, and more advanced solutions are required. The formulated problem can be modeled as a sequential decision-making process through a proper MDP model, and RL-based solution methods can be adapted to solve it. With the presence of multiple heterogeneous EC platforms with different properties, i.e., speed, location, services, resources, a traditional single agent-based RL solution can be computationally expensive and might not even be feasible. In recent times, several new advanced RL methods have been introduced, especially for solving challenging problems over dynamic scenarios.

The considered problem can effectively be decomposed into multiple sub-problems impacting each other's performance and enabling a multi-level decision-making process with reduced complexity. The HRL method can be adapted for such multi-level decision-making processes. Therefore, we have considered an HRL-based solution method for solving the network selection and offloading problem over multi-service, multi-layer EC facilities.

P1 can be decomposed into three hierarchical learning processes, i.e., $\{\mathcal{P}^1, \mathcal{P}^2, \mathcal{P}^3\}$. In the first process, $\mathcal{P}^1$, RL agent aims to select a proper EC layer j for performing the task computation. Based upon the selected layer $j \in J$, with J being a maximum number of edge layers available for offloading, $\mathcal{P}^2$ aims to select a proper EC node e belonging to the jth layer for processing the kth VUs data. Next, through $\mathcal{P}^3$, VU aims to select the proper amount of data α_{x_k} to be offloaded towards e for minimizing the overall cost.

5.3.2.1 MDP Models

In general, the MDP for any problem p can be defined as a tuple $\langle \mathcal{S}^p, \mathcal{A}^p, \mathcal{R}^p, \hat{\mathcal{P}}^p, \gamma^p \rangle$, with state space $\mathcal{S}^p$, action space $\mathcal{A}^p$, reward $\mathcal{R}^p$, state-transition probabilities $\hat{\mathcal{P}}^p$ and discount factor γ^p. By solving $\mathcal{P}^1$, we aim to find a proper EC layer for computing the users' task. We define $\mathcal{S}^1 = \{s_1^1, \cdots, s_1^h, \cdots, s_1^H\}$ as a discrete state space associated with $\mathcal{P}^1$ with H states. Here, s_1^h is the hth state function of the kth VU and jth EC layer properties. The hth state of $\mathcal{P}^1$, based on VU k and layer $j \in J$ at time instance i is defined as,

$$s_1^h(\tau_i) = \{\bar{h}_{kj}(\tau_i), \bar{N}_j(\tau_i), \bar{s}_j(\tau_i), Pr_j^{\bar{\Xi}_{x_k}}(\tau_i)\}$$

where $\bar{h}_{kj}(\tau_i)$ is the distance measure modeling the relative distance between layer j and VU k, $\bar{N}_j(\tau_i)$ is the average number of VUs selecting the layer j for processing their data, $\bar{s}_j(\tau_i)$ is the mobility state of ENs defined as the average speed of all ENs belonging to particular j. Also, $Pr_j^{\bar{\Xi}_{x_k}}(\tau_i)$ measures the probability a service $\bar{\Xi}_{x_k}$ is present on the selected jth layer. Then, $\mathcal{A}^1 = \{a_1^1, \cdots, a_1^{\hat{h}}, \cdots, a_1^{\hat{H}}\}$ is the discrete action space for $\mathcal{P}^1$ with $\hat{H}$ actions, where the $\hat{h}$th action is defined as,

$$a_1^{\hat{h}} = \{0, 1\}_{1\times J} \quad \text{with,} \quad \sum a_1^{\hat{h}} \leq 1$$

where $a_1^{\hat{h}}$ is a binary vector modeling the VUs layer selection decision. The performance of $\mathcal{P}^1$ can be impacted by the node selection and offloading policies of $\mathcal{P}^2$ and $\mathcal{P}^3$. Therefore the reward received will be based on the policies adopted by these MDPs.

Sub-problem $\mathcal{P}^2$ receives the information from the higher level $\mathcal{P}^1$ in terms of selected layer j and other state parameters. It aims to select a proper EN from layer j for serving the VU. The node selection operation can be based on several parameters such as VUs location, service demand, speed, layer properties, etc. Here we introduce $\mathcal{S}^2 = \{s_2^1, \cdots, s_2^m, \cdots, s_2^M\}$ as a discrete state space associated with $\mathcal{P}^2$ with M states, where s_2^m is the mth state based on the kth VU data and the properties of jth EC layer selected in $\mathcal{P}^1$. The mth state associated with VU k and jth layer node e at τ_i is defined as,

$$s_2^m(\tau_i) = \{\hat{N}_e, \hat{s}_{kj_e}, \hat{T}_{kj_e}^{soj}\}$$

where $\hat{N}_e$ is the resource state of eth node modeled as an average number of VUs requesting services from e, $\hat{s}_{kj_e}$ is a binary service state that takes the value 1 if the service requested by kth VU is available at e, else 0. Then, $\hat{T}_{kj_e}^{soj}$ is the sojourn time state, modeled as a binary variable that becomes 1 if the eth EN is able to cover the VU k with more than ζ of its coverage space, else 0. In addition, $\mathcal{A}^2 = \{a_2^1, \cdots, a_2^{\hat{m}}, \cdots, a_2^{\hat{M}}\}$ is the discrete action space for $\mathcal{P}^2$ with $\hat{M}$ actions, where the $\hat{m}$th action is defined as,

$$a_2^{\hat{m}} = \{0,1\}_{1\times\bar{E}_j} \quad \text{with} \quad \sum a_2^{\hat{m}} \leq 1,$$

where $\bar{E}_j$ is the maximum number of egde nodes from layer j that can cover any VU. The performance of $\mathcal{P}^2$ can be impacted by the offloading policies of $\mathcal{P}^3$.

Sub-problem $\mathcal{P}^3$ receives information from higher levels about selected layers j, node e, and other state parameters. It aims to select a proper offloading parameter α_{x_k} for the offloading kth VUs data. Here we introduce $\mathcal{S}^3 = \{s_3^1, \cdots, s_3^l, \cdots, s_3^L\}$ as a discrete state space associated with $\mathcal{P}^3$ with L states, where s_3^l is the lth state based upon the VUs data and the properties of selected node e from jth EC layer. The lth state at time instance τ_i is defined as

$$s_3^l = \left\{F_{k,e}^1(\tau_i), F_{k,e}^2(\tau_i), F_{k,e}^3(\tau_i)\right\}$$

where, $F_{k,e}^1(\tau_i)$,$F_{k,e}^2(\tau_i)$, and $F_{k,e}^3(\tau_i)$ are three binary functions, defined as:

$$F_{k,e}^1(\tau_i) = \begin{cases} 0 & T_{k,e}^{off}(\alpha_{x_k}(\tau_i)) \leq T_{v_k,e}^{soj}(\tau_i) \\ 1 & \text{else} \end{cases} \tag{5.35}$$

$$F_{k,e}^2(\tau_i) = \begin{cases} 0 & T^{x_k}\left(\alpha_{x_k}(\tau_i)\right) \leq T_{x_k} \\ 1 & \text{else} \end{cases} \tag{5.36}$$

$$F_{k,e}^3(\tau_i) = \begin{cases} 0 & E^{x_k}(\alpha_{x_k}(\tau_i), a_{(k,e)}(\tau_i)) \leq w_k E_{c,k}^{x_k} \\ 1 & \text{else} \end{cases} \tag{5.37}$$

In addition, $\mathcal{A}^3 = \{a_3^1, \cdots, a_3^{\hat{l}}, \cdots, a_3^{\hat{L}}\}$ is the discrete action space for $\mathcal{P}^3$ with $\hat{L}$ actions, where the $\hat{l}$th action, $a_2^{\hat{l}} \in \{\alpha_{x_k}(\tau_i), \alpha_{x_k}(\tau_i) \pm \Lambda\}$ where $0 < \Lambda < 1$, is a step change of offloading amount.

We define a reward $\mathcal{R}^3$ for measuring the overall performance of network selection and offloading decisions given by:

$$\begin{aligned}\mathcal{R}^3(s_1^h, a_1^{\hat{h}}, s_2^m, a_2^{\hat{m}}, s_3^l, a_3^{\hat{l}}) = \gamma_1 T^{x_k}\left(\alpha_{x_k}(\tau_i), a_{(k,e)}(\tau_i)\right) \\ + \gamma_2 E^{x_k}\left(\alpha_{x_k}(\tau_i), a_{(k,e)}(\tau_i)\right) + w_1 F_{k,e}^1(\tau_i) \\ + w_2 F_{k,e}^2(\tau_i) + w_3 F_{k,e}^3(\tau_i) \quad (5.38)\end{aligned}$$

where the first two measures the latency and energy performance. The next three elements are measuring performance in terms of service time, sojourn time, and energy constraints. If the node selection policies violate the constraints, an additional weighted penalty value is added with positive weights w_1, w_2, and w_3.

5.3.2.2 Deep Q Network based solution

Given the complex and dynamic nature of the considered scenario, we consider Q learning as a model-free RL for finding optimal policies. It is one of the highly explored model-free strategies for determining the optimal policy in unknown environments. In this case, policy π_p for problem p maps every state $s \in \mathcal{S}^p$ to action $a \in \mathcal{A}^p$. The Q-learning strategy is based on a state-action function, i.e., Q-function, defined as,

$$Q^{\pi_p}(s', a') = R^p(s', a') + \gamma \sum_{\hat{s} \in \mathcal{S}^p} \hat{\mathcal{P}}_{s'\hat{s}}^p(a') V^{\pi_p}(\hat{s})$$

representing a discounted cumulative reward from state s' when action a' is taken before following the policy π_p. The optimal Q value can be represented as

$$Q^{\pi_p^*}(s',a') = R^p(s',a') + \gamma \sum_{\hat{s}\in\mathcal{S}^p} \hat{\mathcal{P}}^p_{s'\hat{s}}(a')V^{\pi_p^*}(\hat{s})$$

where $V^{\pi_p^*}(\hat{s}) = \min_{\hat{a}\in\mathcal{A}^p} Q^{\pi_p^*}(s',\hat{a})$ with optimal policy π_p^*. The Q values can be estimated through a recursive approach where,

$$Q_{t+1}(s',a') = Q_t(s',a') + \epsilon \cdot \left(r + \gamma \max_{\hat{a}} Q_t(s',\hat{a}) - Q_t(s',a')\right)$$

where ϵ is a learning rate. The Q-function can be estimated through various traditional methods such as the temporal difference approach, tabular methods, etc. In the case of complex RL problems, with high dimensional state/action spaces, the use of novel methods, such as function approximation, can be beneficial in terms of overall training complexity and generalization. The Q-function can be estimated through a neural network-based function approximation technique with $Q(s',a';\theta) \approx Q(s',a')$, where θ represents the weights of the neural network. Through the training process, the values of θ can be adjusted to reduce the mean square error values.

Among several neural network-based deep learning solutions, Deep Q Network (DQN) is one of the highly explored methods to estimate the policy of the RL agents in an unstable environment, mainly due to their simplicity. The considered DQN solution involves two networks (i.e., primary and target Q networks) for each level for estimating the Q function values effectively. The primary network estimates the real/primary Q-value while the target Q-values are estimated through the target network. The RL agent uses the backpropagation and gradient descent processes with mean square error (MSE)-based loss function for reducing the gap between the primary and the target Q-values where the loss function is defined as:

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{\hat{a}} Q_t(s', \hat{a}, \theta^{'}) - Q(s, a, \theta)\right)^2\right] \tag{5.39}$$

where the primary values $Q(s, a, \theta)$ are based upon primary network parameters θ, and $r + \gamma \max_{\hat{a}} Q_t(s', \hat{a}, \theta^{'})$ is the target Q value based upon the target network parameters $\theta^{'}$.

In the considered HRL framework three RL agents exploiting DQN are considered to find a proper coverage node selection policy for $\mathcal{P}^1$, $\mathcal{P}^2$ and $\mathcal{P}^3$ to determine the EC layer, EC node, and offloading amounts. The agent associated with $\mathcal{P}^1$ senses the environment state by processing the data associated with users' demands, layers properties, etc., and selects the proper EC layer. This information is then received by the agent associated with $\mathcal{P}^2$ agent along with the current state information. The $\mathcal{P}^2$ agent then uses this information to select a proper EN for data processing. The intrinsic reward $\mathcal{R}^2$ based upon the feedback signal is used to update the EN selection policies, while the global reward $\mathcal{R}^1$ is considered while updating the $\mathcal{P}^1$ policies. The next agent for $\mathcal{P}^3$ receives the information from $\mathcal{P}^2$ regarding the selected EN and its properties, used while defining the offloading amount. The intrinsic reward $\mathcal{R}^3$ is used to update the policy of $\mathcal{P}^3$ agent. Figure 5.12, details the interaction between different elements of a proposed HRL framework for a network selection and offloading process.

Algorithm 8 details the DQN-based HRL process for the formulated network selection and offloading problem. The process begins with the initialization of the primary/target neural networks (w_H^P, w_H^T), (w_M^P, w_M^T), (w_L^P, w_L^T) associated with $\mathcal{P}^1$, $\mathcal{P}^2$ and $\mathcal{P}^3$ (lines 1-2). The neural networks associated with $\mathcal{P}^1$, $\mathcal{P}^2$ and $\mathcal{P}^3$ have, respectively, $\bar{L}^1$, $\bar{L}^2$ and $\bar{L}^3$ fully connected layers with $n_l, \forall l \in \bar{L}^1/\bar{L}^2/\bar{L}^3$, neurons.

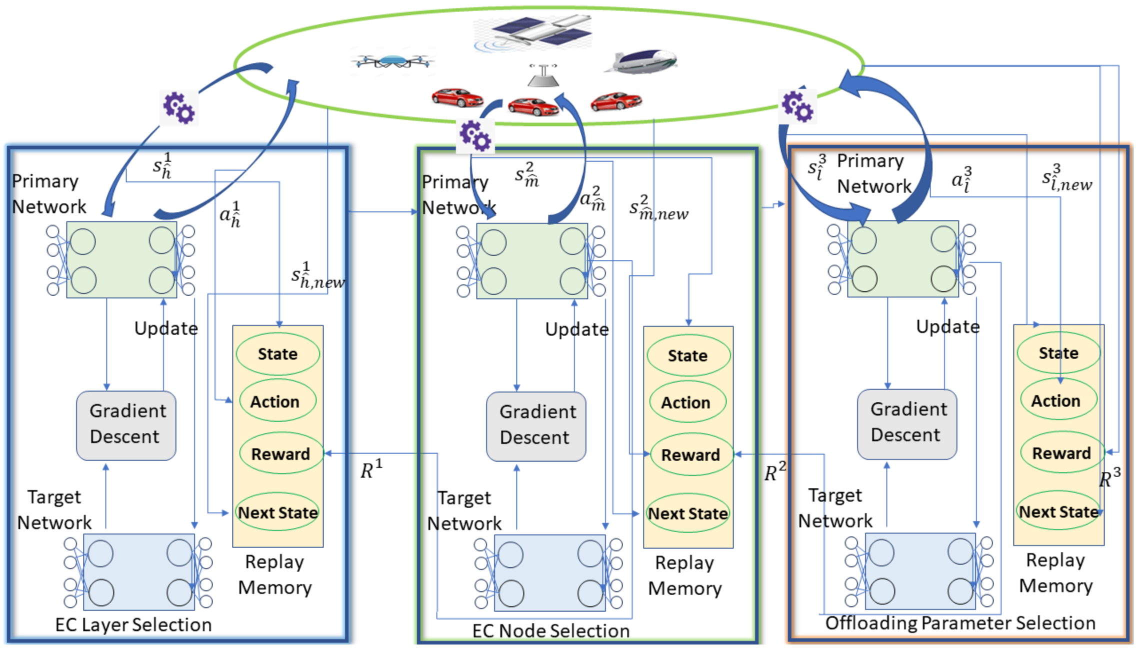


FIGURE 5.12: Functional scheme for the proposed HRL solution.

The training process lasts for $\bar{N}$ episodes with maximum $\mathcal{I}$ epochs per episode. Each training episode begins with the random initial state s_0 (lines 4-5). In each iteration it, DQNs are trained through the batch gradient descent approach described in Algorithm 9. The iteration begins by selecting a higher-level action $a^1_{\hat{h}}$ through Epsilon Greedy Policy (EGP) with parameter e^1 (line 8). The information related to the selected edge layer and its properties is then passed to the next-level DQN i.e., node selection. Based on the selected layer and the state s^2_m action $a^2_{\hat{m}}$ is selected through EGP with parameter e^2 (lines 10-11). The information associated with the selected EN is then communicated with the lower level DQN corresponding to the $\mathcal{P}^3$ for defining the offloading policy. Next, from state s^3_l, action $a^3_{\hat{l}}$ is selected through EGP with parameter e^3 and corresponding intrinsic reward $\mathcal{R}^3$ is determined (lines 12-15). The tuple $\langle s^3_l, a^3_{\hat{l}}, \mathcal{R}^3, s^3_{l,\text{new}} \rangle$ is then saved into replay memory $\mathcal{D}^3$ (line 16). Next, the DQN function is called for updating the parameters of w^P_L, w^T_L (line 17), which is then used to determine the intrinsic reward $\mathcal{R}^2$ (line 18). After that, the DQN for $\mathcal{P}^2$ is updated through a gradient descent approach (lines 19-20). Next, a global reward $\mathcal{R}^1$ is defined (line 21). After that, the DQN for $\mathcal{P}^1$ is updated through a gradient descent approach (lines 22-23). The DQN function defined in Algorithm 9 takes input as replay memory $\mathcal{D}^1/\mathcal{D}^2/\mathcal{D}^3$, the batch size $\hat{k}^1/\hat{k}^2/\hat{k}^3$, learning rate $\epsilon^1/\epsilon^2/\epsilon^3$ and discount factors $\gamma^1/\gamma^2/\gamma^3$. The steps include the random batch selection (lines 1-2), loss value generation (line 3), primary network parameter update through gradient descent step (line 4), and target network parameter update step (line 6).

Algorithm 8 HRL for Network Selection and Offloading

```
Input: $\mathcal{S}^1, \mathcal{A}^1, \mathcal{S}^2, \mathcal{A}^2, \mathcal{S}^3, \mathcal{A}^3, \mathcal{I}, \bar{N}, e^1, e^2, e^3, |\mathcal{D}^1|, |\mathcal{D}^2|, |\mathcal{D}^3|, \epsilon^1, \epsilon^2, \epsilon^3, \gamma^1, \gamma^2, \gamma^3, \bar{it}$
Output: $w_H^P, w_M^P, w_L^P$
 1: Initialize $w_H^P$, $w_M^P$, $w_L^P$
 2: Duplicate policy networks to Target Networks,
            i.e., $w_H^T = w_H^P$, $w_M^T = w_M^P$, $w_L^T = w_L^P$
 3: for all $ep = 1, \ldots, \bar{N}$ do
 4:     Select Random $s_0$
 5:     $s_h^1 \leftarrow s_0, it = 0$
 6:     while $it \neq \mathcal{I}$ do
 7:         $it = it + 1$
 8:         Select action $a_{\hat{h}}^1 \in \mathcal{A}^1$ with probability $e^1$
 9:         Determine next state $(s_{h,\text{new}}^1)$
10:         Select $s_m^2$ based upon selected EC layer and local properties.
11:         Select action $a_{\hat{m}}^2 \in \mathcal{A}^2$ with probability $e^2$
12:         Determine next state $(s_{m,\text{new}}^2)$
13:         Select $s_l^3$ based upon selected coverage node.
14:         Select action $a_{\hat{l}}^3 \in \mathcal{A}^3$ with probability $e^3$
15:         Find next state $s_{l,\text{new}}^3$ and reward $R^3$
16:         Store $\mathcal{D}^3 \leftarrow \langle s_l^3, a_{\hat{l}}^3, R^3, s_{l,\text{new}}^3 \rangle$
17:         $w_L^P, w_L^T$ =DQN$(\mathcal{D}^3, \hat{k}^3, w_L^P, w_L^T, it, \bar{it}, \epsilon^3, \gamma^3)$
18:         Find Intrinsic Reward $\mathcal{R}^2$
19:         Store $\mathcal{D}^2 \leftarrow (s_m^2, a_{\hat{m}}^2, R^2, s_{m,\text{new}}^2)$
20:         $w_M^P, w_M^T$ =DQN$(\mathcal{D}^2, \hat{k}^2, w_M^P, w_M^T, it, \bar{it}, \epsilon^2, \gamma^2)$
21:         Find Global Reward $\mathcal{R}^1$
22:         Store $\mathcal{D}^1 \leftarrow (s_h^1, a_{\hat{h}}^1, R^1, s_{h,\text{new}}^1)$
23:         $w_H^P, w_H^T$ =DQN$(\mathcal{D}^1, \hat{k}^1, w_H^P, w_H^T, it, \bar{it}, \epsilon^1, \gamma^1)$
24:     end while
25: end for
26: return $w_H^P, w_M^P, w_L^P$
```

Algorithm 9 DQN Function

```
Input: $\mathcal{D}, k, w^P, w^T, i, \bar{i}, \epsilon, \gamma$
Output: $\{w^P, w^T\}$
1: function DQN($\mathcal{D}, k, w^P, w^T, i, \bar{i}, \epsilon, \gamma$)
2:     Select Random batch of of $k$ samples from $\mathcal{D}$
3:     Preprocess and pass the batch to $w^P$
4:     Find Loss between primary and Target Q values using (5.39)
5:     With gradient descent step update $w^P$
6:     Update $w^T$ if $rem(i, \bar{i}) = 0$
7: end function
8: return $\{w^P, w^T\}$
```

5.3.2.3 Benchmark Solutions

Three different benchmark solutions are considered for comparison purposes.

Probabilistic VU-EN assignment In this Random Method (RM) $\forall v_k \in \mathcal{V}$ we select the node e randomly. The probability of v_k selecting EN e is given by:

$$Pr\{a_{k,e}(\tau_i) = 1\} = \frac{1}{\mathcal{E}_k(\tau_i)} \tag{5.40}$$

Such random EN selection policies can have a limited impact in terms of task processing latency due to improper offloading policies. Such random solutions can be implemented easily, without guaranteeing optimal load distribution and utilization of edge resources.

Position-based VU-EN Assignments In this Distance-based Method (DM), each nearby competing VU is allocated to the ENs based on the available distance before it passes through the ENs coverage range and the distance between VU and EN. Thus, $\forall v_k \in \mathcal{V}$:

$$a_{k,e}(\tau_i) = 1 \Leftrightarrow \frac{D_{k,e}(\tau_i)}{d_{k,e}(\tau_i)} = \max_{e \in \mathcal{E}_k(\tau_i)} \left\{ \frac{D_{k,e}(\tau_i)}{d_{k,e}(\tau_i)} \right\} \tag{5.41}$$

Such a static approach can reduce the computation complexity of an offloading process with simplified approaches. However, with these fixed policies, the utilization of distributed edge resources can be limited, and possibly several VUs can select the same edge facilities without taking into account the local environment parameters. This can result in high computation and communication costs during the offloading process, especially with the presence of a large number of VUs.

Local Computation (LC) In this case, VUs perform the task computation by themselves without employing any EN. Such an approach can reduce the latency requirements in terms of data transmission toward ENs. However, without the parallel computation process between the EC nodes and resource-limited VUs, the computation latency can grow significantly, unable to satisfy the service latency demands.

5.3.3 Numerical Results

The proposed HRL-based solution and benchmark approaches are simulated over the Python environment with ML-related libraries, including NumPy, Pandas, Math, and Matplotlib. A service area 2 km road network is considered with randomly located VUs. A set of users between 200 and 2000 are considered with a probability of being active $P_a = 0.1$. Users can reply on six possible different services. Cloud facility is able to provide all six services. On the other hand, LEO satellites are able to host 4 services. HAP nodes are equipped with 3 services while RSU and UAVs can provide 2 services each. The services are installed randomly and in advance.

For the DQN simulation, primary and target networks with layers $(\bar{L}^1, \bar{L}^2, \bar{L}^3) = 4$ are considered with the learning parameters $(e^1, e^2, e^3) = 0.7$, $(\mathcal{D}^1, \mathcal{D}^2, \mathcal{D}^3) = 4000$,

Table 5.2: Simulation parameters

Coverage $(R_{r_n}, R_{l_u}, R_{h_p})$	(50, 200, 1000) m
VU Computation Cap. $(c_{v,k} \cdot f_{v,k})$	8 GFLOPS
RSU Computation Cap. $(c_{r_n} \cdot f_{r_n})$	20 GFLOPS
LAP Computation Cap. $(c_{l_u} \cdot f_{l_u})$	20 GFLOPS
HAP Computation Cap. $(c_{h_p} \cdot f_{h_p})$	40 GFLOPS
LEO Computation Cap. $(c_{s_1} \cdot f_{s_1})$	60 GFLOPS
Altitude (h_l, h_h, h_s)	(1, 100, 2000) Km
Bandwidth $(B_{r_n}, B_{l_u}, B_{h_p}, B_{s_1})$	(20, 20, 50, 100) MHz
VU Speed Range $(\vec{v}_{\min}, \vec{v}_{\max})$.	(8 m/s, 14 m/s)
VU Power $(P_{c,v_k}, Pt_{v_k}, Pr_{v_k})$	(1.3, 1.5, 1.3) W
Task Size (D_{x_k})	5 MB
Task Latency Req. $(\bar{T}_{x_k})$	4 Sec
Task Computation Req. (Ω_{x_k})	$(10^3 \cdot D_{x_k})$ Flops
Elevation angle (θ_m)	$\mathcal{U}(20^0, (\pi/2 + 20)^o)$
Weighting Coefficients (η_1, η_2, w_k)	(0.5, 0.5, 0.5)

$(\epsilon^1, \epsilon^2, \epsilon^3) = 0.05$, $(\gamma^1, \gamma^2) = 0.98$. Also, the learning process includes $\bar{N} = 50$ with $\mathcal{I} = 10^3$ and $\bar{it} = 50$. Main simulation parameters are provided in Table 5.2.

Joint Latency and Energy Cost In this work, we aim to minimize the overall latency and energy cost associated with vehicular task processing operations over EC facilities. Fig. 5.13 presents the average cost required for different methods. The cost values can be impacted by the layer/node selection and offloading decisions. Regarding the considered benchmark methods, RM randomly selects the layer/node for processing the VUs' complete task. With such a suboptimal approach, RM requires a much higher amount of latency and energy costs for processing the VUs data. The distance-based benchmark solution, DM, selects the EN according to the distance measures. However, such a static approach can limit the overall flexibility of offloading portion in terms of exploration of edge resources. In addition to this, with the presence of multiple VUs, services, and edge facilities, VUs offloading policies should be adapted accordingly. With these issues, the DM approach also requires higher processing costs. The local computation case has a static cost, since it is

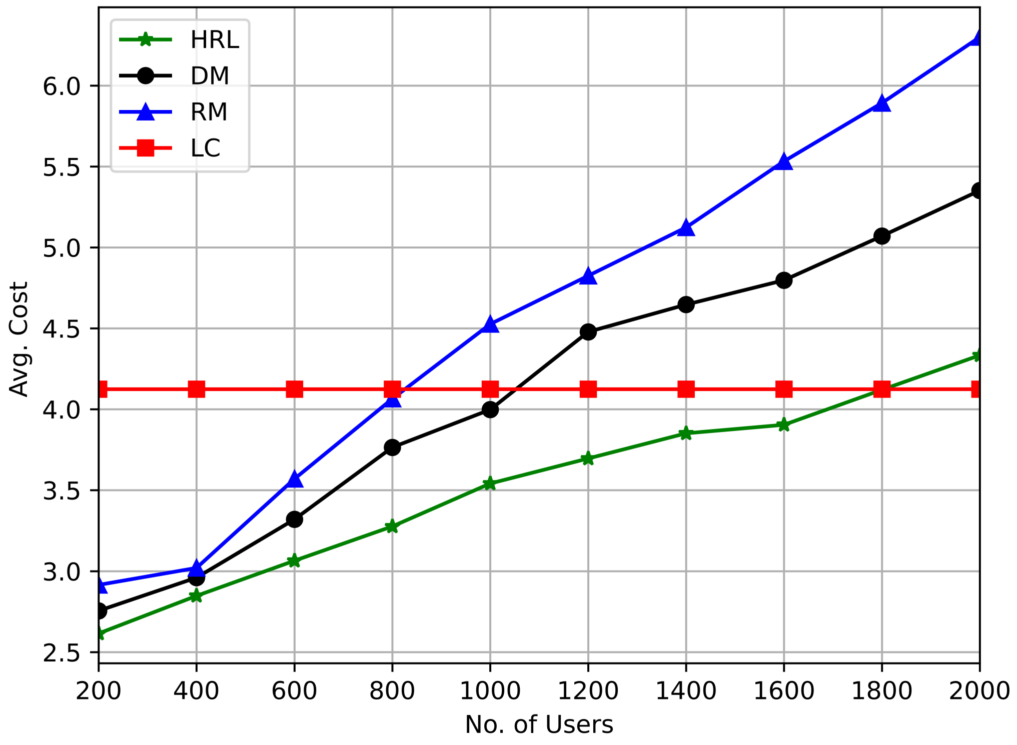


FIGURE 5.13: Joint Latency and Energy Cost.

unable to take advantage of distributed computing environments, resulting in much higher latency costs. The proposed HRL method is able to reduce the overall cost through proper layer/node selections and offloading process. With the help of local environment data, HRL policies can be adapted to unable efficient edge processing operations.

Average Latency Cost Given the latency-constraint nature of vehicular scenarios, it is important to analyze the performance of the proposed HRL approach in terms of latency requirements. In Fig. 5.14, we present the latency performance for different solution methods highlighting the performance gain for the proposed HRL case. It can be seen that with suboptimal node/layer selection policies, RM and DM methods require higher latency costs. With the growing number of VUs, both

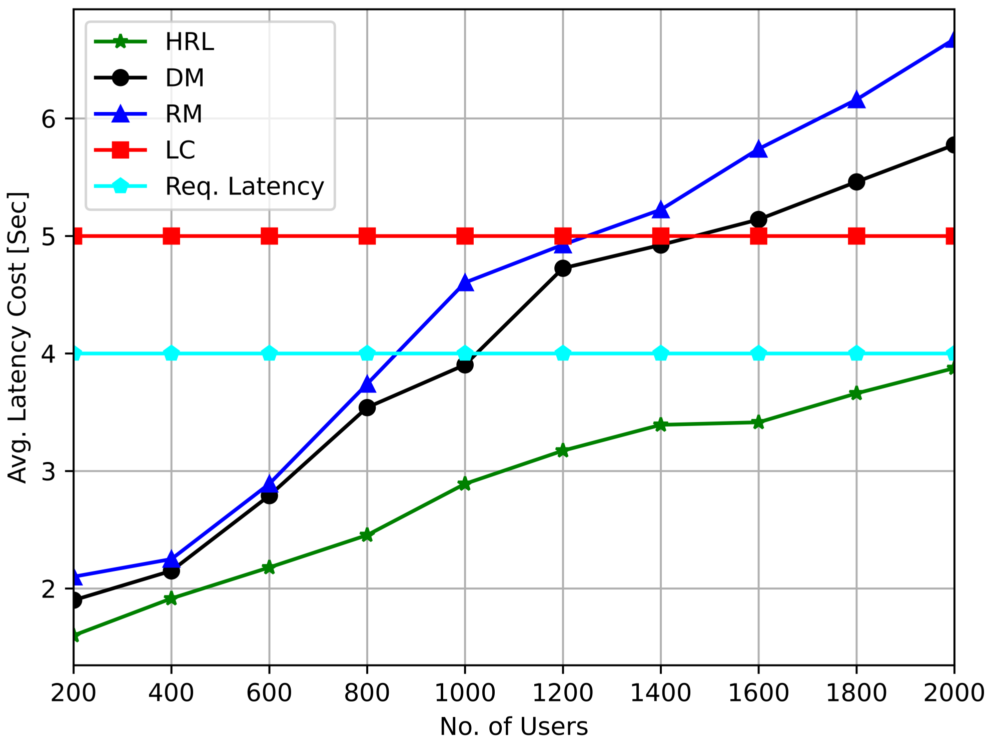


FIGURE 5.14: Total Task Processing Latency.

DM and RM performance decreases and even becomes worse than the LC approach. With the LC method, VUs are unable to satisfy the service latency requirements. The proposed HRL approach can reduce the latency cost significantly through a proper offloading process.

Average Energy Cost. With the presence of dynamic VUs and different non-terrestrial platforms, analyzing the energy cost becomes important. In Fig. 5.15, we have presented the energy requirements for different methods as a weighted average of both VU side and energy side energy elements. For the LC approach, the overall energy requirements can be reduced due to the local computation process. However, as presented before, it suffers from unfeasible computation latency due to reduced computation resources. With long-distance transmissions and improper

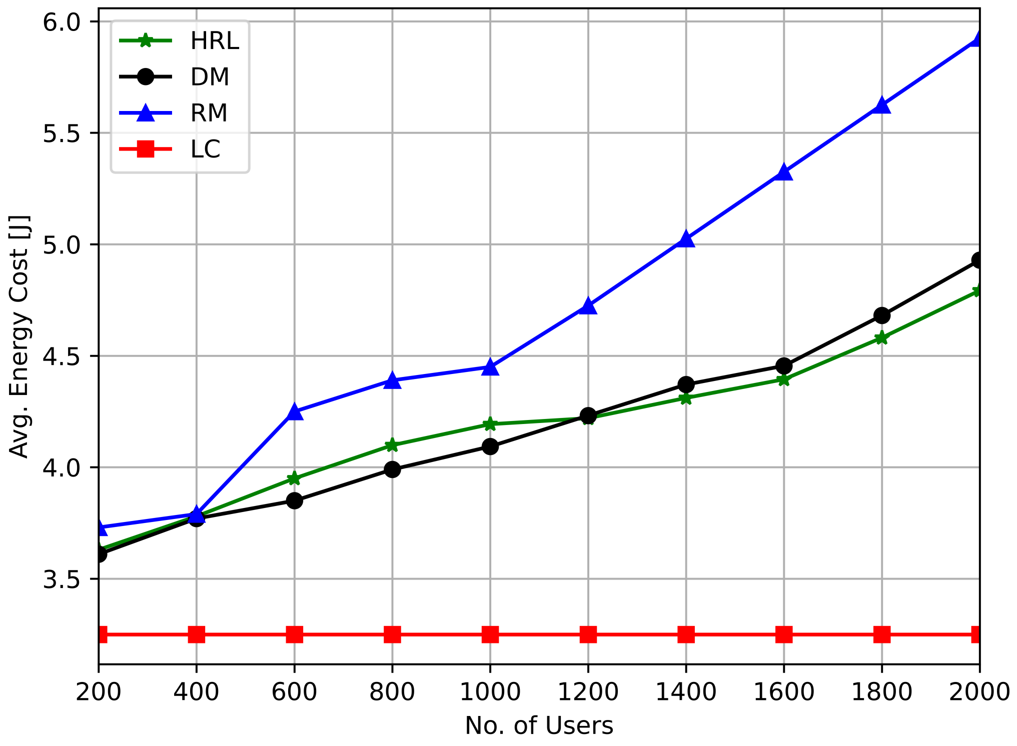


FIGURE 5.15: Energy Requirements.

offloading parameters, RM and DM approaches suffer from higher energy requirements, especially with the growing number of users. The HRL approach requires higher energy costs compared to the LC method with the additional data transmission/reception, and waiting steps. However, it can gain an advantage in terms of latency and handover requirements.

Average Number of Service Time Failures

Apart from the latency and energy costs associated with the offloading process, it is also important to analyze the performance of the proposed solution in terms of service latency requirements. In Fig. 5.16 we have presented the performance of different methods in terms of satisfaction of service latency constraint defined

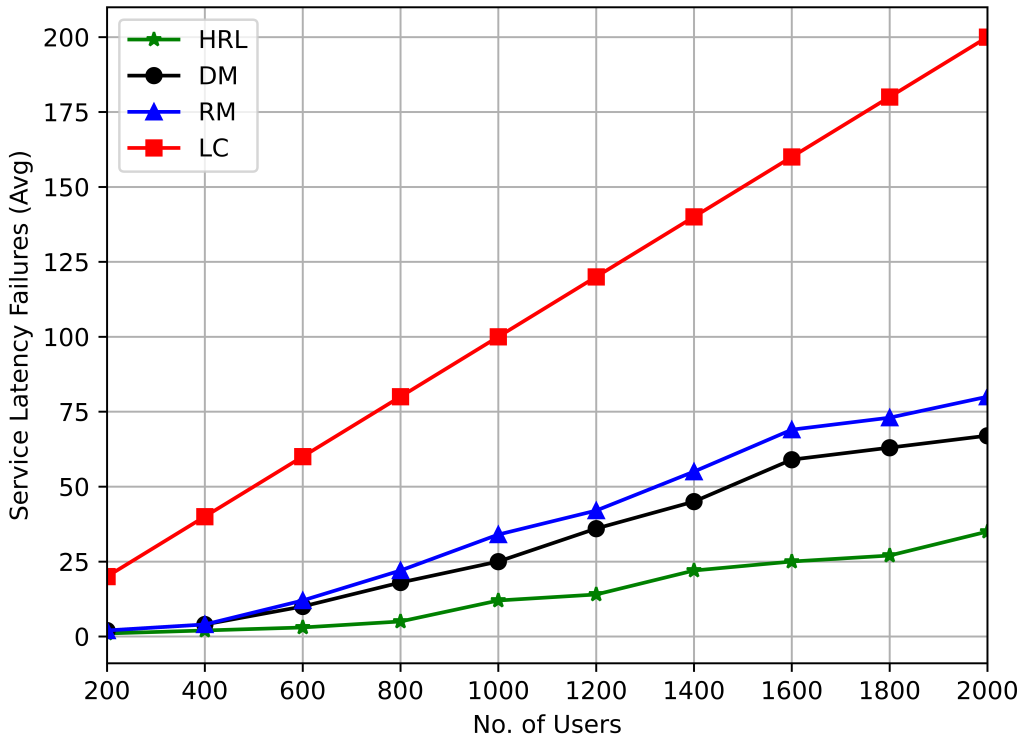


FIGURE 5.16: Service Time Failures.

in (5.34b). With the complete local process, the LC approach is unable to satisfy the service latency requirements. Similarly, the other two benchmark solutions (RM, DM) with static offloading processes, have a large number of service latency failures. The proposed HRL solution can enable efficient task processing based on VUs local environments and task requirements, thus reducing the number of service latency failures. This shows the significance of the proposed HRL solution to enable latency-critical vehicular services.

Average Number of Sojourn Time Failures. With the presence of dynamic VUs and non-terrestrial nodes, it is important to analyze the performance of the proposed solution in terms of a mobility constraint defined in (5.32). With the mobility constraint, the offloading process should be completed before VUs passes

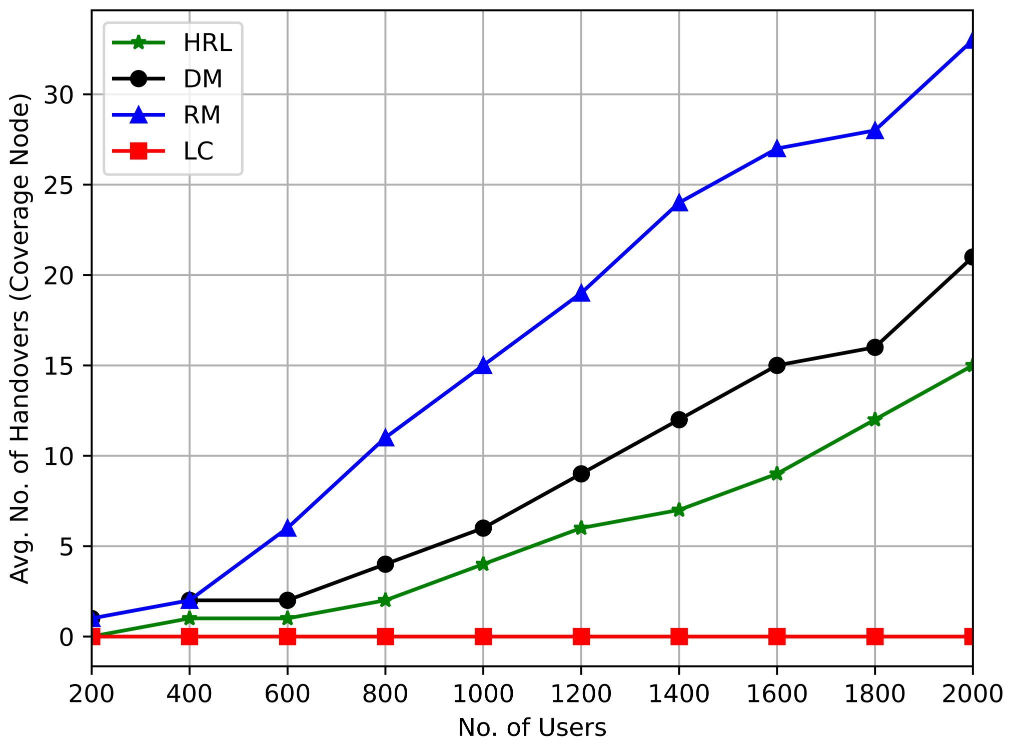


FIGURE 5.17: Sojourn Time Failures.

through the coverage area of a selected EN. Thus the performance can be impacted by the selected layer/node and the offloading portions. As shown in Fig. 5.17, with the random node selection approach, the RM method suffers from many mobility constraint failures. With a distance-based static approach, DM method can have a reduced number of failures, however, performance can be suboptimal. The proposed HRL solution can reduce the mobility constraint failures significantly with the proper offloading decisions in terms of offloading node and amount selections.

Average Number of Service Handovers Required. For the case of considered multi-service vehicular scenario, it is important to analyze the performance in terms of a number of service handover requirements. If the selected EN is unable to provide the requested service, it needs to take additional measures for satisfying the user

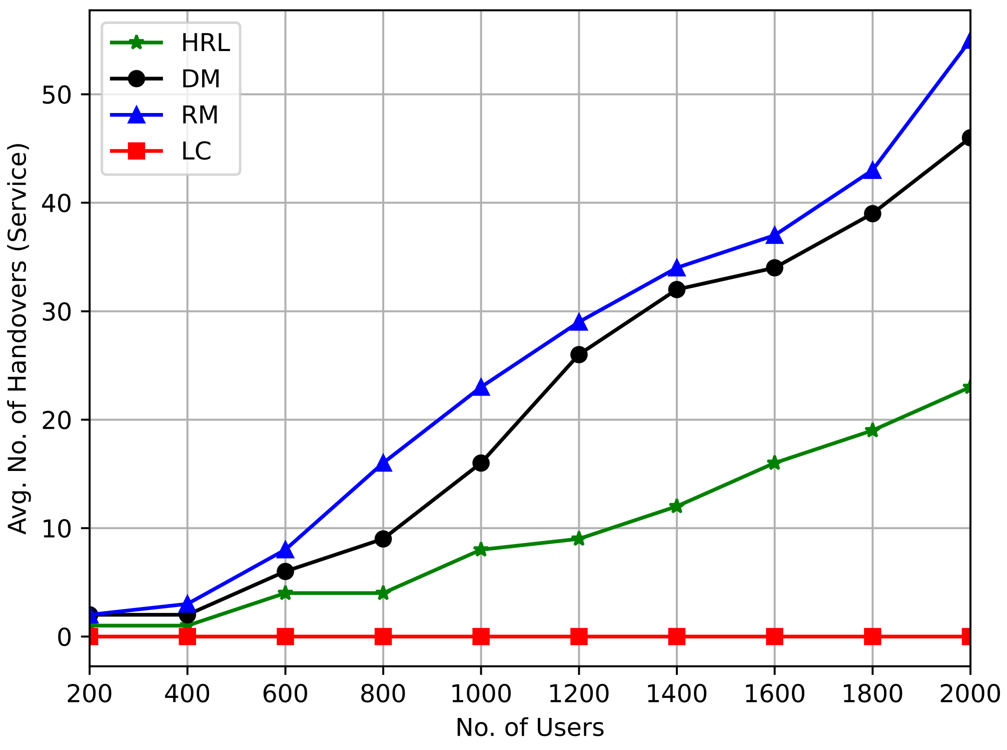


FIGURE 5.18: Service Handover Requirements.

demands. In a considered simulation, ENs that are unable to provide the demanded service, relay the user data to the cloud facility, which is having all the services preinstalled. In Fig. 5.18, we present the performance in terms of such service handover requirements, for different methods. The proposed HRL method can have superior performance compared to the other solutions with static or random node selection strategies.

5.3.4 Conclusions

In this work, we have considered a joint network selection and computation offloading problem over a joint T-NT network for processing vehicular data. A multi-service vehicular scenario is considered allowing VUs to request different services through multiple EC environments. A constrained optimization problem for minimizing the overall latency and energy costs through proper network selection and offloading decisions is formed. The considered problem is modeled as a multi-level sequential decision process and the HRL approach is used to solve it. In particular deep learning-based strategies are applied to find optimal network selection and offloading policies. The performance of a proposed method is analyzed through Python-based solutions and compared with several other benchmark solutions. The considered approach can further be extended in the future for jointly solving the service placements, network selection, and offloading problems effectively.

5.4 Joint Service Placement, Network Selection and Offloading: Multi-time Scale Approach

In joint T-NTN-based VEC networks, composed of multiple layers, each VU has the option to select EC nodes from various layers. Additionally, each node of these platforms, with its storage resources, can hold a set of services. With this in mind, in such multi-user vehicular scenarios, with multiple T/NT layers, service placement, network selection, and computation offloading decisions can be optimized jointly. It is interesting to notice that the service placement, network selection, and offloading decisions can be performed over different time scales, based on specific demands, network topologies, and user requirements. For example, the service placement operation needs a larger time interval and can only be performed/updated at longer time scales mainly due to centralized controller operations and longer service activation time [112]. On the other hand, the network selection operation, based on the VUs' dynamicity and nearby environments, may require a moderate amount of time for establishing a proper connection between VUs and nearby reliable edge servers [68].

Finally, task offloading operations will be based upon the VUs task requirements, and can be performed at shorter time scales, especially in the case of latency-critical applications [86]. With these issues in mind, in the past, researchers have mainly focused on either of these three problems and proposed solutions by making some assumptions about the other two [80]. On the other hand, in some cases, joint optimization of either of these problems has been considered, while assuming the same time scales [69]. However, such solutions cannot be considered optimal since this process may require decisions made at different time scales, impacting each other's performance.

With these issues in mind, in this work, we aim to solve the joint service placement, network selection, and offloading problem over the VN environment aimed at minimizing the latency and energy costs by considering proper time scales. For this, we first model the latency and energy elements involved in this process and map them on a constrained optimization problem over a dynamic VN. Next, we exploit an RL approach to solve the problem, in particular by modeling it as a Markov Decision Process (MDP). With the involvements of multiple time scales, we consider a multi-time scale MDP approach by modeling three different MDPs impacting each other's decisions for the considered problem. This allows us to solve the problem at different time scales effectively.

The main contributions of this work can be summarized in the following points:

- **Multi-time Scale Approach:** A system model is defined with a multi-time scale approach for the service placement, network selection, and computation offloading process with dynamic VUs. Further, a constrained optimization problem is formed to minimize a proper cost function, including also latency and energy terms by performing a dynamic service placement, network selection, and offloading process.

- **MDP Solutions:** The service placement, network selection, and offloading problems are modeled as a sequential decision-making process through proper multi-dimensional MDP models. A multi-time scale MDP process is adapted for enabling decision-making at different time scales and optimal policy is determined through the deep Q-learning approach.

- **Performance Evaluation:** The simulation results of the proposed methods are compared with a set of benchmark methods and their effectiveness is evaluated. In the end, proper conclusions are drawn based on the findings.

5.4.1 System Model and Problem Formulation

We focus on an IoV scenario with multiple EC layers and randomly distributed VUs on the road scenario. We consider a multi-layered ($l = 0, \dots, L$ with $L = 3$) joint air-ground network, composed of HAPs ($l = 3$), UAVs (i.e., LAP nodes) ($l = 2$), RSUs ($l = 1$), deployed along the road paths, and randomly distributed VUs ($l = 0$) traveling on a road in either directions, where $\mathcal{V} = \{v_1, \dots, v_m, \dots, v_M\}$, $\mathcal{R} = \{r_1, \dots, r_n, \dots, r_N\}$, $\mathcal{U} = \{u_1, \dots, u_p, \dots, u_P\}$, correspond to the sets denoting M VUs, N RSUs and P UAVs, respectively. Additionally, one HAP node is denoted as H_h. In the considered vehicular scenario, VUs can request services characterized by different requirements. By assuming that $\mathcal{S} = \{S_1, \dots, S_s, \dots, S_{\bar{S}}\}$ is the set of all the possible services that can be provided, due to the limited available resources, the generic jth EN from lth layer can provide only a subset of services equal to $\hat{\mathcal{S}}_j^l \subset \mathcal{S}$.

The system is modeled in a time-discrete manner, and the network parameters are supposed to be constant over each time interval τ, where τ_i identifies the ith time interval, i.e., $\tau_i = \{\forall t | t \in [i\tau, (i+1)\tau]\}$. The generic mth VU, is characterized by a processing capacity equal to $c_{v,m}$ Floating Point Operations per Second

(FLOPS) per CPU cycle, while its CPU frequency is $f_{v,m}$. Each VU is supposed to be able to communicate on a bandwidth $B^{\mathrm{rsu}}_{v,m}(\tau_i)$ with each RSU, with a bandwidth $B^{\mathrm{LAP}}_{v,m}(\tau_i)$ with each UAV and with a bandwidth $B^{\mathrm{HAP}}_{v,m}(\tau_i)$ with the HAP. Each $v_m \in \mathcal{V}$ is supposed to be active in each time interval with a probability p_a within which it generates a computation task request $\rho_m(\tau_i)$ identified through the tuple $\left\langle D_{\rho_m}, D^r_{\rho_m}, \Omega_{\rho_m}, T_{\rho_m}, S_{\rho_m} \right\rangle$ corresponding to a task with size D_{ρ_m} Byte, expected to give in output a result with size $D^r_{\rho_m}$ Byte, requesting Ω_{ρ_m} CPU execution cycles and a maximum execution latency T_{ρ_m}. Here, $S_{\rho_m} \in \mathcal{S}$ corresponds to a specific service requested by VU v_m that belongs to a set of services $\mathcal{S}$ provided by the network service provider. In addition, the average request rate for the sth service is modeled through the Zipf distribution function given by $\lambda_s(\tau_i) = 1/\kappa s^{\beta}$ where, $\kappa = \sum_s 1/s^{\beta}$ with $\beta \in [0, 1]$ being the popularity skew index [113].

The nth RSU, supposed to be in a fixed position with a coverage radius $R_{r,n}$, is characterized by a processing capability equal to $c_{r,n}$ FLOPS per CPU cycle, with CPU frequency $f_{r,n}$, CPU cores $\mathcal{L}_{r,n}$, and communication capabilities, supposed to be identified through a communication technology able to cover the VUs on ground with an overall bandwidth $B_{r,n}$. Each RSU can provide EC services to the VUs in its coverage space. As mentioned before, with its limited resources, r_n can store only a subset of services $\hat{\mathcal{S}}^1_{r_n} \subset \mathcal{S}$. In addition, the area is supposed to be under the coverage of multiple UAVs with pth UAV at altitude $\bar{h}_{u,p}$ and coverage radius $R_{u,p}$. The pth UAV is supposed to move with a relatively slow speed compared with VUs and is characterized by a processing capability equal to $c_{u,p}$ FLOPS per CPU cycle, with CPU frequency $f_{u,p}$ and $\mathcal{L}_{u,p}$ CPU cores. In addition, its communication capabilities are supposed to be identified through a communication technology able to work on a bandwidth $B_{u,p}$. Each UAV can serve a set of VUs and RSUs in its coverage space. Here, the pth UAV, with its limited resources, can provide up to $\hat{\mathcal{S}}^2_{u_p} \subset \mathcal{S}$ services to

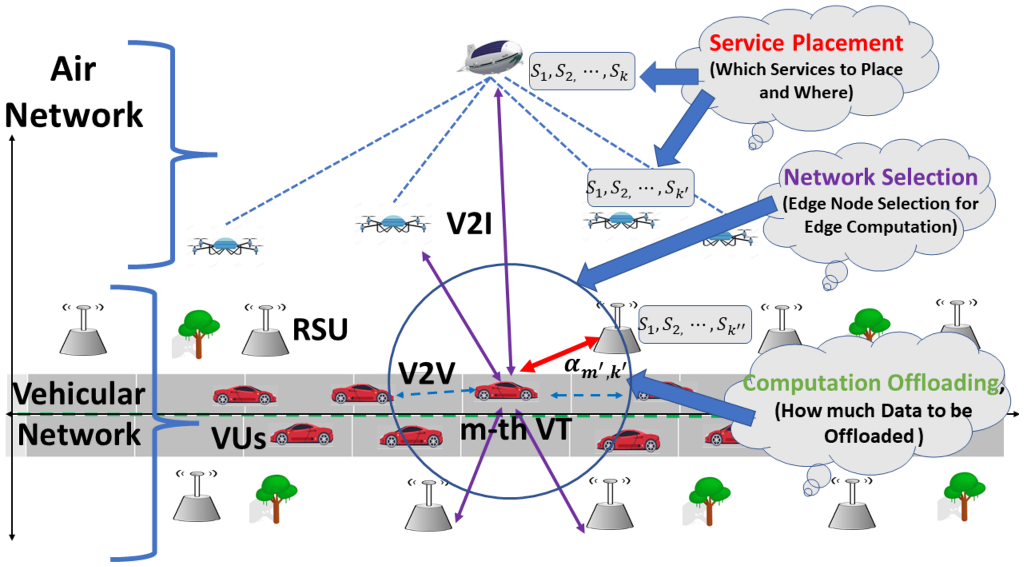

Air Network
Vehicular Network
$S_1, S_2, \cdots, S_k$
Service Placement
(Which Services to Place and Where)
$S_1, S_2, \cdots, S_{k'}$
Network Selection
(Edge Node Selection for Edge Computation)
V2I
RSU
$S_1, S_2, \cdots, S_{k''}$
V2V
$\alpha_{m',k'}$
Computation Offloading
(How much Data to be Offloaded)
VUs
m-th VT

FIGURE 5.19: The T/NT Integrated Scenario.

the VUs. Being a centralized node with powerful computation and communication resources, we assume that the HAP node can provide the whole service set $\mathcal{S}$ to the VUs in its coverage range of R_h meters, with computation capacity equal to c_h FLOPS per CPU cycle, with CPU frequency f_h and $\mathcal{L}_h$ CPU cores and having bandwidth B_h. Also, the HAP node is located at altitude $\bar{h}_h$. Fig. 5.19 shows the basic system elements and various communication links between them.

5.4.1.1 VU Mobility Model

Compared with the highly dynamic VUs, the aerial network platforms move more slowly and often have negligible impacts over VUs overall mobility parameters, i.e., relative distance speed locations, etc. Also, these platforms can follow predefined mobility patterns based on operators' settings. Therefore, in this work, we consider that the air networking nodes (i.e., UAVs and HAP) are located at a fixed position in a given interval of time, while VUs move with a variable speed $\vec{v}_m(\tau_i)$. We suppose that the VUs' speed is bounded within $\vec{v}_{\min}$ and $\vec{v}_{\max}$ while the the mth VU instantaneous speed is modeled through a truncated normal distribution density function [114]:

$$f\left(\vec{v}_m(\tau_i)\right) = \begin{cases} \dfrac{2e^{\frac{-(\vec{v}_m(\tau_i)-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}\left(\mathrm{erf}\left(\frac{\vec{v}_{\max}-\mu}{\sigma\sqrt{2}}\right) - \mathrm{erf}\left(\frac{\vec{v}_{\min}-\mu}{\sigma\sqrt{2}}\right)\right)}, \\ \qquad\qquad \vec{v}_{\min} \leq \vec{v}_m(\tau_i) \leq \vec{v}_{\max} \\ 0, \quad \text{else} \end{cases} \tag{5.42}$$

where μ and σ are the mean and standard deviation of the vehicles speed, and $\mathrm{erf}(x)$ is the Gauss error function over x. The path length, within which the mth VU remains under the coverage of the jth node (i.e., RSU, UAV or HAP), is given by $D_{v_m,j}(\tau_i) = \sqrt{R_j^2 - \left(y_j - y_{v_m}(\tau_i)\right)^2} \pm \left(x_j - x_{v_m}(\tau_i)\right)$ where, $\left(x_{v_m}(\tau_i), y_{v_m}(\tau_i)\right)$ is the location of the mth VU at τ_i and $\left(x_j, y_j\right)$ is the projection over the ground of a generic jth node, which can be RSU, UAV or HAP. The available sojourn time for the mth VU with respect to a generic jth node is $T^{\mathrm{soj}}_{v_m,j}(\tau_i) = \frac{D_{v_m,j}(\tau_i)}{|\vec{v}_m(\tau_i)|}$.

5.4.1.2 Multi-time Scale Approach

The service placement operations often require larger time intervals for updating over different EN mainly due to the virtual service activation latency, longer backhaul delays, etc. [115]. Here, the service placement operations are performed at discrete time intervals lasting Δ^{sp}, where τ_i^{sp} identifies the ith time interval, i.e., $\tau_i^{\mathrm{sp}} = \{\forall t | t \in [i\Delta^{\mathrm{sp}}, (i+1)\,\Delta^{\mathrm{sp}}]\}$. The network selection decisions can be made at a moderate time scale compared with the service placement problem. The time scale is modeled in a time-discrete manner, with time interval Δ^{ns}, where τ_i^{ns} identifies the ith time interval, i.e., $\tau_i^{\mathrm{ns}} = \{\forall t | t \in [i\Delta^{\mathrm{ns}}, (i+1)\,\Delta^{\mathrm{ns}}, 0 < (i+1)\,\Delta^{\mathrm{ns}} \leq \Delta^{\mathrm{sp}}]\}$. Note that the maximum number of time steps is a function of Δ^{sp}, corresponding to the time step of the service placement problem. With the high dynamicity and the frequent task requests, the computation offloading decisions should be performed on a shorter time scale. The time scale for the offloading process is time-discrete with time interval Δ^{off}, where τ_i^{off} identifies the ith time interval, i.e., $\tau_i^{\text{off}} = \left\{\forall t | t \in \left[i\Delta^{\text{off}}, (i+1)\,\Delta^{\text{off}}, 0 < (i+1)\,\Delta^{\text{off}} \leq \Delta^{\text{ns}}\right]\right\}$. Note that the maximum number of offloading time steps is based upon the Δ^{ns} value, corresponding to the time step of the network selection problem.

Over time, vehicular service placement needs to be updated for serving the end-users based on their demands as well as for the proper resource utilization over the EC facilities. We define a binary service placement parameter $b(S_s, j, l, \tau_i^{\text{sp}})$ as,

$$b(S_s, j, l, \tau_i^{\text{sp}}) = \begin{cases} 1 & S_s \in \hat{\mathcal{S}}_j^l \\ 0 & \text{else} \end{cases}$$

with,

$$\sum_{s=1}^{\bar{S}} b(S_s, j, l, \tau_i^{\text{sp}}) \leq |\hat{\mathcal{S}}_j^l|, \quad \forall j \tag{5.43}$$

where $b(S_s, j, l, \tau_i) = 1$ models the network operators decision of placing the service S_s on the jth node at τ_i^{sp}. Notice that the service placement remains the same over the Δ^{sp} time interval, in which multiple network selection and offloading steps are performed.

Based on their limited coverage ranges, each VU can be covered by several RSUs, UAVs, and one HAP node. Here, we define a decision matrix $\mathbf{A}(M, J(l), l, \tau_i^{\text{ns}}) = \left\{a_{(v_m,j,l)}(\tau_i^{\text{ns}}) \in \{0,1\}\right\}$ with dimension $M \times J(l)$, where $J(l)$ is the amount of ENs in the lth layer. Here, $a_{(v_m,j,l)}$ is equal to 1 if the mth VU selects the jth EN from layer l for offloading its task, otherwise it takes value 0. VUs can either select RSU ($l = 1$), UAV ($l = 2$), or HAP ($l = 3$) for offloading their data. Also, to avoid additional complexity we consider that each VU can be assigned to only one EN which can be RSU, UAV, or HAP during the offloading process. Thus,

$$\sum_{l=1}^{L}\sum_{j=1}^{J(l)} a_{(v_m,j,l)}(\tau_i^{\text{ns}}) = 1, \quad \forall v_m \tag{5.44}$$

The number of VUs requesting services from the jth EN is given by $K_{j,l}(\tau_i^{\text{ns}}) = \sum_{m=1}^{M} a_{(v_m,j,l)}(\tau_i^{\text{ns}})$. With their limited resources, ENs can provide services to the VUs before task communication and computation costs become unbearable. We consider that $K_{j,l}^{\max}$ is the maximum number of VUs that can access to the services of the jth node.

We assume to perform partial offloading, where tasks can be split and processed remotely while the remaining portion is processed locally [80]; the offloaded portion by the mth VU at τ_i^{off} is identified as $\alpha_{\rho_m}(\tau_i^{\text{off}}) \in [0,1]$. With multiple VUs requesting services, during the offloading process, the following constraints need to be taken into account:

$$\begin{cases} K_{j,l}(\tau_i^{\text{ns}}) \le K_{j,l}^{\max} & \text{(5.45a)} \\ \sum_{m=1}^{K_{j,l}(\tau_i^{\text{ns}})} c_{j,l}^{\rho_m}(\tau_i^{\text{ns}}) \cdot f_{j,l}^{\rho_m}(\tau_i^{\text{ns}}) \le (\mathcal{L}_{j,l} \cdot c_{j,l} \cdot f_{j,l}) & \text{(5.45b)} \\ \sum_{m=1}^{K_{j,l}(\tau_i^{\text{ns}})} b_{j,l}^{\rho_m}(\tau_i^{\text{ns}}) \le B_{j,l} & \text{(5.45c)} \end{cases}$$

$$\forall i;\, j = 1, \dots, J(l);\, l = 1, \dots, L$$

where $c_{j,l}^{\rho_m}(\tau_i^{\text{ns}}) \cdot f_{j,l}^{\rho_m}(\tau_i^{\text{ns}})$ is the processing capacity of the jth EN from layer l assigned to the mth VUs task, $b_{j,l}^{\rho_m}(\tau_i^{\text{ns}})$ is the communication resource assigned to the VU for communicating with the jth EN. We consider that the EN resources are shared equally among the requesting VUs. Eq. (5.45) models an upper bound on the number of users connected, processing capacity and the communication resources of the ENs.

Task Computation Model The generic expression for the time and energy spent for the ρ_mth task computation on a jth device is given by [69]:

$$T_{c,j}^{\rho_m} = \frac{\Omega_{\rho_m}}{c_j^{\rho_m} f_j^{\rho_m}}, \quad E_{c,j}^{\rho_m} = T_{c,j}^{\rho_m} P_{c,j} \tag{5.46}$$

where $c_j^{\rho_m}$, $f_j^{\rho_m}$ and $P_{c,j}$ are the number of FLOPS, CPU-frequency per CPU-cycle assigned to the mth user, and computation power, respectively, whether j identifies a VU (v_m), a RSU (r_n), UAV (u_p) or a HAP (H_h).

Task Communication Model For the case of a partial computation offloading, the transmission time and energy between a generic node j and a generic node k for task ρ_j is given by[6] $T_{tx,jk}^{\rho_j}(\tau_i) = \frac{D_{\rho_j}}{r_{jk}(\tau_i)}$ and $E_{tx,jk}^{\rho_j}(\tau_i) = T_{tx,jk}^{\rho_j}(\tau_i) Pt_j$, respectively, where $r_{jk}(\tau_i)$ is data-rate of the link between the two nodes, while Pt_j is the transmission power of jth node. Similarly, the reception time and energy at the jth node to receive the task of size $D_{\rho_j}^r$ from kth EN are $T_{rx,kj}^{\rho_j}(\tau_i) = \frac{D_{\rho_j}^r}{r_{kj}(\tau_i)}$ and $E_{rx,kj}^{\rho_j}(\tau_i) = T_{rx,kj}^{\rho_j}(\tau_i) Pr_j$, respectively, where Pr_j is the power spent for receiving data. A symmetric channel is considered between j and k.

In this work, for modeling the characteristics of a channel between the jth and the kth node at ith interval [111], we consider that the link gain can be modeled as $h_{j,k}(\tau_i) = \beta_0 \cdot d_{j,k}^{\theta^k}(\tau_i)$, where $d_{j,k}(\tau_i)$ is the distance between node j and k at ith interval, β_0 is the channel power gain at 1 m reference distance, while θ^k is the path loss coefficient for the the communication link between node j and k. The expression for the channel transmission rate is based on the Shannon capacity formula and can be written as:

$$r_{jk}(\tau_i) = b_l^{\rho_j}(\tau_i) \log_2 \left(1 + \frac{Pt_j \cdot h_{j,k}(\tau_i)}{N_0}\right) \quad \forall j, k$$

[6]In the following we identify with j and k the indexes of any generic node. Hence, j and k can have any index among v_m, r_n, u_p and H_h.

where Pt_j is the transmission power of a node j, $b_k^{\rho_j}(\tau_i)$ is the communication bandwidth, and $N_0 = N_T b_k^{\rho_j}(\tau_i)$ is the thermal noise power with noise power spectral density N_T.

Task Offloading Process

If mth VU is assigned to jth EN, then the time and energy required to offload the portion of task with offloading parameter α_{ρ_m} to the selected EN and to get back the result in the ith interval is given by:

$$T^{\text{off}}_{v_m,j}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \alpha_{\rho_m}(\tau_i^{\text{off}})\left(T^{\rho_m}_{tx,v_mj}(\tau_i^{\text{off}}) + T^{\rho_m}_{c,j}(\tau_i^{\text{off}}) + T^{\rho_m}_{rx,jv_m}(\tau_i^{\text{off}})\right) \tag{5.47a}$$

$$E^{\text{off}}_{v_m,j}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \alpha_{\rho_m}(\tau_i^{\text{off}})\left(E^{\rho_m}_{tx,v_mj}(\tau_i^{\text{off}}) + E^{\rho_m}_{rx,jv_m}(\tau_i^{\text{off}})\right) \tag{5.47b}$$

where, similarly to other approaches, e.g., [78, 116], the analysis has been simplified by limiting to the user-side energy consumption.

Local Computation The amount of time and energy required for the local computation of the remaining task in the ith interval is (from (5.46)),

$$T^{\text{loc}}_{v_m}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \left(1 - \alpha_{\rho_m}(\tau_i^{\text{off}})\right) T^{\rho_m}_{c,v_m} \tag{5.48a}$$

$$E^{\text{loc}}_{v_m}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \left(1 - \alpha_{\rho_m}(\tau_i^{\text{off}})\right) E^{\rho_m}_{c,v_m} \tag{5.48b}$$

Partial Computation Offloading From (5.47) and (5.48), the delay and the energy consumed during the task processing phases when partial offloading is performed (in the ith offloading interval) can be written as:

$$T^{\rho_m}_{v_m,j}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \max\left\{T^{\text{off}}_{v_m,j}(\alpha_{\rho_m}(\tau_i^{\text{off}})), T^{\text{loc}}_{v_m}(\alpha_{\rho_m}(\tau_i^{\text{off}}))\right\}$$

$$E^{\rho_m}_{v_m,j}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = E^{\text{off}}_{v_m,j}(\alpha_{\rho_m}(\tau_i^{\text{off}})) + E^{\text{loc}}_{v_m}(\alpha_{\rho_m}(\tau_i^{\text{off}}))$$

where the local and offloaded computing are supposed to be performed in parallel. Each VU should finish the offloading process and receive the results back within the sojourn time, hence:

$$T^{\text{off}}_{v_m,j}(\alpha_{\rho_m}(\tau_i^{\text{off}})) \leq T^{\text{soj}}_{v_m,j}(\tau_i^{\text{off}}) \quad \forall i \tag{5.49}$$

Service Placement Penalty In a multi-service VN, if a selected EN is unable to provide a requested service, VUs need to pay additional costs in terms of handover costs. Various mechanisms can be adapted in such situations. For example, the selected EN can transfer the VUs data to the nearby EN able to provide a requested service. In another case, the selected EN can request a centralized orchestrator for providing the requested service. Such operations can lead to additional penalties in the overall offloading process. Therefore, here we introduce a generic service placement penalty during the offloading process defined as

$$c(\tau_i^{\text{off}}) = \begin{cases} \zeta & \text{if } b(S_{\rho_m}, j, l, \tau_i^{\text{sp}}) \neq 1 \wedge a_{(v_m,j,l)}(\tau_i^{\text{ns}}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

where ζ is the constant penalty when selected EN is not able to provide a requested service.

5.4.1.3 Problem Formulation

The main aim of this work is to optimize the network-wide performance of the EC-enabled multiservice VN. We aim to optimize the performance in terms of overall latency, energy, and service placement during the offloading process. Our main

$$\mathbf{P1}: \min_{\mathcal{A},\mathbf{A},\mathcal{B}} \left\{ \frac{1}{T\cdot M} \sum_{\tau_{\bar{i}}^{\mathrm{sp}}=0}^{T-1} \sum_{\tau_{\hat{i}}^{\mathrm{ns}}=\tau_{\bar{i}}^{\mathrm{sp}}}^{\tau_{\bar{i}}^{\mathrm{sp}}+\Delta^{\mathrm{sp}}-\Delta^{\mathrm{ns}}} \sum_{\tau_{i}^{\mathrm{off}}=\tau_{\hat{i}}^{\mathrm{ns}}}^{\tau_{\hat{i}}^{\mathrm{ns}}+\Delta^{\mathrm{ns}}-\Delta^{\mathrm{off}}} \left(\sum_{l=1}^{L} \sum_{j=1}^{J(l)} \sum_{m=1}^{M} \left[\gamma_1 T_{v_m,j}^{\rho_m}\left(\alpha_{\rho_m}(\tau_i^{\mathrm{off}})\right) + \gamma_2 E_{v_m,j}^{\rho_m}\left(\alpha_{\rho_m}(\tau_i^{\mathrm{off}})\right) + c(\tau_i^{\mathrm{off}}) \right] \right) \right\} \quad (5.9)$$

objective is to optimize the overall networking cost through the proper selection of ENs for service placement and computation offloading operations simultaneously. Additionally, offloading the optimal amount of data toward them further reduces the overall latency and energy cost. For this, we formulate the joint latency, energy, and service placement cost minimization problem as in (5.9), subject to:

$$\mathbf{C1}: \text{Eq. } (5.43) \quad (5.51)$$

$$\mathbf{C2}: \text{Eq. } (5.44) \quad (5.52)$$

$$\mathbf{C3}: \text{Eqs. } (5.45a), (5.45b) \text{ and } (5.45c) \quad (5.53)$$

$$\mathbf{C4}: \text{Eq. } (5.49) \quad (5.54)$$

$$\mathbf{C5}: T_{v_m,j}^{\rho_m}\left(\alpha_{\rho_m}(\tau_i^{\mathrm{off}})\right) \le T_{\rho_m} \quad \forall \mathcal{V}, \forall i, j \quad (5.55)$$

$$\mathbf{C6}: E_{v_m,j}^{\mathrm{off}}(\alpha_{\rho_m}(\tau_i^{\mathrm{off}})) < E_{v_m}^{\mathrm{loc}}(\alpha_{\rho_m}) \quad (5.56)$$

$$\mathbf{C7}: 0 \le \gamma_1, \gamma_2 \le 1;\ \gamma_1 + \gamma_2 = 1 \quad (5.57)$$

where $\mathcal{A} = \{\alpha_{\rho_m}(\tau_i^{\mathrm{off}})\}^M$ is the computation offloading matrix, $\mathbf{A} = \{\mathbf{A}(M, J(l), l, (\tau_{\hat{i}}^{\mathrm{ns}}))\}$ is the set of VU-EN assignment matrix, $\mathcal{B} = \{b(S_s, j, (\tau_{\bar{i}}^{\mathrm{sp}}))\}$ is the service placement matrix, γ_1 and γ_2 are weight coefficients for balancing latency and energy consumption, and T is the whole time interval considered. **C1** puts a limit on the maximum number of services placed on each EN j. **C2** stands that each VU can select at most one EN for the computation offloading. **C3** provides the limits over the number of user requests, processing capacity and bandwidth resource blocks requested by VUs towards ENs. According to **C4**, for avoiding handover phenomena and related latency, each VU should complete the offloading process before it passes through the selected ENs coverage. **C5** puts a limit on the maximum processing time as one of the task requirements. In order to have a valid offloading process, according to **C6**, the weighted energy consumed on VU for processing a complete task should be lower than the total weighted energy require to compute a complete task locally. **C7** stands that the two weighting coefficients (γ_1, γ_2) should be between 0 and 1 with a sum equal to 1.

5.4.2 Multi-time Scale Optimization

In the considered scenario, the *service placement decisions* taken by the centralized operator can impact the users' network selection possibilities and corresponding outcomes in terms of accessing the services with reduced costs. On the other hand, the *network selection decisions* made by the VUs can further impact the *offloading decision* and, thus, corresponding task processing costs. Thus, these processes and corresponding decisions can form a hierarchy of decisions impacting each other's performances. In addition to this, these decisions should be made at different time scales (i.e., $\Delta^{\text{sp}}, \Delta^{\text{ns}}, \Delta^{\text{off}}$). This leads to a multi-time scale optimization process involving multiple layers of decisions impacting each other. The considered multi-time scale optimization problem can be solved effectively through sequential decision-making processes, e.g., MDP, where a multi-time scale MDP model [117] can be considered to solve the problem of joint service placements, network selection, and computation offloading effectively. In the following, we define a Multi-Time scale MDP (MDP-MT) as represented in Fig. 5.20 through several basic elements discussed in the following.

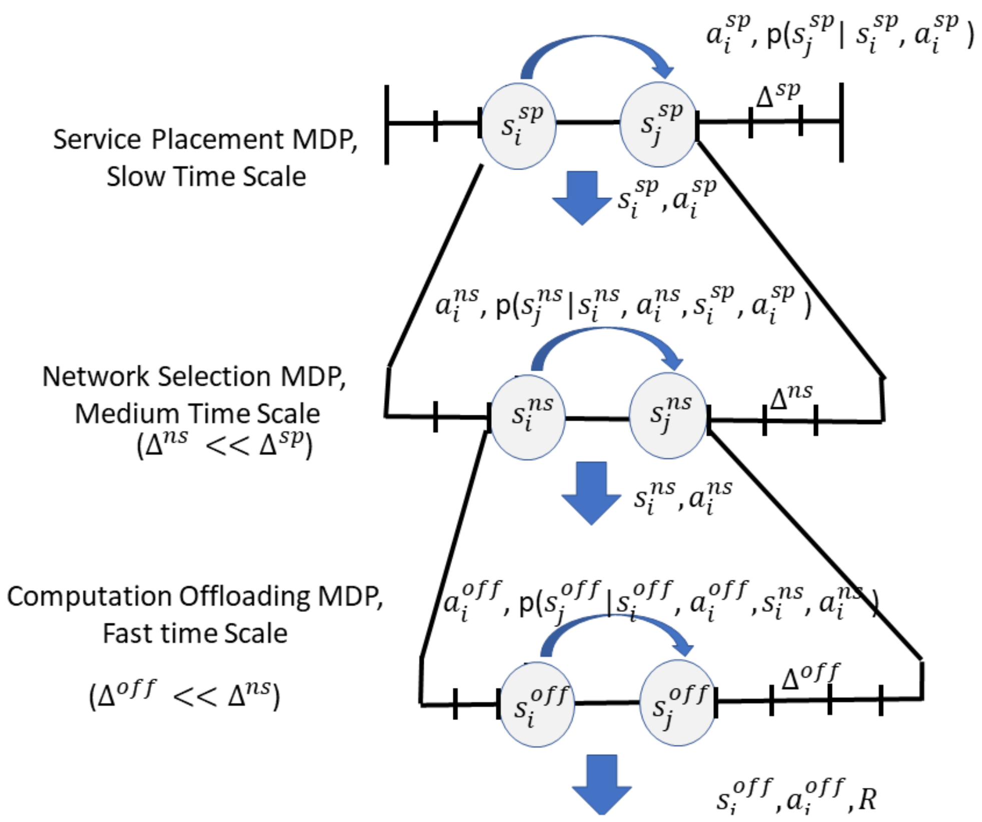

a_i^{sp}, p(s_j^{sp}| s_i^{sp}, a_i^{sp})
Δ^{sp}
Service Placement MDP, Slow Time Scale
s_i^{sp}
s_j^{sp}
s_i^{sp}, a_i^{sp}
a_i^{ns}, p(s_j^{ns}|s_i^{ns}, a_i^{ns}, s_i^{sp}, a_i^{sp})
Network Selection MDP, Medium Time Scale (Δ^{ns} << Δ^{sp})
s_i^{ns}
s_j^{ns}
Δ^{ns}
s_i^{ns}, a_i^{ns}
Computation Offloading MDP, Fast time Scale (Δ^{off} << Δ^{ns})
a_i^{off}, p(s_j^{off}|s_i^{off}, a_i^{off}, s_i^{ns}, a_i^{ns})
s_i^{off}
s_j^{off}
Δ^{off}
s_i^{off}, a_i^{off}, R

FIGURE 5.20: Multi-scale MDP Model for the Service Placement, Network Selection, and Offloading Problem.

5.4.2.1 Service Placement MDP

The service placement MDP corresponds to the MDP model for solving the network service placement problem over resource-constrained EN for satisfying the VUs' demands. The service placement decisions can be taken over a longer time scale Δ^{sp}.

The discrete state space for the service placement MDP is defined as $\mathcal{S}^{\text{sp}} = \{s_1^{\text{sp}}, \cdots, s_u^{\text{sp}}, \cdots, s_U^{\text{sp}}\}$ with maximum U states. $\mathcal{S}^{\text{sp}}$ is modeled as a function of the number of EN available and the service placement updates over time. Thus, $s_u^{\text{sp}}(\tau_i^{\text{sp}})$=$\{N(\tau_i^{\text{sp}}), P(\Delta_i^{\text{sp}}), H(\tau_i^{\text{sp}}), \hat{P}_R(\tau_i^{\text{sp}}), \hat{P_U}(\tau_i^{\text{sp}}), \hat{P}_H(\tau_i^{\text{sp}})\}$, where $\hat{P}_R(\tau_i^{\text{sp}})_{N(\tau_i^{\text{sp}})\times S}$, $\hat{P}_U(\tau_i^{\text{sp}})_{P(\tau_i^{\text{sp}})\times S}$, and $\hat{P}_H(\tau_i^{\text{sp}})_{H(\tau_i^{\text{sp}})\times S}$ are the binary matrices modeling the change in the service placement over different edge layers at τ_i^{sp}. For example, if the selected action places the sth service over nth RSU then $\hat{P}_R(\tau_i^{\text{sp}})(i, j) = 1$, else it takes value zero.

A discrete set of actions are defined through $\mathcal{A}^{\text{sp}} = \{a_1^{\text{sp}}, \cdots, a_{\bar{u}}^{\text{sp}}, \cdots, s_{\bar{U}}^{\text{sp}}\}$ with maximum $\bar{U}$ actions. $\mathcal{A}^{\text{sp}}$ includes all the feasible service placement options that can be employed by an orchestrator. For limiting the complexity of the MDP we have assumed that all the ENs from the same edge layer have the same subset of services placed on them.

The performance of service placement MDP is measured through the feedback signal generated as a sum of the total reward received during the offloading and network selection process as shown in Fig. 5.21, i.e., the hierarchical feedback process.

5.4.2.2 Network Selection MDP

The network selection MDP model corresponds to the network selection problem where decisions are made at a moderate time scale Δ^{ns} compared with the service placement problem. The network selection problem aims to find a proper EN that is able to provide the requested service. If the requested service is not available at the selected EN then the additional costs may be included in terms of service handovers. Thus, the decisions made by the network selection MDP will also be based upon the current state-action pairs of the service placement MDP model. In addition to this, several local environment parameters such as other competing VUs, available ENs, and their states, the requested service type, etc., can impact the network selection decision. Providing a one-fit-all model in such dynamic situations can reduce overall performance. With this in mind, here, we induce local environment parameters collected through the V2X technology into the MDP process by modeling it through different scenarios. In particular, vehicular scenarios set $\Omega = \{k_1, \cdots, k_g, \cdots, k_G\}$

based upon local vehicular density $\mathcal{D}$, the number of RSUs R_m, UAVs U_m and HAP nodes H_h covering the VU and the requested service type. Here G is the maximum number of considered scenarios. Thus the generic gth scenario, k_g, is defined as a tuple $k_g = \langle \mathcal{D}, R_m, U_m, R_m, S_{\rho_m} \rangle$ with,

$$\mathcal{D} = \begin{cases} 0 & \text{if } 1 \leq M < M_1 \\ 1 & \text{if } M_1 \leq M < M_2 \\ 2 & \text{if } M_2 \leq M \end{cases}$$

where M_1 and M_2 are parameters introduced for classifying the VUs traffic scenarios into low, medium, and high density.

Next, we define the state-space for the network selection MDP as $\mathcal{S}^{\text{ns}} = \{s_m^{\text{ns}}(\tau_i^{\text{ns}})\}$ with $s_m^{\text{ns}}(\tau_i^{\text{ns}}) \in \{(s_{v_m}^{\text{ns}}, s_e^{\text{ns}})\}$. The individual state $s_m^{\text{ns}}(\tau_i^{\text{ns}})$ is based upon the VU side state $(s_{v_m}^{\text{ns}})$, and the selected ENs state s_e^{ns} where e can be a RSU, UAV or HAP node. The VUs state $s_m^{\text{ns}}(\tau_i^{\text{ns}})$ is modeled through the requested service S_{ρ_m}, $d_{m,e}$, the distance between VU and the EN, and $D_{v_m,e}$, the distance before VU passes through the coverage area of e.

The action space for VUs in scenario g is defined as $\mathcal{A}_{\kappa_g}^{\text{sp}} = \{a_{\bar{m},j}^{\text{ns}}(\tau_i^{\text{ns}})\}$ for the network selection MDP corresponds to all possible sets of actions with individual actions, $a_{\bar{m},g}^{\text{ns}}(\tau_i^{\text{ns}}) = [\{0,1\}_{1\times R_m}, \{0,1\}_{1\times U_m}, \{0,1\}_{1\times H_m}]$, with $\sum a_{\bar{m},g}^{\text{ns}}(\tau_i^{\text{ns}}) = 1$

The performance of network selection MDP is measured through the feedback signal generated as a sum of rewards received during the offloading process, as shown in Fig. 5.21.

5.4.2.3 Computation Offloading MDP

The computation offloading problem aims to find a proper amount of data to be offloaded toward a selected EN. With the high dynamicity and the frequent task requests, such decisions should be performed over a fast time scale Δ^{off}. The computation offloading process should be concluded before VU passes through the coverage range of a selected EN. In addition, the complete task processing operation should be performed within a given task latency requirement, and a proper amount of data should be offloaded for minimizing the energy costs of VUs' local data computation and data transmission operations. The performance of offloading MDP can be impacted by the decisions of network selection and service placement MDPs. The incorrect EN selection during the network selection process or the imperfect alignments of services over different ENs can lead to limited performance during the offloading phases.

By taking into account the various performance requirements of MDP, here we introduce a discrete state space for the offloading MDP problem that is based upon following three binary functions that model the behavior of offloading MDP over time. If the mth VU is assigned to the nth EN and performs offloading operation with offloading parameter α_{ρ_m}, the environment can be modeled through three proper binary functions, as:

$$F^{1}_{\rho_m,n}(\tau_i^{\text{off}}) = \begin{cases} 0 & T^{\text{off}}_{m,n}(\alpha_{\rho_m}(\tau_i^{\text{off}})) \leq T^{\text{soj}}_{m,n}(\tau_i^{\text{off}}) \\ 1 & \text{else} \end{cases}$$

$$F^{2}_{\rho_m,n}(\tau_i^{\text{off}}) = \begin{cases} 0 & T^{\rho_m}_{m,n}\left(\alpha_{\rho_m}(\tau_i^{\text{off}})\right) \leq T_{\rho_m} \\ 1 & \text{else} \end{cases}$$

$$F^3_{\rho_m,n}(\tau_i^{\text{off}}) = \begin{cases} 0 & E^{\text{off}}_{m,n}(\alpha_{\rho_m}(\tau_i^{\text{off}})) < w_1 E^{\rho_m}_{c,m} \\ 1 & \text{else} \end{cases}$$

where $F^1_{\rho_m,n}(\tau_i^{\text{off}})$, $F^2_{\rho_m,n}(\tau_i^{\text{off}})$ and $F^3_{\rho_m,n}(\tau_i^{\text{off}})$ are the binary functions depending upon the sojourn time constraint (5.49), application latency requirement (5.55) and the energy constraint (5.56), respectively.

The discrete state space for the computation offloading MDP is defined as $\mathcal{S}^{\text{off}} = \left\{s_1^{\text{off}}, \cdots, s_l^{\text{off}}, \cdots, s_L^{\text{off}}\right\}$ with maximum L states with the individual state defined as $s_l^{\text{off}}(\tau_i^{\text{off}}) = \left\{F^2_{\rho_m,n}(\tau_i^{\text{off}}), F^2_{\rho_m,n}(\tau_i^{\text{off}}), F^3_{\rho_m,n}(\tau_i^{\text{off}})\right\}$. The action space $\mathcal{A}^{\text{off}} = \{a_{\bar{l}}^{\text{off}}\}$ for the computation offloading MDP is defined as $\mathcal{A}^{\text{off}} = [0, \Lambda, 2\Lambda, \cdots, 1]$, where Λ is the step change in the value of offloading parameter α_{ρ_m}.

5.4.2.4 Reward Function

The performance of the MDP can be described through a joint reward function defined as,

$$r^{\text{off}}(s_l^{\text{off}}) = \gamma_1 T^{\rho_m}_{v_m,j}\left(\alpha_{\rho_m}(\tau_i^{\text{off}})\right) + \gamma_2 E^{\rho_m}_{v_m,j}(\alpha_{\rho_m}(\tau_i^{\text{off}})) + c(\tau_i^{\text{off}}) + F^1_{\rho_m,n}(\tau_i^{\text{off}}) + F^2_{\rho_m,n}(\tau_i^{\text{off}}) + F^3_{\rho_m,n}(\tau_i^{\text{off}}) \quad (5.58)$$

The total reward received is the sum of latency, energy costs, service placement, and additional constraint failure penalties.

$$R^{\mathrm{sp}}(s_u^{\mathrm{sp}}(\Delta_k^{\mathrm{sp}}), a_{\bar{u}}^{\mathrm{sp}}(\Delta_k^{\mathrm{sp}}), s_m^{\mathrm{ns}}, \pi_m^{\mathrm{ns}}, s_l^{\mathrm{off}}, \pi_l^{\mathrm{off}}) = \mathbb{E}_{\substack{s_0^{\mathrm{ns}}, s_0^{\mathrm{off}} \\ s_u^{\mathrm{sp}}, a_{\bar{u}}^{\mathrm{sp}}}} \Bigg\{ \sum_{\tau_i^{\mathrm{ns}}=\tau_k^{\mathrm{sp}}}^{(\tau_k^{\mathrm{sp}}+\Delta^{\mathrm{sp}}-\Delta^{\mathrm{ns}})} \alpha_1^{\sigma_1(\tau_i^{\mathrm{ns}})} \sum_{\tau_j^{\mathrm{off}}=\tau_i^{\mathrm{ns}}}^{(\tau_i^{\mathrm{ns}}+\Delta^{\mathrm{ns}}-\Delta^{\mathrm{off}})} \alpha_2^{\sigma_2(\tau_j^{\mathrm{off}})}$$

$$R^{\mathrm{off}}\Bigg(s_m^{\mathrm{ns}}(\tau_i^{\mathrm{ns}}), \pi^{\mathrm{ns}}\Big(s_m^{\mathrm{ns}}(\tau_i^{\mathrm{ns}}), s_u^{\mathrm{sp}}(\tau_k^{\mathrm{sp}}), a_{\bar{u}}^{\mathrm{sp}}(\tau_k^{\mathrm{sp}})\Big), s_l^{\mathrm{off}}(\tau_j^{\mathrm{off}}),$$

$$\pi^{\mathrm{off}}\Big(s_l^{\mathrm{off}}(\tau_j^{\mathrm{off}}), s_m^{\mathrm{ns}}(\tau_i^{\mathrm{ns}}), \pi^{\mathrm{ns}}\Big(s_m^{\mathrm{ns}}(\tau_i^{\mathrm{ns}}), s_u^{\mathrm{sp}}(\tau_k^{\mathrm{sp}}), a_{\bar{u}}^{\mathrm{sp}}(\tau_k^{\mathrm{sp}})\Big), s_u^{\mathrm{sp}}(\tau_k^{\mathrm{sp}}), a_{\bar{u}}^{\mathrm{sp}}(\tau_k^{\mathrm{sp}})\Big)\Bigg)\Bigg\} \tag{5.18}$$

5.4.3 Deep Q-Learning for Service Placement, Network Selection, and Computation Offloading

In the previous section, the different elements of the MDP models were presented. By solving the proposed MDP models, VUs can find a proper service placement, node selection, and offloading amount able to minimize the overall latency, energy, and service placement costs.

For any time instant τ_i the state space $\mathcal{ST}$ is equal to $\{s(\tau_i)\}$, where $s(\tau_i) = (s_l^{\mathrm{off}}, s_m^{\mathrm{ns}}, s_u^{\mathrm{sp}})$ is the instantaneous state of a multi-time scale MDP, i.e., a combination of all three MDPs states. For the large time interval Δ^{sp}, s_u^{sp} remains unchanged while multiple transactions can occur for s_l^{off} and s_m^{ns}. The solutions can be defined as a policy function $\pi \in \Pi$:

$$\pi = \Big\{\pi^{\mathrm{off}}(s_l^{\mathrm{off}}(\tau_i+\delta)), \pi^{\mathrm{ns}}(s_m^{\mathrm{ns}}(\tau_i+\delta)), \pi^{\mathrm{sp}}(s_u^{\mathrm{sp}}(\tau_i+\delta))\Big\}$$

that maps every state $s \in \mathcal{ST}$ to action $a = \{(a_{\bar{l}}^{\mathrm{off}}, a_{\bar{m}}^{\mathrm{ns}}, a_{\bar{u}}^{\mathrm{sp}})\} \in \mathcal{AS}$. Given the offloading policy π^{off}, network selection policy π^{ns}, and the lower level reward R^{off}, over a large time scale Δ^{sp}, we define a $\Delta^{\mathrm{sp}}/\Delta^{\mathrm{off}}$-horizon total expected reward as (5.18), where

$$V(s_l^{\text{off}}, s_m^{\text{ns}}, s_u^{\text{sp}})^{\pi^*} = \min_{a_u^{\text{sp}} \in \mathcal{A}^{\text{sp}}} \left\{ \min_{\pi_m^{\text{ns}} \in \Pi^{\text{ns}}} \left\{ \min_{\pi_l^{\text{off}} \in \Pi^{\text{off}}} \left\{ R^{\text{sp}} \left(s_u^{\text{sp}}, a_u^{\text{sp}}, s_m^{\text{ns}}, \pi_m^{\text{ns}}, s_l^{\text{off}}, \pi_l^{\text{off}} \right) \right. \right. \right.$$
$$\left. \left. \left. + \gamma \sum_{\forall s_{\hat{l}}^{\text{off}}} \sum_{\forall s_{\hat{m}}^{\text{ns}}} \sum_{\forall s_{\hat{u}}^{\text{sp}}} P^{\text{off}}_{(s_l^{\text{off}}, s_{\hat{l}}^{\text{off}})} (\pi_l^{\text{off}}(s_l^{\text{off}})) P^{\text{ns}}_{(s_m^{\text{ns}}, s_{\hat{m}}^{\text{ns}})} (\pi_m^{\text{ns}}) P^{\text{sp}}_{(s_u^{\text{sp}}, s_{\hat{u}}^{\text{sp}})} (a_u^{\text{sp}}) V^*(s_{\hat{l}}^{\text{off}}, s_{\hat{m}}^{\text{ns}}, s_{\hat{u}}^{\text{sp}}) \right\} \right\} \right\} \tag{5.19}$$

$$\sigma_1(n\Delta^{\text{sp}}/\Delta^{\text{ns}} + r) = r, \quad \forall n > 0, r = 0, \cdots, \Delta^{\text{sp}}/\Delta^{\text{ns}},$$
$$\sigma_2(n\Delta^{\text{ns}}/\Delta^{\text{off}} + r) = r, \quad \forall n > 0, r = 0, \cdots, \Delta^{\text{ns}}/\Delta^{\text{off}},$$
$$0 < \alpha_1, \alpha_2 \leq 1,$$

and s_0^{ns} and s_0^{off} are the initial state values for Δ_k^{sp} and Δ_i^{ns}, respectively. Here, the total expected reward achieved by the offloading and network selection level MDPs will act as a single-step reward for a service placement MDP.

Selecting different actions can result in different policy functions, where the aim is to find an optimal policy that corresponds to the minimum delay and energy cost during vehicular task processing. For every policy π, a value function $V_\pi(s(\tau_i))$, corresponding to a state $s(\tau_i)$ can be defined for analyzing its performance. In general, $V_\pi(s(\tau_i))$ corresponds to an expected value of a discounted sum of total reward received by following the policy π from state $s(\tau_i)$, and can be defined as (5.18). The optimal policy π^* corresponding to the value function V is defined through Bellman equation as in (5.19).

Here, $P^{\text{off}}_{(s_l^{\text{off}}, s_{\hat{l}}^{\text{off}})}(\pi_l^{\text{off}}(s_l^{\text{off}}))$, $P^{\text{ns}}_{(s_m^{\text{ns}}, s_{\hat{m}}^{\text{ns}})}(\pi_m^{\text{ns}}(s_m^{\text{ns}}))$, and $P^{\text{sp}}_{(s_u^{\text{sp}}, s_{\hat{u}}^{\text{sp}})}(a_u^{\text{sp}})$ model the environment dynamics based upon the state transition probabilities. Also, R^{off} is the mean value of the immediate reward r^{off} defined in (5.58).

Given the complex and dynamic nature of the considered vehicular scenario it is hard to define the environmental dynamics. With this in mind, we consider a model-free RL for solving the proposed multi-time scale MDP model to find the optimal policies. Among others, Q learning is one of the highly explored model-free strategies for determining the optimal π^* in unknown environments. The Q-learning strategy is based upon a state-action function, i.e., Q-function, defined as,

$$Q^{\pi}(s', a') = R(s', a') + \gamma \sum_{\hat{s} \in S} P_{s'\hat{s}}(a') V^{\pi}(\hat{s})$$

representing a discounted cumulative reward from state s' when action a' is taken before following the policy π. The optimal Q value can be represented as

$$Q^{\pi^*}(s', a') = R(s', a') + \gamma \sum_{\hat{s} \in S} P_{s'\hat{s}}(a') V^{\pi^*}(\hat{s})$$

where $V^{\pi^*}(\hat{s}) = \min_{\hat{a} \in A} Q^{\pi^*}(s', \hat{a})$. The Q values can be estimated through a recursive approach where,

$$Q_{t+1}(s', a') = Q_t(s', a') + \epsilon \cdot \left(r + \gamma \max_{\hat{a}} Q_t(s', \hat{a}) - Q_t(s', a') \right)$$

where ϵ is a learning rate. The Q-function can be estimated through a neural network-based function approximation technique with $Q(s', a'; \theta) \approx Q(s', a')$, where θ represents the weights of the neural network. Through the training process, the values of θ can be adjusted to reduce the mean square error values.

In the Deep Q Network (DQN) based approach two networks (i.e., primary and target Q networks) are considered for a reliable estimation of Q functions over different time scales. The primary network estimates the real/primary Q-value while the target Q-values are estimated through the target network. The RL agent uses the backpropagation and gradient descent processes with mean square error (MSE)-based loss function for reducing the gap between the primary and the target Q-values where the loss function is defined as:

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{\hat{a}} Q_t(s', \hat{a}, \theta^{'}) - Q(x, a, \theta)\right)^2\right] \tag{5.61}$$

where the primary values $Q(x, a, \theta)$ are based upon primary network parameters θ, and $r + \gamma \max_{\hat{a}} Q_t(s', \hat{a}, \theta^{'})$ is the target Q value based upon the target network parameters $\theta^{'}$.

Fig. 5.21 describes the proposed Deep Q-learning method for solving the multi-time scale MDP model to find the optimal service placements, network selection, and offloading policies. Three DQN architectures are considered for solving the problem over different time scales. For the service placement case, DQN architectures are composed of a primary network and a target network with $\bar{K}^{\text{sp}}$ layers, each with $\bar{n}^{\text{sp}}_{\bar{k}}$ neurons, where $\bar{k} = 1, \dots, \bar{K}^{\text{sp}}$. Similarly, for each scenario k_g, network selection and computation offloading DQN with $\bar{K}^{\text{ns}}_g$ and $\bar{K}^{\text{off}}_g$ layers are considered, with neurons $\bar{n}^{\text{ns}}_{g,\bar{k}}$ and $\bar{n}^{\text{off}}_{g,\bar{k}}$, where $\bar{k} = 1, \dots, \bar{K}^{\text{ns}}_g$ and $\bar{k} = 1, \dots, \bar{K}^{\text{off}}_g$. Moreover, reply memories having size $\mathcal{D}^{\text{sp}}$, $\mathcal{D}^{\text{ns}}_g$, and $\mathcal{D}^{\text{off}}_g$ are considered for storing the agents' past experiences.

In the beginning, the primary network associated with the service placement problem senses the vehicular environment for processing the current state. Service placement decisions are updated over a large time scale τ^{sp}_i through a proper action based upon the Epsilon Greedy Policy (EGP) with parameter e^{sp}. The state action pair $(s^{\text{sp}}_u(\tau^{\text{sp}}_i), a^{\text{sp}}_{\bar{u}}(\tau^{\text{sp}}_i))$ is then forwarded to the other two DQNs. The primary network of a second DQN senses the current state of the environment and updates

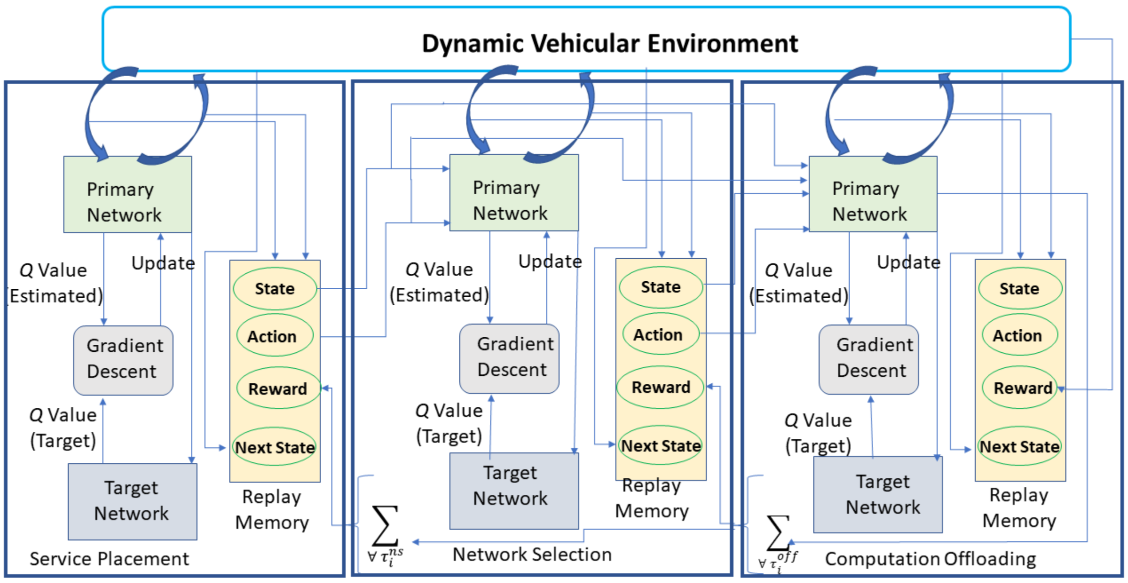


FIGURE 5.21: Proposed Deep Q-Learning Solution

the network selection decisions through EGP with parameter e_g^{ns} over each Δ^{ns} interval. These state action pairs $(s_u^{\text{ns}}(\tau_i^{\text{ns}}), a_{\bar{u}}^{\text{ns}}(\tau_i^{\text{ns}}))$ are then forwarded towards the offloading DQN. The offloading decisions are based upon the current environment dynamics impacted by $a_{\bar{u}}^{\text{sp}}(\tau_i^{\text{sp}})$, $a_{\bar{u}}^{\text{ns}}(\tau_i^{\text{ns}})$ and the EGP strategies with parameter $e^{\text{off},g}$ over Δ^{off} intervals. It also receives the instantaneous reward which along with the other entities, i.e, current state, action, and next state are stored in the reply buffer. With multiple offloading decisions, the cumulative reward of the offloading process is backpropagated to the network selection and service placement DQNs.

Algorithm 10 details the DQN process for the multi-time scale service placement, network selection, and computation offloading problem. The process begins with the definition and the initialization of primary and target networks for considered tasks (Line 1-2). After that training process iterates over $\bar{N}$ training iterations where in each iteration DQN models are updated (Line 3-33). In each training iteration, an initially random set of states are selected (Line 5). Then over multiple epochs, i.e., up to I_{SP} the model is trained through a gradient descent approach by moving the weight values appropriated to minimize the loss function value as a function of target and primary Q values. In each training epoch action $\boldsymbol{a}^{\mathrm{ns}}$ is selected through EGP with probability $\boldsymbol{e}$ and applied over the service placement network for generating the next state (Line 13-14). Since the reward is based upon the performance of moderate and small time scale MDPs, next the network selection and offloading process MDP models correspond to the different scenarios are training in a similar, manner over $\boldsymbol{I}_{\mathrm{ns}}$ and $\boldsymbol{I}_{\mathrm{off}}$ epoch respectively.

Algorithm 10 Multi-time Scale Deep Q-Learning

```
Input: $\Delta^{\mathrm{sp}}, \Delta^{\mathrm{ns}}, \Delta^{\mathrm{off}}, \mathcal{S}^{\mathrm{sp}}, \mathcal{S}^{\mathrm{ns}}, \mathcal{S}^{\mathrm{off}}, \mathcal{A}^{\mathrm{sp}}, \mathcal{A}^{\mathrm{ns}}, \mathcal{A}^{\mathrm{off}}, \mathcal{S}, \mathcal{V}, \mathcal{R}, \mathcal{U}, h, \bar{N},$
        $\mathcal{I}_{\mathrm{sp}}, \mathcal{I}_{\mathrm{ns}}, \mathcal{I}_{\mathrm{off}}, \bar{i}, G, e^{\mathrm{sp}}, e_g^{\mathrm{ns}}, e_g^{\mathrm{off}}, \mathcal{D}^{\mathrm{sp}}, \mathcal{D}_g^{\mathrm{ns}}, \mathcal{D}_g^{\mathrm{off}},$
        $\epsilon^{\mathrm{sp}}, \epsilon_g^{\mathrm{ns}}, \epsilon_g^{\mathrm{off}}, \epsilon^{\mathrm{sp}}, \gamma_g^{\mathrm{ns}}, \epsilon_g^{\mathrm{off}}$
Output: $w^{p,\mathrm{sp}}$, $\{w_g^{p,\mathrm{ns}}, w_g^{p,\mathrm{off}}, \forall g \in K\}$
 1: Initialize $w^{p,\mathrm{sp}}$, $\{w_g^{p,\mathrm{ns}}, \forall g \in K\}$, $\{w_g^{p,\mathrm{off}}, \forall g \in K\}$
 2: Duplicate policy networks to Target Networks,
        i.e., $w^{T,\mathrm{sp}} = w^{p,\mathrm{sp}}, \{w_g^{T,\mathrm{ns}} = w^{p,\mathrm{ns}}\}, \{w_g^{T,\mathrm{off}} = w^{p,\mathrm{off}}\}$
 3: for all $ep = 1, \dots, \bar{N}$ do
 4:     Select Random $s_0^{\mathrm{sp}}$, $\{s_{0,g}^{\mathrm{ns}}\}$, $\{s_{0,g}^{\mathrm{off}}\}$
 5:     $s^{\mathrm{sp}} \leftarrow s_0^{\mathrm{sp}}, \{s_g^{\mathrm{ns}} \leftarrow s_{0,g}^{\mathrm{ns}}, \quad s_g^{\mathrm{off}} \leftarrow s_{0,g}^{\mathrm{off}}, \forall g\}, \quad it = 0$
 6:     while $i_{\mathrm{sp}} \neq \mathcal{I}_{\mathrm{sp}}$ do
 7:         $i_{\mathrm{sp}} = i_{\mathrm{sp}} + 1$
 8:         Select action $a^{\mathrm{sp}} \in \mathcal{A}^{\mathrm{sp}}$ with probability $e^{\mathrm{sp}}$
 9:         Determine next state $(s_{\mathrm{new}}^{\mathrm{sp}})$
10:         for all $g = 1, \dots, G$ do
11:             while $i_{\mathrm{ns}} \neq I_{\mathrm{ns}}$ do
12:                 $i_{\mathrm{ns}} = i_{\mathrm{ns}} + 1$
13:                 Select action $a^{\mathrm{ns}} \in \mathcal{A}^{\mathrm{ns}}$ with probability $e_g^{\mathrm{ns}}$
14:                 Determine next state $(s_{\mathrm{new}}^{\mathrm{ns}})$
15:                 while $i_{\mathrm{off}} \neq I_{\mathrm{off}}$ do
16:                     Select $a^{\mathrm{off}} \in \mathcal{A}^{\mathrm{off}}$ with probability $e_g^{\mathrm{off}}$
17:                     Find next state $s_{\mathrm{new}}^{\mathrm{off}}$ and reward $R^{\mathrm{off}}$
18:                     Store $\mathcal{D}_g^{\mathrm{off}} \leftarrow (s^{\mathrm{off}}, a^{\mathrm{off}}, R^{\mathrm{off}}, s_{\mathrm{new}}^{\mathrm{off}})$
19:                     $w_g^{p,\mathrm{off}}, w_g^{T,\mathrm{off}} =$
                            $\mathrm{DQN}(\mathcal{D}_g^{\mathrm{off}}, k, w_g^{p,\mathrm{off}}, w_g^{T,\mathrm{off}}, i_{\mathrm{off}}, \bar{i}, \epsilon_g^{\mathrm{off}}, \gamma_g^{\mathrm{off}})$
20:                     $s^{\mathrm{off}} \leftarrow s^{\mathrm{off,new}}$
21:                 end while
22:                 Use $w_g^{p,\mathrm{off}}$ to generate feedback,
                     i.e, Reward Signal $R^{\mathrm{ns}} = \sum_{\tau_j^{\mathrm{off}}=\tau_i^{\mathrm{ns}}}^{\tau_i^{\mathrm{ns}}+\Delta^{\mathrm{ns}}-\Delta^{\mathrm{off}}} R^{\mathrm{off}}(\tau_j^{\mathrm{off}})$
23:                 Store $\mathcal{D}_g^{\mathrm{ns}} \leftarrow (s^{\mathrm{ns}}, a^{\mathrm{ns}}, R^{\mathrm{ns}}, s_{\mathrm{new}}^{\mathrm{ns}})$
24:                 $w_g^{p,\mathrm{ns}}, w_g^{T,\mathrm{ns}} =$
                            $\mathrm{DQN}(\mathcal{D}_g^{\mathrm{ns}}, k, w_g^{p,\mathrm{ns}}, w_g^{T,\mathrm{ns}}, i_{\mathrm{ns}}, \bar{i}, \epsilon_g^{\mathrm{ns}}, \gamma_g^{\mathrm{ns}})$
25:                 $s^{\mathrm{ns}} \leftarrow s^{\mathrm{ns,new}}$
26:             end while
27:         end for
28:         Use $w_g^{p,\mathrm{ns}}$ to generate feedback,
```

DQN Function defined in Algorithm 11 embodies the core training process adapted by the DQN networks. The process involves the selection of a random batch of k samples from the memory buffer $\mathcal{D}$ (Line 2), defining the loss function value based upon the performance of the network (Line 3-4) with discount factors $(\gamma^{\text{sp}}, \gamma_g^{\text{ns}}, \gamma_g^{\text{off}})$, learning rates $(\epsilon^{\text{sp}}, \epsilon_g^{\text{ns}}, \epsilon_g^{\text{off}})$, and the gradient descent updates (Line 5) and target network update state if the appropriate number of epochs, i.e., $\bar{i}$, are performed.

Algorithm 11 DQN Function

```
Input: $\mathcal{D}, k, w^p, w^T, i, \bar{i}, \epsilon, \gamma$
Output: $\{w^p, w^T\}$

1: function DQN($\mathcal{D}, k, w^p, w^T, i, \bar{i}, \epsilon, \gamma$)
2:     Select Random batch of of $k$ samples from $\mathcal{D}$
3:     Preprocess and pass the batch to $w^p$
4:     Find Loss between primary and Target Q values using (5.61)
5:     With gradient descent step update $w^p$
6:     Update $w^T$ if $rem(i, \bar{i}) = 0$
7: end function
8: return $\{w^p, w^T\}$
```

The computation complexity for the basic DQN architecture can be defined as $\mathcal{O}(\mathcal{S} \cdot \mathcal{A} \cdot \mathcal{I})$ with $\mathcal{S}$, $\mathcal{A}$, being dimensions of state and action spaces and $\mathcal{I}$ being training epoch performed per iteration [69]. Therefore, for a considered multi-time scale approach, the computation complexity is given by $\mathcal{O}(\mathcal{S}_g \cdot \mathcal{A}_g \cdot \mathcal{I}_g)$ with $\mathcal{S}_g = \mathcal{S}^{\text{sp}} \bigcup \mathcal{S}_g^{\text{ns}} \bigcup \mathcal{S}_g^{\text{off}}$ as a union of state spaces for the particular gth scenario. Similarly, $\mathcal{A}_g = \mathcal{A}^{\text{sp}} \bigcup \mathcal{A}_g^{\text{ns}} \bigcup \mathcal{A}_g^{\text{off}}$ is the dimension of action space and $\mathcal{I}_g = I_{\text{sp}} + I_{\text{ns}} + I_{\text{off}}$. It should be noted that with the definition of multiple vehicular scenarios, the overall state space for the network selection and offloading processes can be reduced significantly, and with that, through the paralization approach the complexity can be limited.

5.4.4 Numerical Results

The proposed DQN-based multi-time scale MDP methods are simulated over a Python-based simulator for analyzing the performance. The following benchmark solutions are also considered for comparison purposes:

a) **Static Approach (SA):** In this case, the services are randomly placed and their placement is fixed over time. The network selection operation is based on a minimum distance approach, where VUs select the nearest EN to offload their complete task.

b) **MDP (Network Selection and Offloading) with Static Service Placements (MDP-SSP):** In this approach, the service placement is performed randomly and fixed over time. On the other hand, the network selection and offloading decisions are made through MDP with different time scales. This approach allows us to measure the impact of dynamic service placements over time.

c) **MDP (Service Placement and Offloading) with Random Network Selection (MDP-RS):** In this approach, each VU randomly selects the ENs while service placement and offloading decisions are made through the MDP approach with multiple time scales. This allows us to evaluate the performance of network selection operations performed by VUs over time.

d) **MDP (Service Placement and Network Selection) with Full Offloading (MDP-FO):** In this approach, service placement and network selection operations are performed through the MDP model with different time scales. Each VU performs the full offloading towards the selected EN.

TABLE 5.3: Simulation parameters

HAP Coverage (R_h)	2 km
UAV Coverage ($R_{u,p}$)	100 m
RSU Coverage (($R_{r,n}$))	50 m
VU Computation Cap. ($c_{v,m} \cdot f_{v,m}$)	2 GFLOPS
RSU Computation Cap. ($\mathcal{L}_{r,n} \cdot c_{r,n} \cdot f_{r,n}$)	10 GFLOPS
UAV Computation Cap. ($\mathcal{L}_{u,p} \cdot c_{u,p} \cdot f_{u,p}$)	10 GFLOPS
HAP Computation Cap. ($\mathcal{L}_h \cdot c_h \cdot f_h$)	30 GFLOPS
HAP Altitude ($\bar{h}_h$)	10 km
UAV Altitude ($\bar{h}_{u,p}$)	1 km
HAP Bandwidth (B_h)	250 MHz
UAV Bandwidth ($B_{u,p}$)	75 MHz
RSU Bandwidth ($B_{r,n}$)	25 MHz
VU Speed Range ($\vec{v}_{\min}, \vec{v}_{\max}$).	(8 m/s, 14 m/s)
VU Power ($P_{c,v_m}, Pt_{v_m}, Pr_{v_m}$)	(1.1, 1.5, 1.3) W

Simulation is performed considering an IoV scenario with varying numbers of VUs between 200 and 2000 and three edge layers (i.e., RSUs, UAVs, and HAP) are considered. We have considered $N = 50$ RSUs, $P = 30$ UAVs, and one HAP node for serving VUs with $\bar{S} = 5$ different services. Also, $\hat{\mathcal{S}}_n^1 = 3, \forall n$, $\hat{\mathcal{S}}_p^2 = 2, \forall p$, and $\hat{\mathcal{S}}_h^3 = 5$ stand for the bound on the number of services accommodated by different edge facilities. The generic mth VU generates a task request ρ_m with probability $p_a = 0.1$, having parameters $D_{\rho_m} = 5\,\text{MB}$, $D_{\rho_m}^r = 1\,\text{MB}$, $\Omega_{\rho_m} = 10^3 \cdot D_{\rho_m}$, and $T_{\rho_m} = 2\,\text{s}$. The service demand is based upon the Zipf distribution with parameter $\beta = 0.8$. Also, VUs speed is defined as in (5.42), where $\mu = 10$ and $\sigma = 3$. Also, $\zeta = 0.1$, $\gamma_1 = 0.5$, and $\gamma_2 = 0.5$ is considered during the problem formulation. The vehicular density parameters, $M_1 = 500$ and $M_2 = 1200$, are considered in the scenario definition. For DQN simulation primary and target networks with layers $\bar{K}^{\text{sp}} = 5$, $\bar{K}_g^{\text{ns}} = \bar{K}_g^{\text{off}} = 3$, are considered with learning parameters $e^{\text{sp}} = 0.7$, $e_g^{\text{ns}} = e_g^{\text{off}} = 0.65$, $\mathcal{D}^{\text{sp}} = 4000$, $\mathcal{D}_g^{\text{ns}} = \mathcal{D}_g^{\text{off}} = 2000$, $\epsilon^{\text{sp}} = \epsilon_g^{\text{ns}} = \epsilon_g^{\text{off}} = 0.05$, $\gamma^{\text{sp}} = \gamma_g^{\text{ns}} = \epsilon_g^{\text{off}} = 0.98$. Also the learning process includes $\bar{N} = 50$ with $I_{\text{sp}} = I_{\text{ns}} = I_{\text{off}} = 10^3$ and $\bar{i} = 50$. The other important system model parameters are provided in Table 5.3.

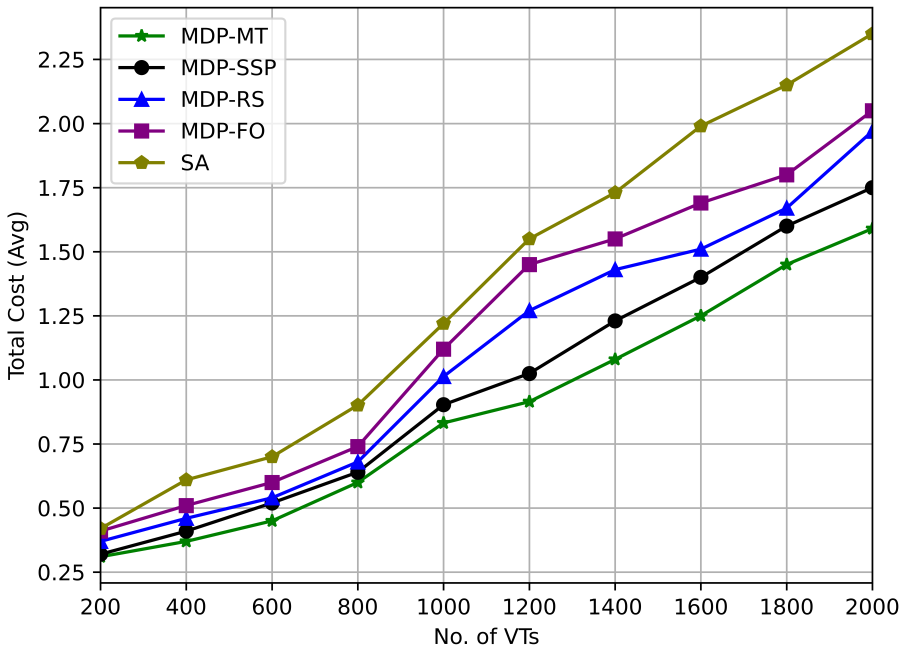


FIGURE 5.22: Performance results in terms of overall cost function with variable number of active vehicles.

1) Joint Cost Analysis with Varying Vehicular Density In this work, we aim to minimize the joint cost function of latency, energy, and the additional service placement penalty values. In Fig. 5.22, we present the average cost values for different solution approaches with varying numbers of VUs. The overall cost required for the SA approach is significantly high compared to the other approaches since all three decisions are performed through a heuristic approach. On the other hand, the other three MDP approaches (MDP-FO/RS/SSP) are optimizing the decisions for two MDPs while using the static approach for the remaining one. Such methods can reduce the overall cost requirements, however, resulting in suboptimal solutions. This highlights the importance of performing multi-time scale optimization for service placements, network selection, and offloading processes together. Indeed, the cost required for the proposed MDP-MT approach improves the overall performance.

2) Number of Handover Required Apart from the overall cost reduction, the reliability of the solutions can be an important criterion to analyze the performance.

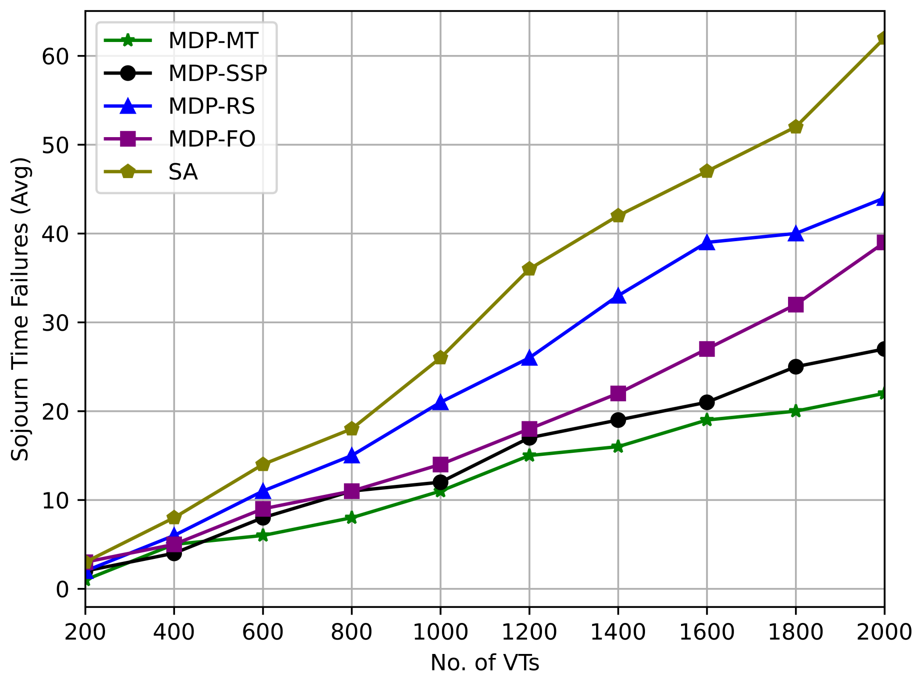


FIGURE 5.23: Percentage of VUs with Handover Requirements.

With this, in Fig. 5.23, we present the overall handover requirements in terms of sojourn time constraint failure during the offloading process. The SA approach with static service placement, minimum distance-based network selection, and complete offloading requires a large number of handovers. Though the MDP-FO approach performs the dynamic service placement and network selection operations, with full offloading it suffers with higher number of failures. Similarly, the MDP-RS approach suffers due to imperfect node selection through random allocations. The MDP-SSP approach can reduce the number of failures with proper network selection and offloading, however, due to static service placement the process still suffers with handover demands. The proposed MDP-MT approach can have the potential to provide a reliable solution with joint optimization and can be a useful solution for vehicular scenarios with higher reliability requirements.

3) Number of Failures in terms of Service Latency Another way to measure the reliability of the proposed solutions is through the fulfillment service latency

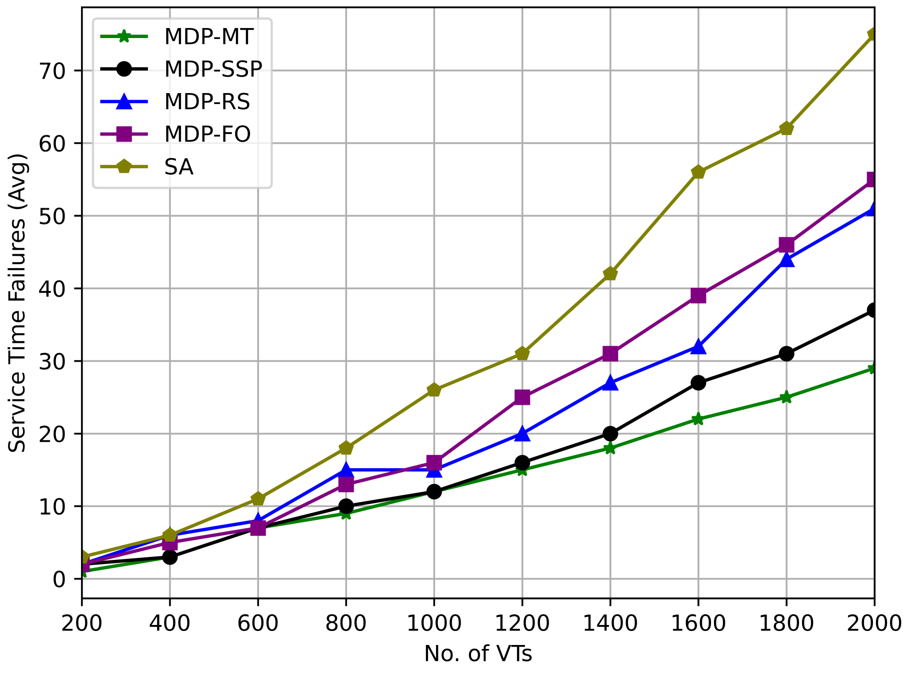


FIGURE 5.24: Percentage of VUs with service time constraint violation.

demand. In Fig. 5.24, we present the amount of service latency constraint failures for different solutions. Similarly to the previous solutions, the SA approach with static decisions suffers from several failures. On the other hand, the proposed MDP-MT solutions can reduce service latency failures through dynamic service placements, proper network selections, and offloading decisions. The other MDP solutions can reduce the overall number of failures compared to the SA method, however, their performance is suboptimal. With reduced flexibility in the offloading process (i.e., full offloading in MDP-FO or random network selection in MDP-RA) both of these methods suffer with more failures compared to the MDP-MT solutions. Also, the MDP-SSP approach with static service placements has a higher number of failures. This highlights the importance of dynamic service placement in dynamic vehicular scenarios.

Table 5.4: Average Percentage of Data Offloading.

# VNs	200	400	600	800	1000	1200	1400	1600	1800	2000
MDP-MT	.76	.74	.73	.68	.66	.63	.59	.55	.48	.47
MDP-SSP	.78	.79	.78	.73	.72	.67	.65	.62	.58	.56
MDP-RS	.65	.59	.62	.56	.51	.44	.39	.35	.32	.35
MDP-FO	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

4) Performance in terms of Offloading Percentage Table 5.4 presents the average percentage amount of data offloaded by VUs towards EC facilities. Different MDPs can have different performances based on their decision-making strategies. MDP-FO adapts the full offloading strategy resulting in 100% offloading with reduced flexibility. While MDP-RS uses the random node selection strategy, resulting in suboptimal offloading decisions. Indeed the overall percentage of offloading is reduced significantly for the MDP-RS process, adding an additional burden on the local computing resources. Similarly, the MDP-SSP approach characterized by static service placement can suffer from suboptimal offloading decisions with reduced flexibility in terms of node selection and offloading. The proposed MDP-MT, with a hierarchical decision-making process, can adapt according to the changing vehicular densities through dynamic service placement, proper node, and adequate offloading parameter selections.

5) Performance in terms of EN Selection With the presence of multiple EC layers with heterogeneous nodes, it is important to analyze the performance in terms of edge resource utilization over differing vehicular densities. For this, in Fig. 5.25, we present the simulation results in terms of percentage number of VUs selecting different edge layers with varying numbers of VUs. In the beginning, with a lower number of VUs on the road, RSUs and UAV resources are exploited by VUs for offloading their tasks. With increasing vehicular density, the overall percentage of VUs selecting RSUs reduces while the number of VUs exploiting the HAP resources increases. It should be noted that these results can be strongly influenced by the

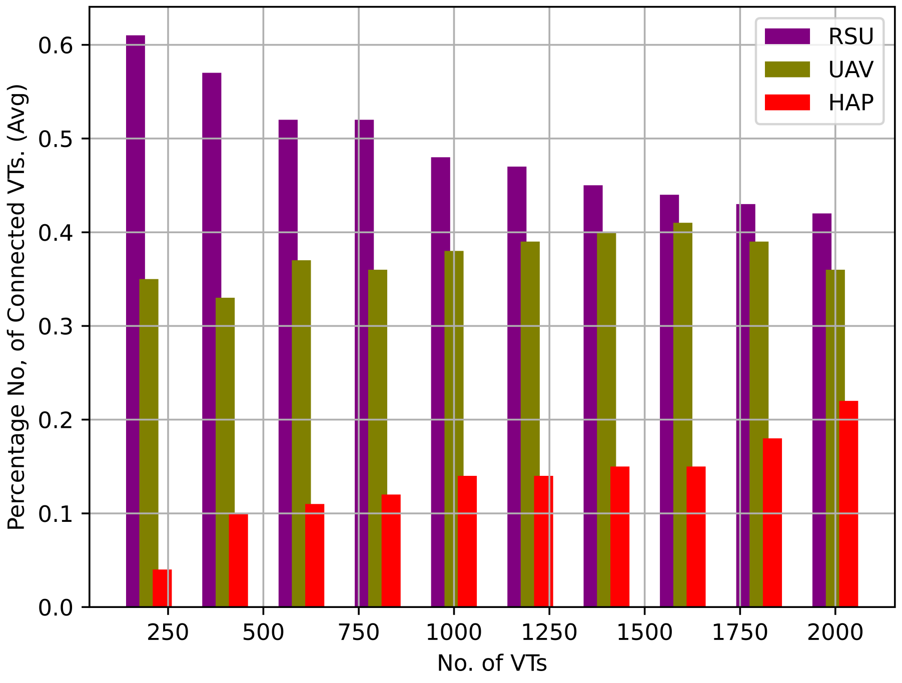


FIGURE 5.25: Avg. No. of EN Selected for Offloading.

underneath system elements. For example, since we have considered a single HAP node the overall percentage of VUs exploiting the HAP resources is always low, while RSUs with dense deployments can serve a large portion of VUs. Such trends can be explained further with other deployment options.

5.4.5 Conclusion

In this work, we have proposed a multi-time scale MDP approach for solving the joint service placement, network selection, and computation offloading problem over multiple EC platforms-enabled dynamic vehicular scenarios. The proposed approach can model the decision-making process over different time scales based on the network and user requirements. The optimization problem is formed for jointly minimizing the latency, energy, and service placement costs over different vehicular scenarios.

Advanced DQN-based solutions are considered for solving the complex MDP in finding the optimal policies reducing the overall cost with improved reliability. The numerical results acquired through a Python-based simulator show several advantages over the traditional benchmark solutions. With the complex nature of a considered joint optimization process and the proposed multi-time scale MDP, to avoid excessive discussions, in this work, we have resorted to the basic DQN approach. In recent years several new advanced forms of DRL algorithms are proposed with additional benefits. In the future, we aim to extend the proposed framework for accommodating such advanced DRL solutions with improved efficiency and reduced training costs.

Chapter 6

Distributed Intelligence for IoV

Some content of this chapter is based on the following articles [76, 114];

1) " Shinde, Swapnil Sadashiv, Arash Bozorgchenani, Daniele Tarchi, and Qiang Ni. "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems." IEEE Transactions on Vehicular Technology 71, no. 2 (2021): 2041-2057.".

2) " Shinde, Swapnil Sadashiv, and Daniele Tarchi. "Joint Air-Ground Distributed Federated Learning for Intelligent Transportation Systems." IEEE Transactions on Intelligent Transportation Systems (2023).".

6.1 Introduction

Distributed intelligence is one of the key demands of IoV users. To enable the next-generation intelligent vehicular system, novel Machine Learning (ML) solutions are required. In the case of vehicular scenarios, with the presence of IoT subsystems, a large amount of distributed data is available at the user level. To enable the traditional ML solutions, this data is required to be transmitted to the centralized servers inducing large amounts of communication overheads and corresponding costs. On the other hand with the innovative IoT solutions, the quality of data samples generated by the vehicular nodes is much improved. The size of data generated by the users can be huge, making it infeasible to transmit such huge data to the centralized servers. On the other hand, with the innovations in hardware technologies, new vehicles are having improved computation and storage facilities. This can allow vehicular terminals to train the fairly complex ML models locally. The novel ML trends such as distributed learning can be explored with such local ML model training capabilities of vehicular terminals. Additionally, it is important to explore the possibility of exploring the distributed computation communication resources of edge nodes for enabling efficient distributed learning solutions in vehicular systems. We have proposed novel distributed intelligence solutions with the support of local vehicular data and the edge computing facilities of joint Terrestrial and Non-terrestrial Networks (T-NTN). In particular, we propose novel FL solutions for integrating advanced intelligent solutions in vehicular scenarios with the support of edge computing facilities.

6.2 FL-based Computation Offloading in IoV

While developing edge intelligence solutions for IoV use cases, it is important to take into account the service demands, resource limitations, VUs mobility, etc. FL is one of the promising approaches that can be considered to solve challenging vehicular problems. However, the number of resources required to achieve the FL convergence can be higher, and implementing the complete FL process over resource-limited

Vehicular Networks (VNs) can be challenging. Therefore it is important to optimize the FL process by considering the tradeoff between the resource requirements and the achieved performance. In this work, we have developed a joint air-ground network-based FL framework for solving the computation offloading problem for the case of IoV. Next, we aim to optimize the FL and offloading process jointly to minimize the latency and energy costs.

The main contributions of this work can be summarized in:

- We define an air-ground integrated FL-inspired distributed learning platform for enabling a run-time evaluation of the computation offloading parameters. We consider the HAPs as FL servers and VUs as distributed FL devices/clients, while the RSUs act as processing devices accepting computation offloaded by the VUs acting as sources.
- We model the joint learning and offloading process through its delay and energy consumption, and then we define the joint delay and energy minimization problem as a constrained non-linear optimization problem. The main aim is to select the optimal number of FL process iterations for each VU given a target delay requirement while keeping the energy consumption under a certain level.
- Three RSU-based clustering approaches are introduced for solving the problem (i.e., full clustering, probabilistic clustering, distance-based clustering) along with a VU-based distributed approach.
- A GA evolutionary computing method is proposed for solving clustered and distributed approaches. Two other benchmark methods and one simple Heuristic approach based on a reduced size solution space are also considered for performance comparison.

- Performance evaluation is carried out under different VEC environments where the effectiveness of the proposed scheme is shown.

6.2.1 System Model and Problem Formulation

We consider an integrated air-ground network composed of one HAP, a set $\mathcal{V} = \{v_1, \ldots, v_m, \ldots, v_M\}$ of M VUs, and a set $\mathcal{R} = \{r_1, \ldots, r_n, \ldots, r_N\}$ of N RSUs, placed in the area, supposed to be modeled as a two-lane road scenario.

The generic mth VU, supposed to move in either of the two directions, is characterized by a processing capability equal to $c_{v,m}$ Floating Point Operations per Second (FLOPS) per CPU cycle, while its CPU frequency is $f_{v,m}$. Each VU is supposed to be able to communicate on a bandwidth $B^{\text{rsu}}_{v,m}$ during terrestrial communication. On the other hand, while communicating with the HAP in the non-terrestrial communication network, it is supposed to communicate with a bandwidth $B^{\text{HAP}}_{v,m}$. Many new vehicular applications and services, including autonomous driving, online gaming, multimedia content streaming infotainment services, etc., come with strict application latency requirements. Such latency constraints need to be taken into account while solving VN problems [118, 119]. Therefore, in this work, the mth VU is supposed to generate tasks to be processed, where the task x_m is identified through the tuple $\langle D_{x_m}, \Omega_{x_m}, \bar{T}_{x_m} \rangle$ where D_{x_m} is the task size in Byte, Ω_{x_m} are the requested CPU execution cycles and $\bar{T}_{x_m}$ is the maximum latency of the requested service.

The nth RSU, supposed to be in a fixed position, is characterized by a processing capability equal to $c_{r,n}$ FLOPS per CPU cycle, with CPU frequency $f_{r,n}$, and communication capabilities, supposed to be identified through a communication technology able to work on a bandwidth $B_{r,n}$ and covering an area with radius $R_{r,n}$. Each RSU provides computation offloading services to the VUs within its coverage area. In

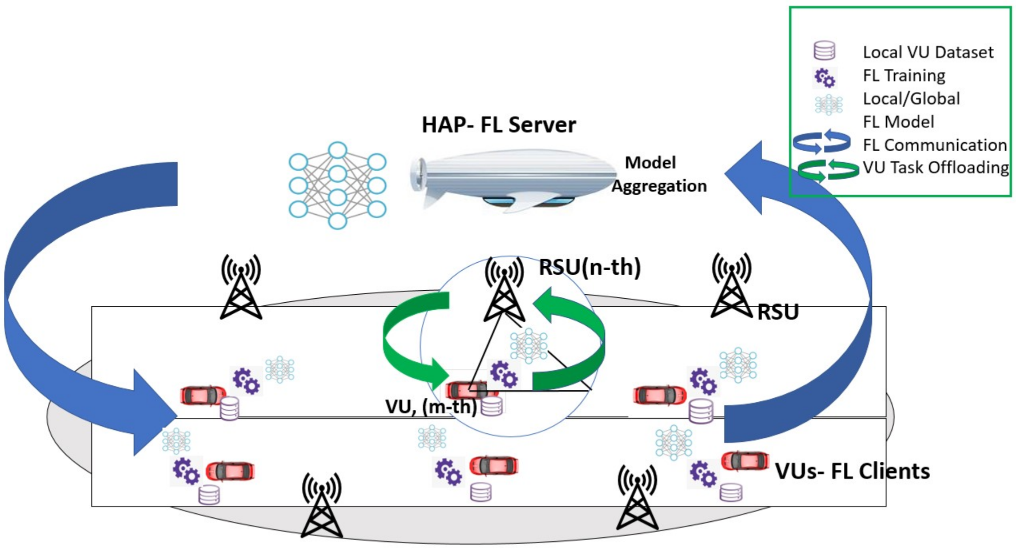


FIGURE 6.1: System Architecture

addition, the area is supposed to be under the coverage of one HAP equipped with an edge computing server with much superior computation capabilities compared with the RSU/VUs [120]. Moreover, we consider multi-beam antenna forming techniques, placed at an altitude of h_{HAP} above the ground, where each antenna beam is supposed to cover a geographical area of radius R_{HAP} and having a communication bandwidth B_{HAP}. In the following we will refer to a single beam as the coverage of the HAP. It should be noted that though HAP coverage is reduced to a single beam for notation simplicity, our approach can easily be scaled for the overall HAP coverage with multiple beams.

Fig. 6.1 highlights the main system elements of considered network architecture. It is worth to be noted that we have considered only one centralized HAP in the following. In a realistic scenario, for having a better fault tolerance, it can be complemented by ground-based 5G base stations (5G-gNB) or replaced by a decentralized HAP network, composed by multiple HAPs.

6.2.1.1 Vehicular Mobility Model

The generic mth VU is supposed to be located at position $\{x_{v,m}(t), y_{v,m}(t)\}$ at time t. Each vehicle is supposed to move along the x-axis, defining the directions of the two-lane road, with a speed $\bar{v}_m$, supposed to be constant. The nth RSU is considered to be in a fixed position $\{x_{r,n}, y_{r,n}\}$. Hence, it is possible to define the remaining distance within which the mth VU remains under the coverage of the nth RSU as,

$$\Pi_{m,n} = \sqrt{R_{r,n}^2 - \left(y_{r,n} - y_{v,m}\right)^2} \pm \left(x_{r,n} - x_{v,m}\right) \quad (6.1)$$

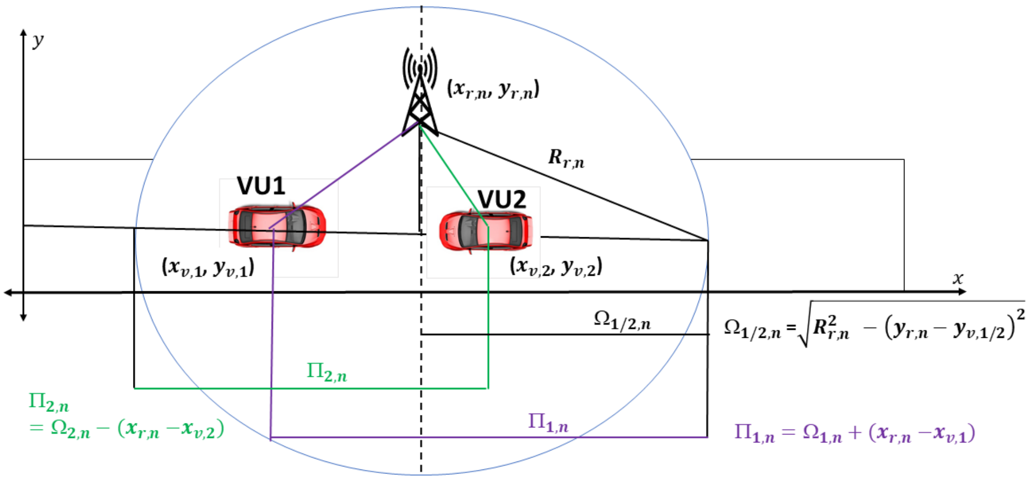
where $R_{r,n}$ is the coverage radius of the nth RSU and $\pm$ identifies the two possible directions taken by the mth VU. It is worth to be noticed that the dependency on t in (6.1) has been omitted for a better clarity; moreover, we assume that it is calculated at a time instant t when the mth VU requests an offloading service. In addition, it is worth to be noticed that (6.1) is valid only if the mth VU is within the nth RSU coverage area.

The time available by the mth VU before leaving the nth RSU coverage, i.e., *sojourn time*, is defined as:

$$T_{m,n}^{\text{soj}} = \frac{\Pi_{m,n}}{\bar{v}_m} \quad (6.2)$$

Similarly, supposing that the center beam of the HAP coverage on the ground is in a fixed position $\{x_{\text{HAP}}, y_{\text{HAP}}\}$, it is possible to define the remaining distance within which the mth VU remains under the HAP beam coverage as:

$$\Pi_{m,\text{HAP}} = \sqrt{R_{\text{HAP}}^2 - \left(y_{\text{HAP}} - y_{v,m}\right)^2} \pm \left(x_{\text{HAP}} - x_{v,m}\right) \quad (6.3)$$



y
$(x_{r,n}, y_{r,n})$
$R_{r,n}$
VU1
VU2
$(x_{v,1}, y_{v,1})$
$(x_{v,2}, y_{v,2})$
x
$\Omega_{1/2,n}$
$\Omega_{1/2,n} = \sqrt{R_{r,n}^2 - (y_{r,n} - y_{v,1/2})^2}$
$\Pi_{2,n}$
$\Pi_{2,n} = \Omega_{2,n} - (x_{r,n} - x_{v,2})$
$\Pi_{1,n}$
$\Pi_{1,n} = \Omega_{1,n} + (x_{r,n} - x_{v,1})$

FIGURE 6.2: VU Mobility Scenarios and Corresponding Distance Matrices

which is valid only if the mth VU is within the HAP coverage area as well. Hence, the HAP sojourn time can be defined as:

$$T_{m,\text{HAP}}^{\text{soj}} = \frac{\Pi_{m,\text{HAP}}}{\bar{v}_m}. \tag{6.4}$$

In Fig. 6.2, the working scenario is depicted, where we suppose, as an example, the presence of two VUs (i.e., VU1 and VU2) moving in opposite directions, where VU1 is traveling towards right and VU2 moving towards left. The available distance for each of the VUs before passing through the coverage area of RSU is determined by using (6.1). A similar analysis can be performed for determining the distances concerning the HAP.

6.2.1.2 Partial Offloading Model

The VUs are supposed to be able to offload their tasks to the RSUs within their connecting area. In order to minimize the time spent for the offloading process we assume that the mth VU is able to split its task in two portions and offload a portion $\alpha_m \in [0, 1]$ to any of the RSUs within its connecting area[1], while the remaining $(1 - \alpha_m)$ can be locally computed [121].

6.2.1.3 Task Offloading Process

When mth VU selects the nth RSU for offloading its task, the process is composed of the data transmission toward the selected RSU, task processing at the RSU, and the reception of computed data back at the VU. Each of these steps consume some amount of time and energy, as detailed in the following. In the following, we resort to the Shannon capacity formula for evaluating the data rate between any node i and j as a function of the distance between them, defined as:

$$r_{i,j}(B_i, d_{i,j}) = B_i \log_2 \left(1 + \frac{P_i^{\text{tx}} \cdot h(d_{i,j})}{N_0} \right) \tag{6.5}$$

where P_i^{tx} is the transmission power of the generic device i, $h(d_{i,j})$ is the channel gain at a distance $d_{i,j}$ between the device i and the device j, and $N_0 = N_T B_i$ is the noise power, where N_T and B_i are the noise power spectral density and bandwidth associated to the ith device during communication.

VU-RSU Communication The total time and energy required for full offloading of task x_m from the mth VU towards the nth RSU is given by,

$$T_{m,n}^{x_m,\text{tx}} = \frac{D_{x_m}}{r_{m,n}(B_{v,m}^{\text{rsu}}, d_{m,n})}, \quad E_{m,n}^{x_m,\text{tx}} = P_m^{\text{tx}} \cdot T_{m,n}^{x_m,\text{tx}}, \tag{6.6}$$

where, $r_{m,n}(B_{v,m}^{\text{rsu}}, d_{m,n})$ is the data rate between mth VU and nth RSU, that depends on the available radio resources i.e., $B_{v,m}^{\text{rsu}}$, and the distance between two devices,

[1]More specifically to be managed by the co-located server.

i.e., $d_{m,n}$. Also, P_m^{tx} is the transmission power of m-th VU while transmitting data towards the RSU.

RSU Computation The task computation at the RSU side depends on the CPU execution cycles requested by the task, i.e., Ω_{x_m} and available RSU processing resources, i.e., $c_{r,n}$ and $f_{r,n}$; hence, the processing time can be modeled as:

$$T_n^{x_m,\mathrm{c}} = \frac{\Omega_{x_m}}{c_{r,n} f_{r,n}}. \tag{6.7}$$

Here, we assume that the RSUs are connected to the electrical grid, hence their energy cost is negligible, while the VUs are in idle state, whose energy consumption can be neglected as it can be considered an unavoidable basic energy consumption.

RSU-VU Communication The completion of the task offloading process is performed by sending back the result to the VU. The time and energy required by the mth VU for receiving the result from nth RSU is given by,

$$T_{m,n}^{x_m,\mathrm{rx}} = \frac{D_{x_m,\mathrm{rx}}}{r_{n,m}(B_{r,n}, d_{n,m})}, \quad E_{m,n}^{x_m,\mathrm{rx}} = P_m^{\mathrm{rx}} \cdot T_{m,n}^{x_m,\mathrm{rx}}, \tag{6.8}$$

where, $D_{x_m,\mathrm{rx}}$ is the task size processing result at the RSU side, and $r_{n,m}(B_{r,n}, d_{n,m})$ is the downlink data rate between the nth RSU and the mth VU. P_m^{rx} is the reception power of m-th VU while receiving data from the RSU.

In general, RSUs are located in the proximity of VUs, resulting in negligible task propagation time during uplink and downlink communication. Therefore, in this work, we do not consider the propagation time when modeling the delay of the task offloading process. Thus, the total time and energy required for the complete task offloading process is,

$$\hat{T}^{\text{off}}_{m,n} = T^{x_m,\text{tx}}_{m,n} + T^{x_m,\text{c}}_{n} + T^{x_m,\text{rx}}_{m,n} \tag{6.9}$$

$$\hat{E}^{\text{off}}_{m,n} = E^{x_m,\text{tx}}_{m,n} + E^{x_m,\text{rx}}_{m,n} \tag{6.10}$$

Since we assume that the mth VU offloads a portion α_m of the task x_m towards the nth RSU, the overall time required to perform the offloading process is:

$$T^{\text{off}}_{m,n}(\alpha_m) = \alpha_m \cdot \hat{T}^{\text{off}}_{m,n} \tag{6.11}$$

where we suppose that both communication and processing latency terms scale linearly. Similarly, the overall energy consumed by the mth VU for performing the offloading process is:

$$E^{\text{off}}_{m,n}(\alpha_m) = \alpha_m \cdot \hat{E}^{\text{off}}_{m,n} \tag{6.12}$$

6.2.1.4 Local VU Computation Process

Each VU is able to locally compute its task and the amount of time and energy required is based on its processing resources, i.e., c_m and f_m. Hence, the local processing time and energy consumption is:

$$T^{x_m,\text{c}}_{m} = \frac{\Omega_{x_m}}{c_{v,m} f_{v,m}}, \quad E^{x_m,\text{c}}_{m} = P^{\text{c}}_{m} \cdot T^{x_m,\text{c}}_{m}, \tag{6.15}$$

where, P^{c}_{m} is the computational power used during local task computation at the mth VU. Due to the partial offloading, the amount of time and energy required for the local computation at the mth VU is:

$$T^{\text{loc}}_{m}(\alpha_m) = (1 - \alpha_m) T^{x_m,\text{c}}_{m} \tag{6.16}$$

$$E_m^{\mathrm{loc}}(\alpha_m) = (1-\alpha_m)E_m^{x_m,\mathrm{c}} \tag{6.17}$$

where α_m is the portion of the task to be offloaded by the mth VU; hence, the overall processing time for the task x_m results:

$$T_m^{x_m}(\alpha_m) = \max\left\{T_{m,n}^{\mathrm{off}}(\alpha_m), T_m^{\mathrm{loc}}(\alpha_m)\right\} \tag{6.18}$$

where $T_{m,n}^{\mathrm{off}}(\alpha_m)$ is the time needed for offloading the portion of a task $\alpha_m \cdot x_m$ to the nth RSU, while $T_m^{\mathrm{loc}}(\alpha_m)$ is the time for locally processing the remaining task $(1-\alpha_m)\cdot x_m$ by the mth VU. We suppose that offloading and local computation can be performed in parallel. Similarly, the overall processing energy for the task x_m results:

$$E_m^{x_m}(\alpha_m) = E_{m,n}^{\mathrm{off}}(\alpha_m) + E_m^{\mathrm{loc}}(\alpha_m) \tag{6.19}$$

where $E_{m,n}^{\mathrm{off}}(\alpha_m)$ is the energy consumed offloading the portion of a task $\alpha_m \cdot x_m$ to the nth RSU, while $E_m^{\mathrm{loc}}(\alpha_m)$ is the energy consumed during locally processing the remaining task $(1-\alpha_m)\cdot x_m$ on mth VU.

6.2.1.5 Partial Offloading Problem

The partial computation offloading problem corresponds to set the offloading parameters α_m in an optimal way such that service latency ($\bar{T}_{x_m}$), sojourn time ($T_{m,n}^{\mathrm{soj}}$) and the overall energy consumption constraints are respected. To this aim, we assume that in an energy efficient partial offloading operation, the energy spent for the task x_m ($E_m^{x_m}(\alpha_m)$) with offloading parameter α_m, is less than the amount of energy required to completely compute it locally (i.e., $E_m^{x_m,\mathrm{c}}$), since otherwise offloading would not be beneficial. Therefore, the joint latency and energy constrained optimization problem corresponds to find the optimal $\mathcal{A} = \{\alpha_1, \ldots, \alpha_m, \ldots, \alpha_M\}$ parameters such that:

$$\mathbf{P1}: \ \mathcal{A}^* = \underset{\mathcal{A}}{\text{argmin}} \left\{ \frac{1}{M} \sum_{m=1}^{M} \left(\eta_1 T_m^{x_m}(\alpha_m) + \eta_2 E_m^{x_m}(\alpha_m) \right) \right\} \tag{6.20}$$

subject to the following constraints,

$$T_m^{x_m}(\alpha_m) \leq \bar{T}_{x_m}, \qquad \forall m \tag{6.21a}$$

$$T_{m,n}^{\text{off}}(\alpha_m) \leq T_{m,n}^{\text{soj}}, \qquad \forall m, \forall n \tag{6.21b}$$

$$E_m^{x_m}(\alpha_m) \leq E_m^{x_m,\text{c}}, \qquad \forall m \tag{6.21c}$$

$$\sum_{n=1}^{N} a(m,n) \leq 1, \qquad \forall m \in M \tag{6.21d}$$

$$\sum_{m=1}^{M} a(m,n) \cdot B_{v,m}^{\text{rsu}} \leq B_{r,n}, \qquad \forall n \in N \tag{6.21e}$$

$$0 \leq \alpha_m \leq 1, \qquad \forall m \in M \tag{6.21f}$$

$$0 \leq \eta_1, \eta_2 \leq 1, \tag{6.21g}$$

where (6.21a) shows that the total task processing time for each VU should be limited by the task latency requirement and (6.21b) represents that each VU should complete the computation offloading process while it is in the RSU coverage for avoiding additional latency costs. From (6.21c), the overall processing energy of task should be upper bounded by the amount of energy required to compute it locally. A binary assignment variable $a(m,n)$ is considered equal to 1 if mth VU is assigned to the nth RSU, and 0 otherwise. According to (6.21d), each VU can offload tasks to no more than one RSU, while (6.21e) shows that the bandwidth resources available for all active VUs[2] in a particular RSU coverage is upper bounded by its bandwidth. Eq. (6.21f) limits the offloading parameter value between 0 and 1. Moreover, in (6.21g), η_1 and η_2 are two weight coefficients between 0 and 1, for

[2]We assume that only a subset of the VUs, named active, have data to process, and potentially to be offloaded to an RSU.

balancing latency and energy consumption.

In a real scenario the amount of data to be offloaded from each VU towards an RSU while respecting the system constraints is hard to be estimated; several factors, including VUs position, velocity, directions, RSU resources, task requirements, surrounding environmental conditions, make the problem hard to be solved. Many of these parameters are hard to be accessed given their stochastic behaviors. Therefore, finding a set of optimal offloading parameters ($\mathcal{A}^*$) in a highly dynamic environment like VN is a challenging problem to be solved, and advanced optimization methods are needed. Belonging to the class of the ML approaches, FL has been recently introduced as an effective way for performing data augmentation and significantly reducing the communication overhead in comparison with direct data-sample exchanges, allowing also to enhance VUs privacy issues. In order to properly address the latency and energy constrained offloading problem defined in (6.20), we propose to exploit a FL framework for estimating the set $\mathcal{A}$, composing the offloading portions of all VUs, based on the VU side parameters.

In Fig. 6.3 we provide a more detailed step-by-step view of the considered joint FL and task offloading process optimization problem and the proposed solution methodologies; it is possible to notice that the VUs parameters act as input for the FL-inspired distributed process (Step 1), whose goal is to properly set the number of iterations to be performed (Step 2) in order to have a proper solution for setting the offloading parameters (Step 3) to be later used by each VU (Step 4).

6.2.1.6 Federated Learning Model

FL is based on the idea that the same ML algorithm is present at both FL server and FL clients' sides, where a centrally located FL server assists distributed clients

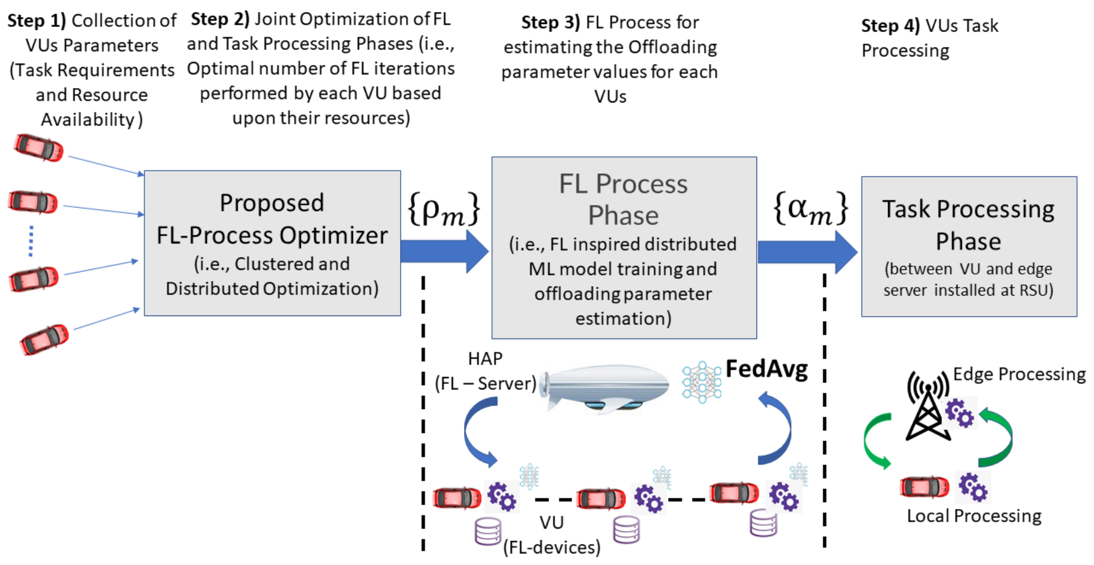

Step 1) Collection of VUs Parameters (Task Requirements and Resource Availability)
Step 2) Joint Optimization of FL and Task Processing Phases (i.e., Optimal number of FL iterations performed by each VU based upon their resources)
Step 3) FL Process for estimating the Offloading parameter values for each VUs
Step 4) VUs Task Processing
Proposed FL-Process Optimizer (i.e., Clustered and Distributed Optimization)
{ρ_m}
FL Process Phase (i.e., FL inspired distributed ML model training and offloading parameter estimation)
{α_m}
Task Processing Phase (between VU and edge server installed at RSU)
HAP (FL – Server)
FedAvg
VU (FL-devices)
Edge Processing
Local Processing

FIGURE 6.3: Proposed scheme for the joint FL and task-offloading processes optimization.

during the learning process. Instead of only executing the ML algorithm in a centralized server node, it is executed in a federated way among all the involved nodes through the exchange of a set of parameters defining the weights of the implemented ML algorithm. To do this, the FL process is composed of several steps: information exchange between FL-server and devices for initializing the learning model over devices, local device training, parameters exchange over wireless links between devices and the FL-server, parameter collection and aggregation on the server. In the FL process, we assume that the HAP acts as FL server, assisting the VUs acting as FL clients for making the offloading decision. For each offloading request, the VUs perform numerous FL iterations with the HAP aiming at properly setting the offloading portion toward the selected RSU. It has to be noticed that the HAP computation infrastructure can be implemented by resorting to the function virtualization approach through different virtualization technologies, e.g., virtual machines, containers, and hypervisors, for performing the FL process. Moreover, the interaction with FL clients can happen through predefined interfaces (e.g., implementing the REST API technology) allowing a smarter interaction. However, such considerations are beyond the scope of this work, that instead mainly focuses on the optimization of the joint FL-offloading framework.

Even though FL allows to reach a global optimum in distributed environments, the dynamicity of VN scenarios introduces an additional challenge. Indeed, FL process cannot be considered as a granted process, as it consumes resources by itself. Hence, FL is executed at the cost of a reduction of resources that can be given to the offloading process. It is however, clear from past studies that the number of FL iterations required for reaching a predefined convergence value can be upper bounded [122, 123, 124, 125] depending on several factors, including the ML model, number of users participating in the training process, number of local iteration on the device, type of radio environment, quality of data, etc. Therefore, without loss of generality, we consider that after ρ^{opt} FL iterations each VU will be able to estimate the optimal offloading parameter α_m^{opt}, where $\rho^{\text{opt}} = \frac{\mathcal{K}}{\bar{M}}$; $\mathcal{K}$ can be considered as a numerical constant setting the overall number of FL iterations required to achieve the convergence, while $\bar{M}$ is the number of VUs participating in the FL training process, so higher the participating VUs, lower the required iterations, respecting the FL process behavior.

In this work, we assume the learning process converges after ρ^{opt} FL iterations, when each VU is able to estimate the offloading parameters[3]. For the purpose of this work, we consider that, in case we stop the FL process in advance, some estimation error should be considered, as later explained. Since each FL iteration requires a certain amount of communication and computational resources, performing ρ^{opt} iterations over all VUs can be challenging and sometimes might not be feasible given the limited VUs resources and the latency constraints imposed by both service requirements and sojourn time. The additional energy cost of each FL iteration can also limit

[3]In the case of a practical system, the convergence can be bounded by some stopping criteria, e.g., loss function value.

the number of FL iterations performed by VUs. Therefore, in this work we consider that the generic m-th VU is able to perform up to ρ_m FL iterations with $\rho_m \leq \rho^{\text{opt}}$. The set $\mathcal{I} = \{\rho_1, \cdots, \rho_m, \cdots \rho_M\}$ contains the number of FL iterations performed by each VU.

In order to understand the impact of the FL process we can now introduce the FL iterations latency, the corresponding energy consumption and the joint optimization model.

6.2.1.7 FL Computation Model

The FL computation corresponds to the local training of the ML model based on the on-device dataset. In local device training, the mth VU has to compute the local parameter set $w^{it}_{v,m}$ through the dataset having size K_m data samples; if we assume that, for every iteration, the total number of FLOPs required for each data sample d is ψ_d, the time and energy consumed during FL process at the mth device is given by [84]:

$$T^{\text{FL,c}}_m = \frac{\sum_{d=1}^{K_m} \psi_d}{c_{v,m} f_{v,m}}, \quad E^{\text{FL,c}}_m = P^{\text{c}}_m \cdot T^{\text{FL,c}}_m. \tag{6.22}$$

We suppose for simplicity that the on-device FL processing time and energy is the same for every iteration. Conversely, the FL server is limited to the model aggregation, whose time and energy is considered as negligible given the abundant available resources at HAP.

6.2.1.8 FL Communication Model

In FL, the devices communicate the local model updates towards the HAP in uplink and receive back the updated global model parameters in downlink. Both uplink and downlink communication processes are characterized by transmission and propagation delays, due to the high distance between VUs and HAP. The propagation time required for each FL iteration is given by,

$$T_{m,it}^{\text{FL,prop}} = 2 \cdot \frac{d_{m,\text{HAP}}}{\sigma}, \quad \forall m \tag{6.23}$$

where σ is the propagation speed in the considered transmission medium, $d_{m,\text{HAP}}$ is the distance between the mth VU and the HAP, which can be calculated by using HAP altitude (h_{HAP}) and the mth VU location through simple algebraic passages, and the multiplication by 2 is due to the two-way propagation delay. During the FL processing, at each iteration it the mth VU sends the parameters set $w_{v,m}^{it}$ to the HAP. Supposing that $|w_m^{it}|$ represents the data size of the parameters set expressed in bits [126], the uplink transmission time and energy for the FL parameters in the itth iteration is:

$$T_{m,it}^{\text{FL,tx}} = \frac{|w_{v,m}^{it}|}{r_{m,\text{HAP}}^{it}(B_{v,m}^{\text{HAP}}, d_{m,\text{HAP}})}, \quad E_{m,it}^{\text{FL,tx}} = P_m^{\text{tx}} \cdot T_{m,it}^{\text{FL,tx}}, \tag{6.26}$$

where, $r_{m,\text{HAP}}^{it}$ is the uplink transmission rate between mth VU and the HAP during the itth iteration, which is a function of the VUs bandwidth ($B_{v,m}^{\text{HAP}}$), and the distance ($d_{m,\text{HAP}}$) between the mth VU and the HAP, modeled through the Shannon capacity formula under Rice fading conditions [127]. Since the HAP is accessed by multiple VUs, we assume for simplicity that the HAP bandwidth is equally shared among the connected VUs. Also, P_m^{tx} is the VUs, transmission power while communicating with HAP.

In general, HAP needs to wait for all training VUs to transmit their model parameters before performing the averaging operation. Therefore, the FL transmission time for the itth iteration is given by,

$$T_{it}^{\mathrm{FL,tx}} = \max_{m} \left\{ T_{m,it}^{\mathrm{FL,tx}} \right\}, \quad \forall m \tag{6.27}$$

The HAP performs the aggregation of the received model parameters (e.g., FedAvg [122]) to create a global parameter vector w_G^{it} for the next iteration and transmits it back towards VUs over the downlink communication links. Therefore, in downlink, the global parameters transmission time and energy are given by,

$$T_{m,it}^{\mathrm{FL,rx}} = \frac{|w_G^{it}|}{r_{\mathrm{HAP},m}^{it}(B_{\mathrm{HAP}}, d_{\mathrm{HAP},m})}, \; E_{m,it}^{\mathrm{FL,rx}} = P_m^{\mathrm{rx}} \cdot T_{m,it}^{\mathrm{FL,rx}} \tag{6.28}$$

where $r_{\mathrm{HAP},m}^{it}$ is the downlink transmission rate between the HAP and the mth VU during the itth iteration when the global parameter set is broadcast. P_m^{rx} is the power consumed while receiving data from HAP. Hence, the total time and energy required for a single FL iteration can be detailed as:

$$T_{m,it}^{\mathrm{FL}} = T_m^{\mathrm{FL,c}} + T_{m,it}^{\mathrm{FL,prop}} + T_{it}^{\mathrm{FL,tx}} + T_{m,it}^{\mathrm{FL,rx}} \tag{6.29}$$

$$E_{m,it}^{\mathrm{FL}} = E_m^{\mathrm{FL,c}} + E_{m,it}^{\mathrm{FL,tx}} + E_{m,it}^{\mathrm{FL,rx}} \tag{6.30}$$

6.2.1.9 Joint Offloading and Federated Learning Model

Since the FL process is based on multiple iterations for exchanging the ML model parameters, it is possible to write the total time and energy for the FL process when focusing on the mth VU as,

$$T_m^{\mathrm{FL}}(\rho_m) = \sum_{it=1}^{\rho_m} T_{m,it}^{\mathrm{FL}}, \qquad E_m^{\mathrm{FL}}(\rho_m) = \sum_{it=1}^{\rho_m} E_{m,it}^{\mathrm{FL}} \tag{6.31}$$

where ρ_m is the number of FL iterations performed by mth VU, $T^{\mathrm{FL}}_{m,it}$ is the amount of time spent, and $E^{\mathrm{FL}}_{m,it}$ is the amount of energy consumed for the itth iteration of the FL process depending on both FL communication and computation performance. The time needed for completing both FL iterations and task processing has to be constrained by the maximum service latency requirement, given by:

$$T_m(\rho_m, \alpha_m) = T^{\mathrm{FL}}_m(\rho_m) + T^{x_m}_m(\alpha_m) \leq \bar{T}_{x_m} \tag{6.32}$$

Also the energy consumed for completing both FL iterations and task processing has to be constrained by the energy required to compute a complete task locally, given by:

$$E_m(\rho_m, \alpha_m) = E^{\mathrm{FL}}_m(\rho_m) + E^{x_m}_m(\alpha_m) \leq E^{x_m,\mathrm{c}}_m \tag{6.33}$$

Due to the dynamicity of the vehicular environment, computation offloading and FL process latencies should also be bounded by the VUs sojourn times under RSU and HAP beam coverage. Since the HAP is acting as an FL server, the whole FL phase should be completed by the HAP sojourn time, hence:

$$T^{\mathrm{FL}}_m(\rho_m) \leq T^{\mathrm{soj}}_{m,\mathrm{HAP}} \tag{6.34}$$

In addition, each VU should finish the offloading process within the RSU sojourn time. Thus,

$$T^{\mathrm{FL}}_m(\rho_m) + T^{\mathrm{off}}_{m,n}(\alpha_m) \leq T^{\mathrm{soj}}_{m,n} \tag{6.35}$$

It is worth to be noticed that the sojourn time does not affect the overall processing time, while only the offloading time, since the local computation can be performed also out of the RSU coverage.

6.2.1.10 Problem Formulation

Following (6.11), (6.16), (6.18), (6.29), (6.31), and (6.32) the total time $T_m(\rho_m, \alpha_m)$ required for both phases (i.e., FL and task processing) can be determined. Similarly, from (6.12), (6.17), (6.19), (6.30), (6.31), and (6.33) the total energy $E_m(\rho_m, \alpha_m)$ required for both phases can be calculated. The proposed optimization model aims at minimizing the total time and energy by properly setting the offloading parameters and the FL iterations used for determining the offloading parameters itself. Hence, the problem in (6.20) can be rewritten as:

$$\mathbf{P2}: (\mathcal{I}^*, \mathcal{A}^*) = \underset{\mathcal{I},\mathcal{A}}{\operatorname{argmin}} \left\{ \frac{1}{M} \sum_{m=1}^{M} \left(\eta_1 \cdot T_m \left(\rho_m, \alpha_m \right) + \eta_2 \cdot E_m \left(\rho_m, \alpha_m \right) \right) \right\} \tag{6.36}$$

subject to the constraints (6.21d)-(6.21g), (6.32)-(6.35), and,

$$\sum_{m=1}^{M} B_{v,m}^{\text{HAP}} \leq B_{\text{HAP}} \tag{6.37a}$$

$$0 \leq \rho_m \leq \rho^{\text{opt}} \qquad \forall v_m \in \mathcal{V} \tag{6.37b}$$

where (6.32) is the service latency requirement reformulating (6.21a) including the FL processing time. Also, (6.33) is the reformulated energy constraint defined in (6.21c) with FL process energy. Eq. (6.34) provides an upper bound for the FL process depending on the HAP sojourn time and (6.35) is the reformulated version of (6.21b), defining the upper bound of both task offloading and FL process as the RSU sojourn time: each vehicle should offload the computation data to the RSU and receive results before it leaves its coverage area. According to (6.37a), the sum of bandwidth resources available for all VUs in non-terrestrial communication links should be upper bounded by the HAP bandwidth resources. Eq. (6.37b) upper bounds the number of iterations performed by each VU to ρ^{opt}.

6.2.1.11 Federated Offloading parameter estimation

Solving the problem defined in (6.36) requires finding two sets of optimization variables $(\mathcal{I}, \mathcal{A})$ and thus is hard to be solved. However, $(\mathcal{I}, \mathcal{A})$ are not two separate sets of variable. As more iterations are performed, higher is the reliability with which the offloading parameter is estimated through the FL process. Hence, the offloading parameter α_m can be modeled as function of the number of FL iterations performed with the aim of estimating the optimal α_m^{opt}, i.e., $\alpha_m = \alpha_m(\rho_m)$. Without loss of generality, we assume in the following that in case the m-th VU cannot participate in the FL process, the offloading parameter is $\alpha_m^0 = \alpha_m(\rho_m = 0)$, while in case it can perform ρ^{opt} iterations, the estimated offloading parameter is $\alpha_m^{\text{opt}} = \alpha_m(\rho^{\text{opt}})$. In any other case, the estimated value α_m is a function of ρ_m FL iterations that are performed by the mth VU. The exact relationship between α_m and ρ_m is hard to be set since it depends on several factors such as FL environment, number of VUs participating in the FL process, the communication medium between FL clients and server, etc. To the best of our knowledge there is no model in the literature aiming at setting the aforementioned relationship. Therefore, without loss of generality, we consider here that the estimated α_m can be modeled as a stochastic value whose distribution follows a *truncated normal distribution* with mean μ and variance σ^2, where $0 \leq \alpha_m \leq 1$, since α_m is bounded between 0 and 1 by definition. Therefore, it is possible to define the probability density function $f_{\alpha_m}(\cdot)$ of α_m as,

$$f_{\alpha_m}(\alpha_m; \mu, \sigma) = \begin{cases} \frac{1}{\sigma} \frac{\xi\left(\frac{\alpha_m - \mu}{\sigma}\right)}{\Delta\left(\frac{1-\mu}{\sigma}\right) - \Delta\left(\frac{-\mu}{\sigma}\right)} & \text{if } 0 \leq \alpha_m \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.38}$$

where, $\xi(\cdot)$ and $\Delta(\cdot)$ are, respectively, the probability density function of the related standard normal distribution and its cumulative distribution function, i.e.,

$$\xi(\omega) = \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{\omega^2}{2}\right)}, \quad \Delta(\kappa) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{\kappa}{\sqrt{2}}\right)\right).$$

In this work we assume that the mean value of the distribution of α_m, i.e., μ, and its variance, σ^2, are equal to

$$\mu = \alpha_m^{\text{opt}}(\rho_m), \quad \sigma^2 = \left(\gamma \cdot \frac{\rho^{\text{opt}} - \rho_m}{\rho^{\text{opt}}}\right)^2 \tag{6.39}$$

where γ is a numerical constant, used for controlling the variance of the model. It is worth to be noticed that the variance is defined in a way that higher ρ_m, lower is the variance. This corresponds to say that increasing the number of iterations reflects in a more reliable estimation of α_m provided that $\rho_m \leq \rho^{\text{opt}}$. Moreover, the higher the iterations to be performed, the higher is the time spent in the FL phase, so the lower is the time left for the offloading phase. This is the reason why μ is also function of the iterations. This is consistent with the FL process where more FL iterations turn out in a better estimation of the offloading parameter. During simulations, $f_{\alpha_m}(\alpha_m; \mu, \sigma)$ is used for estimating the α_m for every mth VU, whose quality will depend upon the number of FL iterations performed compared with ρ^{opt}. A qualitative representation is reported in Fig. 6.4 with $\rho^{\text{opt}} = 25$, where as the number of iterations increases, both the distribution variance and the average optimal offloading parameter become smaller, leaving less time for the offloading operation.

According to (6.35) both FL and task offloading processes should be completed within available sojourn time. Fig. 6.5 shows the impact of the constraint (6.35) on the considered vehicular environment. In particular, we can notice that at the

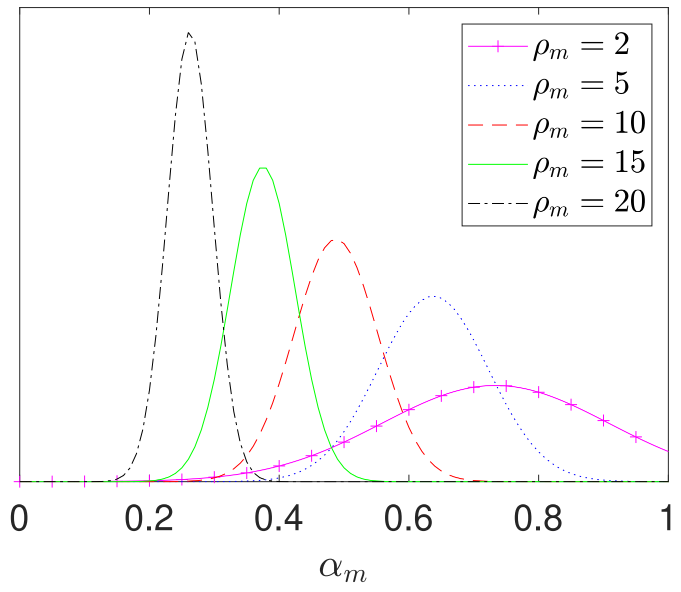
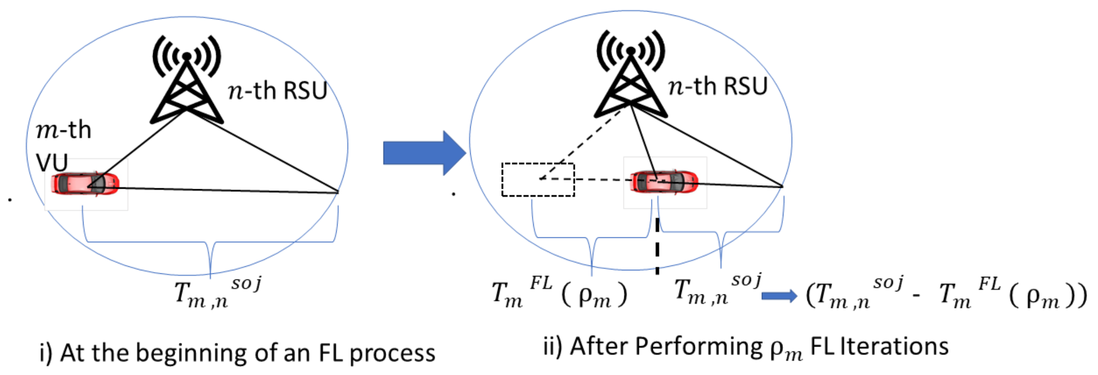


FIGURE 6.4: Truncated Normal Distribution of α_m as a function of the FL iterations.



n-th RSU
m-th VU
$T_{m,n}^{soj}$
i) At the beginning of an FL process
n-th RSU
$T_m^{FL}(\rho_m)$
$T_{m,n}^{soj}$
$(T_{m,n}^{soj} - T_m^{FL}(\rho_m))$
ii) After Performing ρ_m FL Iterations

FIGURE 6.5: FL process impact over the offloading parameter value

beginning, the joint FL and offloading process is bounded by the sojourn time. As the VU moves, despite some iterations that are performed, the remaining time for completing the FL process and starting the offloading is reduced, due to the lower remaining sojourn time.

From the previous description, it is clear that given a certain amount of time, we have to trade-off between offloading and FL processes. Let us introduce now a new parameter, named $\beta_m \in [0, 1]$, modeling the portion of time allocated for the FL process of the mth VU. If $\beta_m = 0$, the whole time is allocated for the task processing

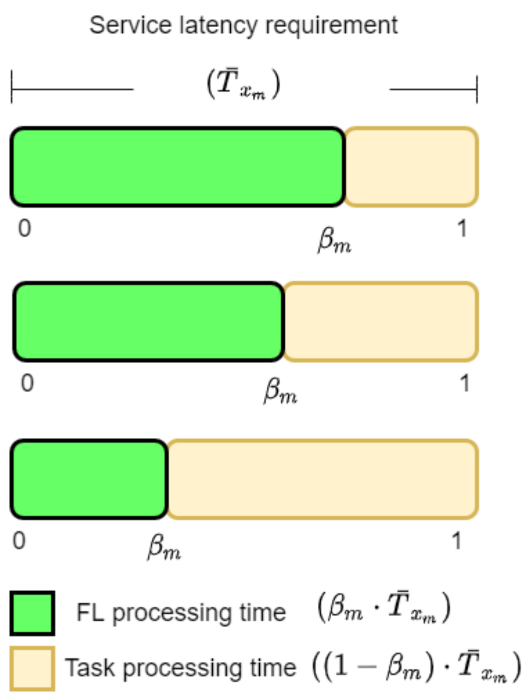

Service latency requirement
$(\bar{T}_{x_m})$
0
β_m
1
0
β_m
1
0
β_m
1
FL processing time $(\beta_m \cdot \bar{T}_{x_m})$
Task processing time $((1-\beta_m)\cdot\bar{T}_{x_m})$

FIGURE 6.6: FL and Task processing time sharing.

phase, while if $\beta_m = 1$ the mth VU uses the whole available time for the FL phase. Considering the target latency of the tasks generated by each VU as a reference time interval, it is possible to set the maximum number of possible iterations for the FL process:

$$\rho_m(\beta_m) \text{ s.t. } T_m^{\text{FL}}(\beta_m) = \sum_{it=1}^{\rho_m(\beta_m)} T_{m,it}^{\text{FL}} \leq \beta_m \cdot \bar{T}_{x_m} \tag{6.40}$$

where $\mathcal{B} = \{\beta_1, \ldots, \beta_m, \ldots, \beta_M\}$. Each VU performs numerous FL iterations aiming at finding the optimal offloading amount to be transferred towards RSU, where any additional FL iteration reduce the variance in (6.38), i.e., its reliability, while reducing its average value.

Fig. 6.6 shows the available resources for both phases as a function of β_m. As β_m increases VU spends more time on the FL process through additional iterations, which reduces the available time for the processing phase since both phases should be completed within the requested service latency.

In the end, the optimization problem defined in (6.36) can be rewritten as,

$$\textbf{P3} : \mathcal{B}^* = \underset{\mathcal{B}}{\operatorname{argmin}} \left\{ \frac{1}{M} \sum_{m=1}^{M} (\eta_1 T_m(\rho_m(\beta_m), \alpha_m(\beta_m)) + \eta_2 E_m(\rho_m(\beta_m), \alpha_m(\beta_m))) \right\} \tag{6.41}$$

subject to the constraints (6.21d)-(6.21g), (6.32)-(6.35), (6.37), (6.40) and,

$$0 \leq \beta_m \leq 1 \qquad \forall v_m \in \mathcal{V} \tag{6.42}$$

where (6.40) limits the maximum number of FL iterations performed by each VU based on the available FL process time and, according to (6.42), β_m can take any value between 0 and 1.

6.2.2 Proposed Solutions

The solution space dimension for the problem **P3** can be estimated as $\mathcal{SP} = \Theta^{(M)}$, where Θ is the number of possible values taken by β_m, i.e., the smaller step size for β_m discretization, the bigger is the solution space. In a certain service area, the number of VUs requesting services can also be huge. Therefore, despite being simplified with respect to **P2**, solving **P3** for the whole set $\mathcal{V}$, even for a discrete solution space, is computationally expensive and requires exploring a huge solution space $\mathcal{SP}$; thus sub-optimal approaches operating on a subspace of $\mathcal{SP}$ are required. In order to address the problem, first, we propose an RSU-based clustering approach where each RSU performs the optimization for the VUs under its coverage. As a second scheme, we consider a distributed approach, where each VU performs the optimization by itself without considering the surrounding VUs. In both cases, a GA is proposed as the solution methodology.

In order to simplify the problem we assume that each VU will be assigned to the nearest RSU for computation offloading, hence:

$$a(m,n) = 1 \Longleftrightarrow n = \underset{n'}{\operatorname{argmin}}\{d_{m,n'}\} \quad \forall m, n' \tag{6.43}$$

where $d_{m,n'}$ is the distance between the mth VU and the n'th RSU. Each VU starts by downloading the FL model from the HAP and infers the initial offloading parameter α_m^0, supposed to be a random value between 0 and 1.

6.2.2.1 Clustered Approach

In the cluster-based sub-optimal approach we assume that it is possible to find β_m by considering the active VUs (i.e., VUs requesting offloading services) under each RSU coverage, where M_n corresponds to the VUs managed by the nth RSU. The RSU communication and computing resources are supposed to be equally shared among all active VUs in its coverage area. The solution vector $\mathcal{B}_n = \{\beta_1, \beta_2, \cdots, \beta_{M_n}\}$ is composed by the M_n values for all the VUs connected to nth RSU. We aim to determine $\mathcal{B}_n^*$, the optimal parameter set for the nth RSU. The overall optimal $\mathcal{B}^*$ can be determined by merging the solutions from all RSUs, i.e., $\mathcal{B}^* = \cup_n \mathcal{B}_n^*$. The problem originally formulated in (6.41) is thus modified as

$$\mathcal{B}_n^* = \underset{\mathcal{B}_n}{\operatorname{argmin}} \left\{ \frac{1}{M_n} \sum_{m=1}^{M_n} [\eta_1 \cdot T_m(\rho_m(\beta_m), \alpha_m(\beta_m)) + \eta_2 \cdot E_m(\rho_m(\beta_m), \alpha_m(\beta_m))] \right\} \tag{6.44}$$

In Algorithm 12 the steps used for the RSU based clustered optimization are presented. At the beginning, VUs are assigned to the RSUs based on the minimum distance criterion in (6.43), from which the number of VUs requesting services from each RSU is determined (Line 1-2). After this, the optimal set of $\mathcal{B}_n$ values for all the VUs associated with a given RSU is determined by using (6.44) (Line 3-6). In the end, the algorithm returns the solution set of all RSUs (Line 7).

Algorithm 12 Clustered Approach

```
Input: $N, M, \{d_{m,n}\}$
Output: $\mathcal{B}^*$

1: $a(m,n) = 1 \iff n = \text{argmin}_{n'}\{d_{m,n'}\} \quad \forall m.$
2: Find $M_n = \sum_{m=1}^{M} a(m,n), \forall n$
3: for all $n = 1, \cdots N$ do
4:     $\forall m \in M_n$
5:     Find $\mathcal{B}_n^*$ by solving (6.44)
6: end for
7: return $\mathcal{B}^* = \{\mathcal{B}_1^*, \cdots, \mathcal{B}_n^*, \cdots \mathcal{B}_N^*\}$
```

Clustering Policies In order to better understand the impact of the clustered approach we have considered three different clustering policies.

- **Full Clustering Policy (FC) -** In this policy, all the active M_n VUs of nth RSU cluster participate to the FL process before performing offloading, hence,

$$M_n = \sum_{m=1}^{M} a(m,n) \quad \forall n$$

- **Probabilistic Clustering Policy (PC) -** In this approach, we randomly classify the M_n VUs of nth RSU cluster into two subgroups $\hat{M}_n^1$ and $\hat{M}_n^2$. VUs belonging to $\hat{M}_n^1$ perform FL process with optimal β_m^* determined through (6.44) while VUs in $\hat{M}_n^2$ performs offloading with initially estimated offloading parameter α_m^0, i.e., $\beta_m = 0$. By this policy we would like to understand the impact of the VUs when participating to the FL process. The classification of VUs into two subgroups is based on a Bernoulli distribution where the probability of the m-th VU being in $\hat{M}_n^1$ is p, i.e., $P(m \in \hat{M}_n^1) = p$ and the probability being in $\hat{M}_n^2$ is $(1-p)$, i.e., $P(m \in \hat{M}_n^2) = (1-p)$.

- **Distance-Based Clustering Policy (DBC) -** In this approach the selection of the VUs is based on the available distance before they move out of the RSU coverage. In this policy we would like to give more importance to those VUs staying longer within the same RSU coverage; hence, we select them for performing FL. Therefore for the nth RSU we have:

$$\hat{M}_n^1 = \left\{ m | \Pi_{m,n} \geq \hat{\Pi},\ m \in M_n \right\}$$

where $\hat{M}_n^1$ are VUs that perform FL iterations before the computation offloading process with optimal β_m determined through (6.44). $\hat{\Pi}$ is the distance bound used for partitioning VUs into two groups. The remaining VUs, will not participate into the FL training process, i.e., $\beta_m = 0$ and given by,

$$\hat{M}_n^2 = \left\{ m | m \notin \hat{M}_n^1,\ m \in M_n \right\}$$

such that $M_n = \hat{M}_n^1 \cup \hat{M}_n^2$

6.2.2.2 Distributed Approach

Due to its dynamic nature, predicting the exact VUs number and their characteristics even if within the same RSU is a difficult task. Some of the main reasons include VUs unpredictable velocity, directions, drivers' behaviors, different types of vehicles, etc. Moreover, in many situations, privacy-protective VUs are reluctant to share their information with surrounding nodes, limiting the VUs capability for understanding the surrounding environment. In this situation, each VU has to offload computation data towards RSU without knowing how many other VUs have already requested the services from that particular RSU with certain assumptions over available RSU resources. In such situations, VUs can act selfishly and assume that no other VUs have requested services from a selected RSU and its complete resource pool can be used. Here, we propose a VU-based distributed approach where VU makes similar assumptions while offloading data towards RSU nodes. Thus, in the VU-based distributed approach, we consider that VUs are not aware of nearby competing VUs and perform the optimization without considering them. The problem originally formulated in (6.41), is modified as

$$\beta_m^* = \underset{\beta_m}{\operatorname{argmin}}\{\eta_1 \cdot T_m(\rho_m(\beta_m), \alpha_m(\beta_m)) + \eta_2 \cdot E_m(\rho_m(\beta_m), \alpha_m(\beta_m))\} \ \forall m \quad (6.45)$$

It is possible to notice that in this case we suppose no mutual influence among different VUs.

6.2.2.3 Genetic Algorithm

We propose a GA-based solution for solving both cluster-based and distributed approaches. GAs are evolutionary search methods inspired by the theory of natural selection and genetics. GA process begins with an initial population space ($\mathcal{PS}$) that constitutes the set of possible solutions (i.e., individuals), each having a chromosome ($\mathcal{C}$). Through an iterative process, involving the creation of a new $\mathcal{PS}$ with possibly better individuals at each step, the sub-optimal solution is obtained. The evaluation process includes the analysis of each $\mathcal{C}$ of the current $\mathcal{PS}$ through a fitness function ($\mathcal{FF}$), the selection of a parent $\mathcal{C}$ is based on a selection function

(S_f), then the formation of new individuals by using mutation and crossover GA operators. In the mutation process, a new $\mathcal{C}$ is formed by altering some of the genes in the selected solution from $(\mathcal{PS})$, while, in the crossover process, two chromosome sets with good fitness function constitute a $\mathcal{C}$ for the next generation by combining their genes. Each evaluation creates a better solution set and finally ends by providing a solution point with a higher fitness value. More comprehensive information on GA and evolutionary algorithms can be found in [107], while here we focus on the main elements for the sake of brevity.

Chromosome In this work, we have considered $\mathcal{C}$ constituted by set of $\beta_m \in \mathcal{B}_n$ values for the nth RSU. Thus, each $\beta_m \in [0, 1]$ acts as a gene for $\mathcal{C}$.

Fitness Function The $\mathcal{FF}$ allows to model the problem to be minimized considering also the constraints; hence, it is defined by using the objective function in (6.44), later written as $f(\mathcal{B}_n)$ for the nth RSU, plus three additional penalty functions related to the constraints in (6.32), (6.33) and (6.35). The fitness function $\mathcal{FF}(\mathcal{B}_n)$ is:

$$\mathcal{FF}(\mathcal{B}_n) = f(\mathcal{B}_n) + \Upsilon_1 \cdot \max(0, C1(\mathcal{B}_n)) + \Upsilon_2 \cdot \max(0, C2(\mathcal{B}_n)) + \Upsilon_3 \cdot \max(0, C3(\mathcal{B}_n)) \tag{6.46}$$

where Υ_1, Υ_2 and Υ_3 are the weighting coefficients for the penalty values, and:

$$C1(\mathcal{B}_n) = \sum_{M_n} \left(T_m(\rho_m(\beta_m), \alpha_m(\beta_m)) - \bar{T}_{x_m}\right)$$

$$C2(\mathcal{B}_n) = \sum_{M_n} \left(E_m(\rho_m(\beta_m), \alpha_m(\beta_m)) - E_m^{x_m,\mathrm{c}}\right)$$

$$C3(\mathcal{B}_n) = \sum_{M_n} \left(T_m^{\mathrm{FL}}(\rho_m(\beta_m)) + T_{m,n}^{\mathrm{off}}(\alpha_m(\beta_m)) - T_{m,n}^{\mathrm{soj}}\right)$$

where $C1(\mathcal{B}_n)$ is the additional fitness penalty for VUs not performing FL and offloading process within the service latency requirement, $C2(\mathcal{B}_n)$ is the penalty for not respecting the energy constraint defined in (6.33) and $C3(\mathcal{B}_n)$ is the supplementary penalty for VUs not performing the offloading process before moving out of RSU coverage.

Selection The selection function S_f used for the parent selection is based on the roulette wheel selection technique, where the selection probability for an individual to be selected depends upon its fitness score. It should be noted that since our problem is latency and energy minimization, parents with the lowest fitness are selected at each round for reproduction stage.

Crossover In the crossover operator, new chromosomes $(C_1^{\text{new}}, C_2^{\text{new}})$ are generated by alternating genes of the parents $(C_1^{\text{old}}, C_2^{\text{old}})$ from a crossover point. Thus, child chromosomes can be written as

$$C_1^{\text{new}} = \Phi C_1^{\text{old}} + (1-\Phi)C_2^{\text{old}}, \quad C_2^{\text{new}} = \Phi C_2^{\text{old}} + (1-\Phi)C_1^{\text{old}}$$

where Φ is the crossover point uniformly distributed in $[\Lambda, (1+\Lambda)]$, i.e., $\Phi \sim U(-\Lambda, 1+\Lambda)$

Mutation We have used a Gaussian mutation technique where selected genes (β_m) from a child C can be altered by adding a random value from a Gaussian distribution, i.e., $\beta_m \longrightarrow \beta_m + \nu$, where, ν is a random variable with a Gaussian distribution, i.e., $\nu \sim \mathcal{N}(\mu, \sigma^2)$.

Algorithm 13 The proposed GA-based Approach

Input: $\mathcal{FF}, G_{\max}, M_n, \Phi, \nu$

Output: $\mathcal{B}_n^*$

```
1: Generate the initial population space $\mathcal{PS}$ with each $\beta_m \in [0,1]$
2: while $i \leq G_{\max}$ do
3:     function EVALUATE($\mathcal{PS}$)
4:         Find $\mathcal{FF}(C)$, $\forall C \in \mathcal{PS}$.
5:     end function
6:     function SEARCH($\mathcal{PS}$)
7:         Select better fit individuals using $S_f$
8:     end function
9:     function CREATE($\mathcal{PS}$)
10:        Generate new $C$s through Crossover and Mutation (using $\Phi, \nu$).
11:        Integrate $C$s with current $\mathcal{PS}$ and sort them using fitness scores i.e., $\mathcal{FF}(C)$
12:    end function
13:    Replace current $\mathcal{PS}$ with new best set of $C$s.
14:    $i = i + 1$
15: end while
16: return $\mathcal{B}_n^*$
```

Algorithm 13 shows the steps used during the implementation of GA for the clustered approach. The main GA steps include the evaluation of $\mathcal{PS}$ (Line 3-5), selection of better fit individuals as parent Cs (Line 6-8), generation of new possibly better fit Cs for the next generation (Line 9-12). The algorithm terminates after a maximum number of iterations $G_{\max}$ are reached. A similar process can be used for the distributed case by considering the individual VUs, where GA performs optimization for each $m \in M_n$ separately. It is worth to be noticed that, when GA is applied to the clustered approaches a set of VUs participates in the GA process. GA process can produce, for some of these VUs, a solution with $\beta_m = 0$, corresponding to exclude such VUs from the FL process. Thus, inherently, GA process is also able to optimize the clusters' size by including/excluding VUs from the FL process, if there is an advantage in terms of cost.

6.2.2.4 Limited Search-based Heuristic Approach (LS-HuA)

In order to compare the results with a simpler while sub-optimal solution, we propose also an intuitive heuristic approach where we consider a reduced-size solution space $\hat{\mathcal{SP}}$ through a user-defined parameter θ_{hu} representing the number of possible values taken by the parameter β_m. In this way we are going to optimize the problem while considering only a subset of possible solutions. For example, in case of $\theta_{hu} = 5$, $\beta_m \in \{0, 0.2, 0.4, 0.6, 0.8\}$. We do not consider the case with $\beta_m = 1$, since it corresponds to completely assign the time interval to the FL process, resulting in an always infeasible solution for active VUs having tasks to be offload. The smaller values of θ_{hu} reduces the simulation time, while limiting the accuracy of a solution provided. On the other hand, larger values of θ_{hu} allow the user to search over the larger $\mathcal{SP}$ for finding an optimal solution (i.e., exhaustive search). Also in this case $\beta_m = 0$ corresponds to exclude the mth VU from the FL process.

Algorithm 14 lists the steps followed during the search process. It includes the creation of a reduced search space (Line 1), initializing the cost function value (f_{hu}) that stores the optimal cost for each iteration (Line 2), and iterating over all possible solution points ($\mathcal{B}_n$) from $\hat{\mathcal{SP}}$ for finding the best possible solution (Line 3-12). In the end, the algorithm returns the best possible solution point $\mathcal{B}_n^{hu}$ found through iterations. In case there is no feasible solution available, VU decides to offload without performing any FL iteration.

Algorithm 14 Limited Search-based Heuristic Approach

Input: M_n, θ_{hu}

Output: $\mathcal{B}_n^{hu}$

```
 1: Create $\hat{\mathcal{SP}} = \{\beta_n\}$ of size $\theta_{hu}^{(M_n)}$ with all possible solution points to be searched in the reduced-size solution space
 2: Initialize $f_{hu} = \infty$,
 3: for all $\mathcal{B}_n \in \hat{\mathcal{SP}}$ do
 4:     Use (6.44) for finding total cost $f(\mathcal{B}_n)$
 5:     Determine all constraint functions values
 6:     if $f(\mathcal{B}_n) \leq f_{hu}$ and all constraints are satisfied then
 7:         $f_{hu} = f(\mathcal{B}_n)$ and $\mathcal{B}_n^{hu} = \mathcal{B}_n$
 8:     end if
 9: end for
10: if $f_{hu} = \infty$ (i.e., no feasible solution found) then
11:     $\mathcal{B}_n^{hu} = \{0\}_{1\times M_n}$
12: end if
13: return $\mathcal{B}_n^{hu}$
```

6.2.2.5 Optimal Offloading Parameter

Here we aim at finding a closed form expression for the optimal offloading parameter $\alpha_m^{\text{opt}}(\beta_m)$ having set β_m. It should be noticed that this particular analysis is carried out by considering that all system parameters are known in advance, which is not the case in reality given the uncertainty of the environment. Thus the results are used for comparison.

In case we fix β_m it is possible to obtain the optimal offloading parameter $\alpha_m^{\text{opt}}(\beta_m)$ by resorting to the equality conditions in (6.18), (6.33) and (6.35). Resorting to (6.18), the optimal offloading parameter $(\alpha_m^{\text{T}_1})$ implies that:

$$T_{m,n}^{\text{off}}(\alpha_m^{\text{T}_1}) = T_m^{\text{loc}}(\alpha_m^{\text{T}_1}) \tag{6.47}$$

Exploiting (6.11) and (6.16), we have the following:

$$T^{\mathrm{off}}_{m,n}(\alpha^{\mathrm{T}_1}_m) = \alpha^{\mathrm{T}_1}_m \cdot \hat{T}^{\mathrm{off}}_{m,n}, \quad T^{\mathrm{loc}}_m(\alpha^{\mathrm{T}_1}_m) = (1-\alpha^{\mathrm{T}_1}_m) \cdot T^{x_m,\mathrm{c}}_m$$

Hence, exploiting (6.47) we have,

$$\alpha^{\mathrm{T}_1}_m = \frac{T^{x_m,\mathrm{c}}_m}{T^{x_m,\mathrm{c}}_m + \hat{T}^{\mathrm{off}}_{m,n}} \tag{6.48}$$

In addition, the equality condition in (6.35) allows to achieve an optimal offloading parameter $\alpha^{\mathrm{T}_2}_m(\beta_m)$ so that,

$$T^{\mathrm{FL}}_m(\rho_m(\beta_m)) + T^{\mathrm{off}}_{m,n}(\alpha^{\mathrm{T}_2}_m(\beta_m)) = T^{\mathrm{soj}}_{m,n}$$

where,

$$T^{\mathrm{off}}_{m,n}(\alpha^{\mathrm{T}_2}_m(\beta_m)) = \alpha^{\mathrm{T}_2}_m(\beta_m) \cdot \hat{T}^{\mathrm{off}}_{m,n}$$

which returns,

$$\alpha^{\mathrm{T}_2}_m(\beta_m) = \frac{T^{\mathrm{soj}}_{m,n} - T^{\mathrm{FL}}_m(\rho_m(\beta_m))}{\hat{T}^{\mathrm{off}}_{m,n}} \tag{6.49}$$

In case the mth VU performs the FL process for a longer time and goes out of the coverage of RSU, i.e., $T^{\mathrm{soj}}_{m,n} < T^{\mathrm{FL}}_m(\rho_m(\beta_m))$, it will not be able to offload any data towards the RSU, i.e., $\alpha^{\mathrm{T}_2}_m(\beta_m) = 0$. Hence, (6.49) can be rewritten as,

$$\alpha^{\mathrm{T}_2}_m(\beta_m) = \max\left\{0, \frac{T^{\mathrm{soj}}_{m,n} - T^{\mathrm{FL}}_m(\rho_m(\beta_m))}{\hat{T}^{\mathrm{off}}_{m,n}}\right\} \tag{6.50}$$

Following the energy constraint defined in (6.33) equality holds for a particular optimal offloading parameter $\alpha^{\mathrm{E}_1}_m(\beta_m)$ and can be written as,

$$E^{\mathrm{FL}}_m(\rho_m(\beta_m)) + E^{x_m}_m(\alpha^{\mathrm{E}_1}_m(\beta_m)) = E^{x_m,\mathrm{c}}_m$$

Exploiting (6.12) and (6.19), we have the following:

$$E_m^{x_m}(\alpha_m^{\mathrm{E}_1}(\beta_m)) = \alpha_m^{\mathrm{E}_1}(\beta_m) \cdot \hat{E}_{m,n}^{\mathrm{off}} + (1 - \alpha_m^{\mathrm{E}_1}(\beta_m)) \cdot E_m^{x_m,\mathrm{c}}$$

that returns,

$$\alpha_m^{\mathrm{E}_1}(\beta_m) = \frac{-E_m^{\mathrm{FL}}(\rho_m(\beta_m))}{\hat{E}_{m,n}^{\mathrm{off}} - E_m^{x_m,\mathrm{c}}} \tag{6.51}$$

In such cases where $\hat{E}_{m,n}^{\mathrm{off}} > E_m^{x_m,\mathrm{c}}$, performing computation offloading is not an option since it requires additional energy and that results into $\alpha_m^{\mathrm{E}_1}(\beta_m) = 0$. Therefore, (6.51) can be modified to,

$$\alpha_m^{\mathrm{E}_1}(\beta_m) = \max\left\{0, \frac{-E_m^{\mathrm{FL}}(\beta_m)}{\hat{E}_{m,n}^{\mathrm{off}} - E_m^{x_m,\mathrm{c}}}\right\} \tag{6.52}$$

In the end, (6.48), (6.50), and (6.52) are considered for finding $\alpha_m^{\mathrm{opt}}(\beta_m)$ as:

$$\alpha_m^{\mathrm{opt}}(\beta_m) = \min\left\{\alpha_m^{\mathrm{T}_1}, \alpha_m^{\mathrm{T}_2}(\beta_m), \alpha_m^{\mathrm{E}_1}(\beta_m)\right\} \tag{6.53}$$

This procedure is used as a reference value in the following for testing the effectiveness of the estimated offloading parameters that depends on the FL and, in turns, on its iterations.

6.2.3 Numerical Results

Numerical results are obtained through computer simulations with Matlab. A variable number of VUs between 100 and 1000 are considered, assuming that each one is generating tasks with a probability equal to 0.2, while the remaining have no task to be offloaded. VUs are uniformly distributed in a two-lane road and travel in

Table 6.1: Simulation Parameters

Simulation parameters	
HAP Beam Coverage (R_{HAP})	2 Km
RSU Coverage ($R_{r,n}$)	25 m
Task Size (D_{x_m})	2.5 MB
Task Computation (Ω_{x_m})	$10^3 \times D_{x_m}$ FLOPS
Task Results ($D_{x_m,\text{rx}}$)	0.5 MB
VU Flops ($c_{v,n} \cdot f_{v,n}$)	8 GFLOPS
VU Tx. Energy (P_m^{tx})	1.3 W [121]
VU Rx. Energy (P_m^{rx})	1.1 W [121]
VU Comp. Energy (P_m^{c})	0.9 W [121]
RSU Flops ($c_{r,n} \cdot f_{r,n}$)	80 GFLOPS
HAP Beam Bandwidth (B_{HAP})	100 MHz
RSU Bandwidth ($B_{r,n}$), $\forall n$	10 MHz
HAP Altitude (h_{HAP})	20 Km [128]

either directions with a velocity $\bar{v}_m$ equal to 10 m/s. Moreover, 80 RSUs are randomly placed on either sides of the lanes. The task latency requirement ($\bar{T}_{x_m}$) has been set to 2 s; this value is consistent with other works in the literature [118, 119] considering similar scenarios and applications. The other parameters considered in simulation are listed in Table 6.1.

The GA weight coefficients are $\Upsilon_1, \Upsilon_3 = 10$, $\Upsilon_2 = 1$, while the crossover function parameter is $\Lambda = 0.1$, and the mutation function parameters are $\mu = 0.02$, $\sigma = 0.1$. Moreover, we set an initial population of 30 chromosomes and $G_{\max}$=50, α_m^0 is uniformly distributed between 0 and 1, while $\mathcal{K}$ is 2000, and both $|w_m^i|$ and $|w_G|$ have size 1000 bits. The parameter γ is set to 0.4 while estimating offloading parameters. In DBC policy $\hat{\Pi}$ is equal to $R_{r,n}/2$, while $p = 0.5$ is used in PC. Also, the numerical value used for both η_1 and η_2 is 0.5. Finally, we have considered $\theta_{hu} = 6$ when evaluating the results for the LS-HuA.

In the following, we present the results by comparing the proposed GA approach with LS-HuA and two static benchmarks:

- *Computation Offloading without Performing any FL Iterations (Without FL)*: In this approach, each VU decides to offload data without performing any FL iteration. Therefore, the offloading operation is performed with α_m^0 without adding any FL cost. Since the initial value of offloading parameter may or may not be optimal, this approach cannot guaranty the optimal performance. Though this approach can have a reduced cost, VU performs the offloading operation without taking into account the available time and energy resources which may diminish performance in terms of constraint failures.

- *Computation Offloading by Performing Complete FL Iterations (Complete FL)*: In this particular method, each VU performs the ρ^{opt} FL iterations before offloading data towards RSU. Thus the offloading operation is performed with α_m^{opt}, as defined in (6.53), when β_m is such that $\rho_m = \rho^{\text{opt}}$. Though VUs can perform offloading with optimal offloading parameters, it is not always feasible to perform ρ^{opt} FL iterations with limited service time, sojourn time, and energy of VU, which limits the performance of this approach.

These two benchmarks do not consider the available resources of VUs while making computation offloading decisions and may have a sub-optimal performance over long-term simulations. In the following figures, GA-FC, GA-DBC, GA-PC, and GA-D are the acronyms used for the Genetic Algorithm technique with FC, DBC, PC, and Distributed Clustering approaches, respectively.

Avg. Latency and Energy Cost with Varying VUs

In Fig. 6.7, the average cost in terms of joint latency and energy consumed for both FL and task processing phases using a variable number of VUs is shown. The results show that GA and LS-HuA techniques have a considerable advantage over the *Complete FL* approach with reduced cost values. Even though the *Without FL* approach has the minimum cost among all the proposed methods, it cannot guarantee a reliable performance in terms of service latency, sojourn time, and energy constraints, as shown and discussed later in Figs. 6.8-6.10. The proposed Clustered GA approaches (i.e., GA-FC, GA-DBC, GA-PC), thanks to a better knowledge of the surrounding environment, performs FL and task processing with a lower cost along with better reliability, which can benefit several latency-critical services demanded by VUs having limited energy resources. Since the required FL iterations to achieve model convergence reduces with the participation of more VUs, the cost of the *Complete FL* process decreases with increasing VUs, but still it fails to achieve the overall performance of proposed GA methods.

Performance in Terms of Sojourn Time Failures

Fig. 6.8 shows the percentage of number of VUs failing to perform the offloading operation before leaving the RSU coverage. According to constraint (6.35), each VU should complete both FL and task offloading processes within available sojourn time. The two benchmark methods lack suitable flexibility while performing the offloading operations as both methods do not utilize the available latency resources properly while performing the offloading operations. That results in higher failures since they are not able to perform both FL and task offloading operations in a limited sojourn time. It should be noted that the complete FL approach has a falling curve, which

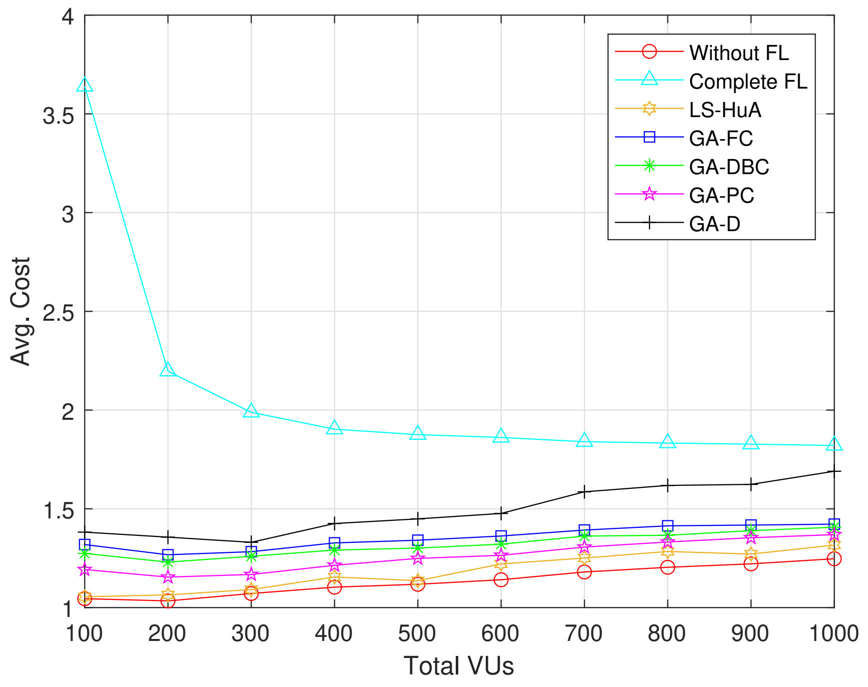


FIGURE 6.7: Cost Function

is due to the fact that by the increase in the number of VUs in a service area, a shorter time will be required to achieve FL convergence. On the other hand, both GA schemes and LS-HuA approach perform an adequate number of FL iterations, before performing the offloading operations, and, as a result, have very few failures with reduced cost. The performance of the *Without FL* worsens with an increasing number of VUs, and at a certain point, it has even higher failures than the *Complete FL* approach.

Performance in Terms of Service Time Outage

Fig. 6.9 shows the percentage of VUs failing to perform both FL and the task processing operation within a demanded service latency. The significant performance improvement in terms of a reduced number of failures can be observed in the GA and LS-HuA results, comparing with the benchmark methods. This is mainly because of the improper allocation of VUs available resources towards FL and task processing

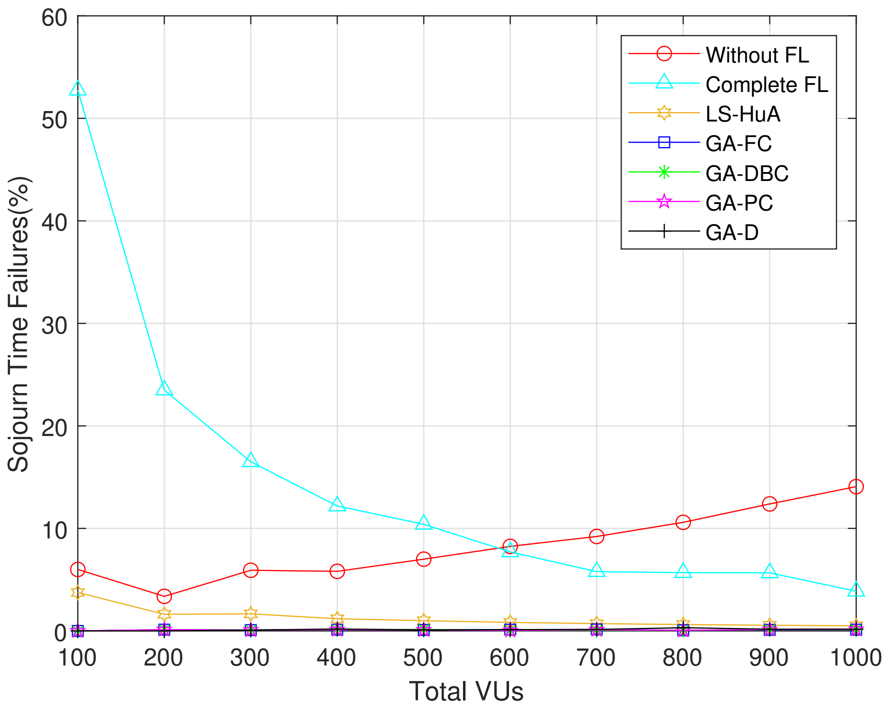


FIGURE 6.8: Percentage of VUs with sojourn time constraint violation

phases in the benchmark methods. These results also highlight the importance of proper allocation of VU resources for the FL and task processing phases (estimation of $\mathcal{B}$), for improving the overall VNs performance.

Performance in Terms of Energy

Fig. 6.10 shows the percentage of VUs violating the energy constraint in (6.33). Since each FL iteration costs energy, performing ρ^{opt} iterations for each VU before offloading during the *Complete FL* approach decreases its reliability in terms of respecting VUs energy constraint and can be seen from these results. On the other hand, GA and LS-HuA approaches have better performance since they allocate a proper number of FL iterations before offloading. Thus these results highlight the importance of performing an adequate number of FL by taking into account the

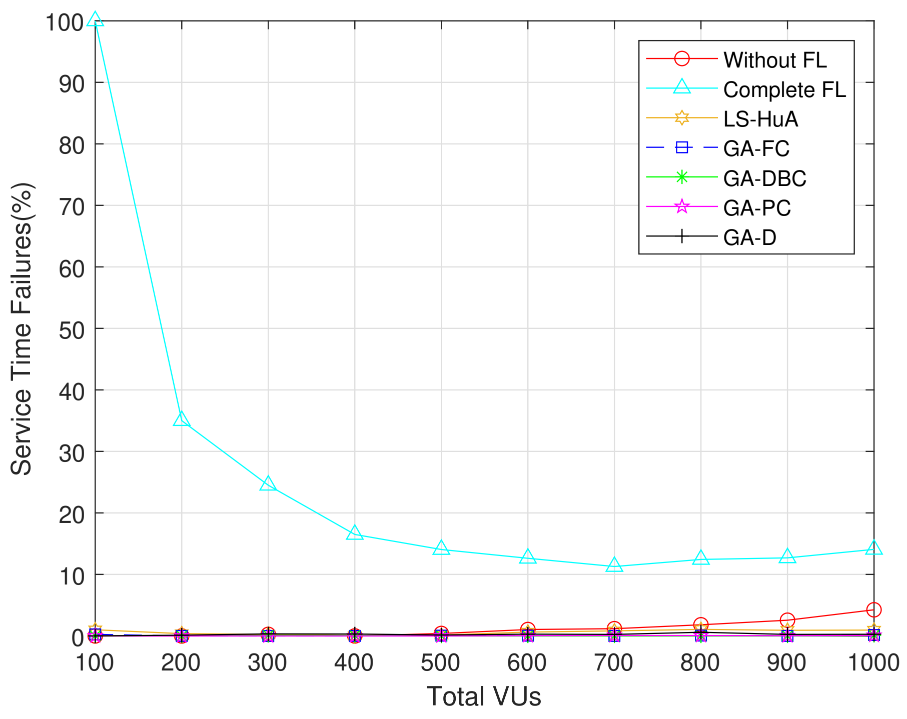


FIGURE 6.9: Percentage of VUs with service time constraint violation

VUs available resources to achieve a reliable performance with FL in dynamic VNs.

Offloading Performance

Fig. 6.11 shows the average error when estimating the offloading parameters. For a given set M of VUs, the error in the estimation process is measured by using the Root Mean Square Error (RMSE) as,

$$\mathcal{E}(M, \mathcal{B}^*) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left|\left(\alpha_m^{\text{opt}}(\beta_m^*)\right)^2 - (\alpha_m(\beta_m^*))^2\right|}$$

where $\alpha_m^{\text{opt}}(\beta_m^*)$ is the offloading parameter estimated in (6.53), while $\alpha_m(\beta_m^*)$ is derived through (6.38). The value $\mathcal{E}$ decreases for the GA-FC approach with higher

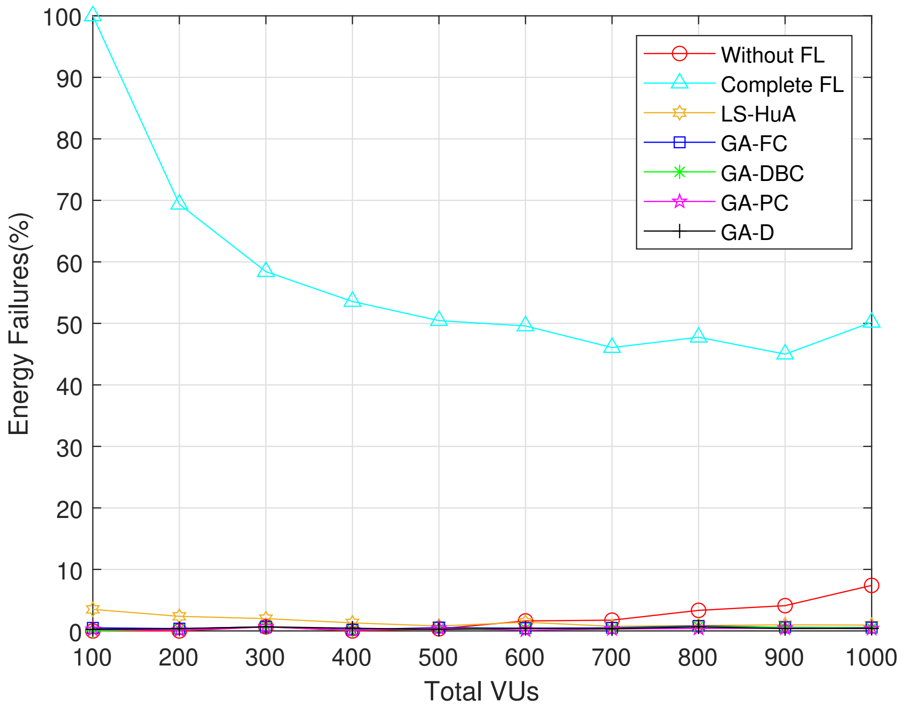


FIGURE 6.10: Percentage of VUs violating the Energy Constraint in (6.33)

values of M, as the number of surrounding VUs increases. Other clustered approaches have reduced offloading performance since only a lower number of VUs participate in the optimization process before performing offloading. Also, with limited available information, the distributed approach fails to adapt itself properly.

Impact of GA Iterations

In the case of GA, the performance can be improved by increasing the number of iterations of the GA. In Fig. 6.12, we compare the performance in terms of average latency and energy cost by considering a different number of GA iterations in the GA-FC policy. It can be seen that as the number of iterations increases, GA performance improves. However, after a certain number of iterations (i.e., 50), performance of the GA process becomes stable, thus, performing a higher number of iterations can

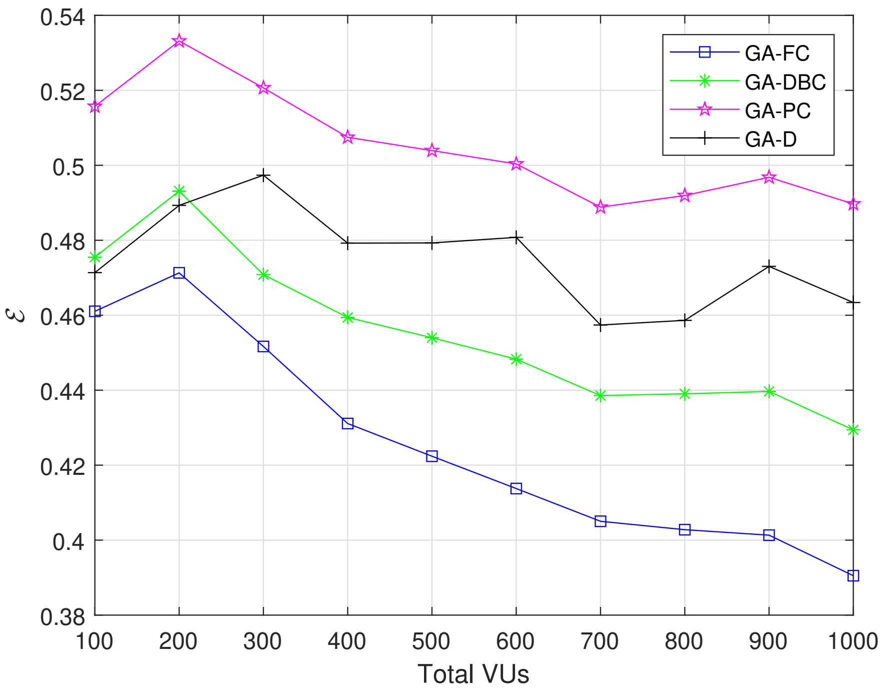


FIGURE 6.11: Avg. Offloading Error

only increase the time complexity of the GA process by several folds, without major gains in terms of offloading solution.

6.2.4 Conclusion

In this work, we performed the optimization for a joint FL and task processing problem over the integrated air-ground network of HAP-assisted VN. For this, we first modeled the computation offloading problem in the vehicular scenario in which each VU can offload a portion of their tasks to the surrounding RSUs. Next, an integrated air-ground network-based FL platform was introduced, where powerful HAPs act as an FL server to assist several VUs (i.e., FL clients) in estimating the better offloading parameters. A joint computation offloading and FL process optimization problem aiming at minimization of overall latency and energy cost was formulated.

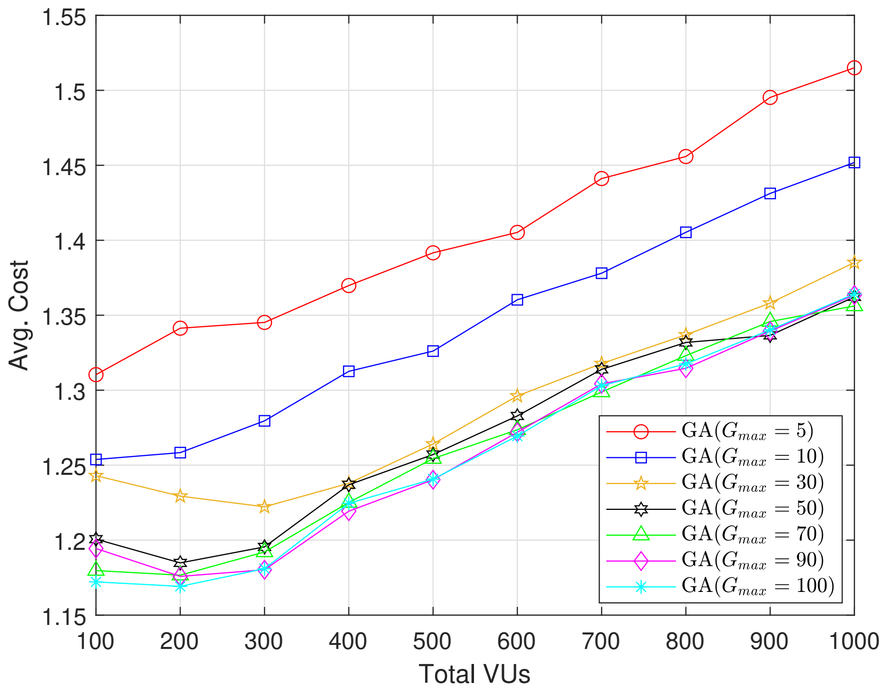


FIGURE 6.12: GA Performance Vs Iterations

The proposed solution methods include the RSU cluster-based approach with several clustering policies and distributed approaches. An evolutionary search-based GA was proposed to find both allocated time for the two phases and estimating the offloaded portions for the VUs. Simulation results demonstrate that our proposed GA-based approaches, when compared with other benchmark solutions, show a network-wide performance improvement.

As future directions of this work, we point out the extension to autonomous driving scenarios, where VUs data can be analyzed for solving vehicular problems through the proposed FL platform. Some other challenges to be faced include (i) a proper RSU selection for offloading, (ii) the possibility of considering a network of multiple decentralized HAPs for a higher fault-tolerance, (iii) the optimization of the number of VUs participating in the FL process considering their available resources, (iv) the extension to intermediate FL layers (e.g., LAPs, UAVs, RSUs) for reducing the communication/computation costs during FL data processing and communication (i.e., Hierarchical FL).

6.3 Distributed FL over Joint Air-ground Networks for Vehicular Applications

In this work, we propose a distributed FL process for vehicular applications. We consider a joint T-NTN composed of VUs, RSUs, UAVs/LAPs, and HAPs. A distributed FL platform allowing a distributed and flexible FedAvg process is proposed exploiting different layers, where RSUs, UAVs, and HAPs are capable of collecting the FL model updates from VUs in each FL iteration for generating the global update vectors/models for the next iterations.

With the possibility of exploiting multiple layers, a proper selection of the nodes where the FL process should be performed is required, aiming at enhancing latency, energy, and FL process performance. However, solving such a network selection problem over a multi-layered dynamic VN can be enormously complex, and traditional optimization techniques are inefficient. In the recent past, various works have highlighted the importance of RL-based approaches for solving the network selection problem over VNs [129]. In this work, we resort to the RL method for solving the FL network selection problem. As a first step, we model the problem as a sequential decision-making process through a Markov Decision Process (MDP) framework which requires a proper design of a state space, action space, reward function, and environment dynamics [130]. The environment dynamics are modeled through a set of time-dependent state transition probability expressions, considering the VUs local environment for designing the MDP with better performance.

The main contributions of this work can be summarized in the following points:

- We design a communication-efficient distributed FL model over a joint air-ground network aiming at reducing FL process latency and energy costs.
- A constrained optimization problem is formulated for minimizing the overall cost (in terms of joint latency, energy and the FL training performance) of the process by a proper assignment of VUs and FL servers.
- A MDP framework is considered for modeling the problem as a sequential decision-making process, and a value iteration technique is used to solve it. The MDP environment is modeled through time-dependent state transition probabilities that take into account the local vehicular environment.
- The performance of the proposed scheme is analyzed by comparing it with different heuristic techniques and conclusions are drawn.

6.3.1 System Model and Problem Formulation

In the following, an urban Internet of Vehicles (IoV) scenario for Intelligent Transportation Systems with connected and intelligent VUs is considered, allowing to request several intelligent services from the nearby edge computing facilities. In recent times, such urban IoV scenarios have gained a lot of attention from the vehicular research community [74, 78]. In particular, we consider a multi-layered joint air-ground network composed of HAPs, UAVs (i.e., LAP nodes), RSUs, deployed along the road paths, and randomly distributed VUs traveling on a road in either directions, where $\mathcal{V} = \{v_1, \dots, v_m, \dots, v_M\}$, $\mathcal{R} = \{r_1, \dots, r_n, \dots, r_N\}$, $\mathcal{U} = \{u_1, \dots, u_l, \dots, u_L\}$, correspond to the sets denoting M VUs, N RSUs and L UAVs, respectively. Each HAP node is denoted through the index h.

The system is modeled in a time-discrete manner, and the network parameters are constant in each time interval τ, where τ_i identifies the ith time interval, i.e., $\tau_i = \{\forall t | t \in [i\tau, (i+1)\tau]\}$. The generic mth VU is characterized by a processing capacity equal to $c_{v,m}$ Floating Point Operations per Second (FLOPS) per CPU cycle, while its CPU frequency is $f_{v,m}$ [76, 84]. VUs are supposed to be able to communicate on a bandwidth $B_{v,m}^{\mathrm{rsu}}$ with the RSUs, in a bandwidth $B_{v,m}^{\mathrm{LAP}}$ with UAVs and on a bandwidth $B_{v,m}^{\mathrm{HAP}}$ with the HAPs. In addition, the mth VU is supposed to hold a set $\mathcal{D}_{v_m}$ with $|\mathcal{D}_{v_m}| = \mathcal{K}_{v_m}$ data samples produced during its operation as a result of the embedded Advanced Driver-Assistance System (ADAS), and later used during the FL training process. FL is here exploited for assisting during vehicle operations, e.g., computation offloading, path planning, object detection.

The nth RSU, supposed to be in a fixed position with a coverage radius $R_{r,n}$, is characterized by a processing capacity equal to $c_{r,n}$ FLOPS per CPU cycle, with CPU frequency $f_{r,n}$, and communication capabilities, supposed to be identified through a communication technology, able to cover the VUs on ground with an overall bandwidth $B_{r,n}^{r\rightarrow v}$. The RSUs are also able to connect with UAVs and the HAPs with a bandwidth $B_{r,n}^{\mathrm{UAV}}$ and $B_{r,n}^{\mathrm{HAP}}$, respectively. The RSUs are connected to the electrical grid for the energy supply. Each RSU can provide edge computing services to the VUs in its coverage space.

In addition, the area is supposed to be under the coverage of multiple UAVs with lth UAV at altitude $\bar{h}_{u,l}$ and coverage radius $R_{u,l}$. We assume that UAVs are charged by exploiting available charging points in the service area. Based on VUs requests, UAVs can move in different directions with optimal path planning, whose management is beyond the scope of this work. While serving VUs, the lth UAV, supposed to move with a relatively slow speed compared with highly mobile VUs, is characterized by a processing capability equal to $c_{u,l}$ FLOPS per CPU cycle, with CPU frequency

$f_{u,l}$. In addition, it is supposed to be able to communicate on a bandwidth $B_{u,l}^{l\to(v,r)}$ and cover an area with radius $R_{u,l}$, while the lth UAV has a bandwidth $B_{u,l}^{\mathrm{HAP}}$ when communicating with the HAP. Each UAV can serve a set of VUs and RSUs in its coverage space.

The generic hth HAP node is placed at an altitude $\bar{h}_h$ above the ground, and characterized by a processing capability c_h FLOPS per CPU cycle, with CPU frequency f_h. Moreover, we consider multi-beam antenna forming techniques, where each antenna beam is supposed to cover a geographical area of radius R_h and has a communication bandwidth $B_h^{h\to(v,r,l)}$. In the following, we will refer to a single beam as the coverage of the HAP. It should be noted that though HAP coverage is reduced to a single beam for notation simplicity, our approach can easily be scaled for the overall HAP coverage with multiple beams. Each RSU, UAV, and HAP provides edge computing services to the VUs, RSUs and UAVs within its coverage area. Fig. 6.13 shows the basic system elements and various communication links between them[4].

6.3.1.1 VU Mobility Model

We suppose that the mth VU moves in a freeway-like mobility scenario with a speed $\vec{v}_m(\tau_i)$ bounded by $\vec{v}_{\min}$ and $\vec{v}_{\max}$ [105], where the instantaneous speed is modeled through a truncated normal distribution density function:

$$f\left(\vec{v}_m(\tau_i)\right) = \begin{cases} \dfrac{2\cdot\exp\left(\frac{-(\vec{v}_m(\tau_i)-\mu)^2}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}\left(\operatorname{erf}\left(\frac{\vec{v}_{\max}-\mu}{\sigma\sqrt{2}}\right)-\operatorname{erf}\left(\frac{\vec{v}_{\min}-\mu}{\sigma\sqrt{2}}\right)\right)}, & \vec{v}_{\min}\le\vec{v}_m(\tau_i)\le\vec{v}_{\max} \\ 0, & \text{else} \end{cases} \tag{6.54}$$

[4]Despite the system model and the analysis is carried out considering the general case of multiple HAPs, the performance will be later evaluated for the simple case with only one HAP. The generic case can be seen as a simple extension of the one HAP scenario.

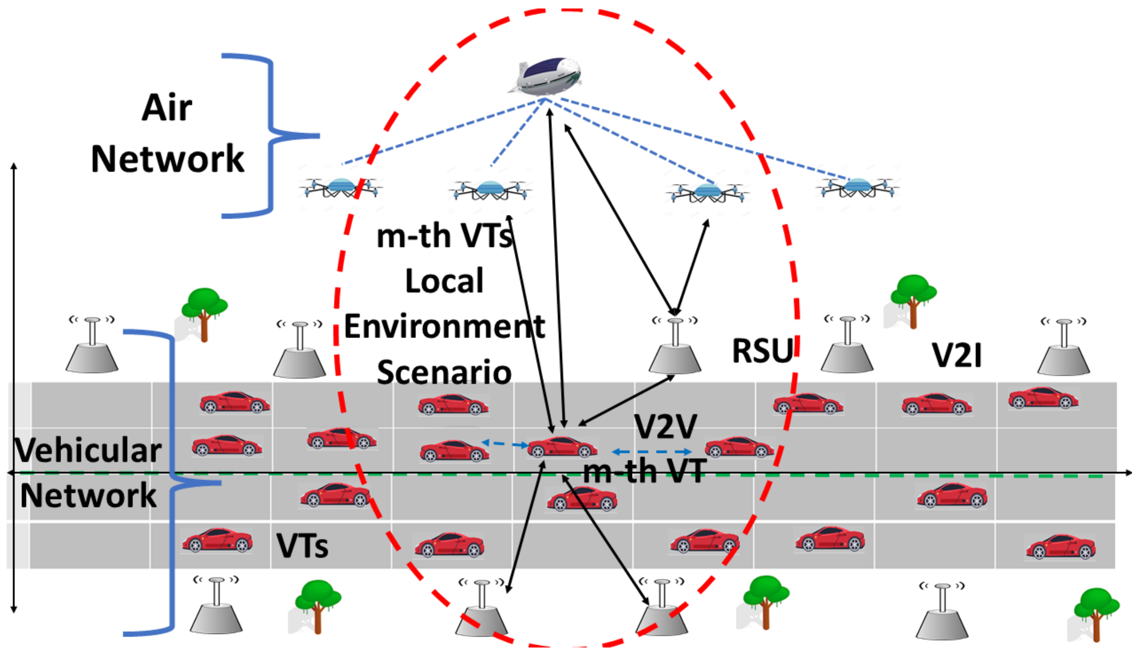


FIGURE 6.13: The T/NT Integrated Scenario.

and μ and σ are the mean and standard deviation of the vehicle's speed, and erf(x) is the Gauss error function over x. The path length within which the mth VU remains under the coverage of jth node (i.e., any of RSUs, UAVs or HAPs) is $D_{v_m,j}(\tau_i)$ and can be given by:

$$D_{v_m,j}(\tau_i) = \sqrt{d_j^2 - \left(y_j - y_{v_m}(\tau_i)\right)^2} \pm \left(x_j - x_{v_m}(\tau_i)\right) \tag{6.55}$$

where, $\left(x_{v_m}(\tau_i), y_{v_m}(\tau_i)\right)$ is the location of the mth VU at τ_i and $\left(x_j, y_j\right)$ is the projection over the ground of a generic jth edge computing node, which can be a RSU, UAV or HAP. The available sojourn time for the mth VU with respect to a generic jth node can be written as:

$$T^{soj}_{v_m,j}(\tau_i) = \frac{D_{v_m,j}(\tau_i)}{|\vec{v}_m(\tau_i)|} \qquad \forall j \tag{6.56}$$

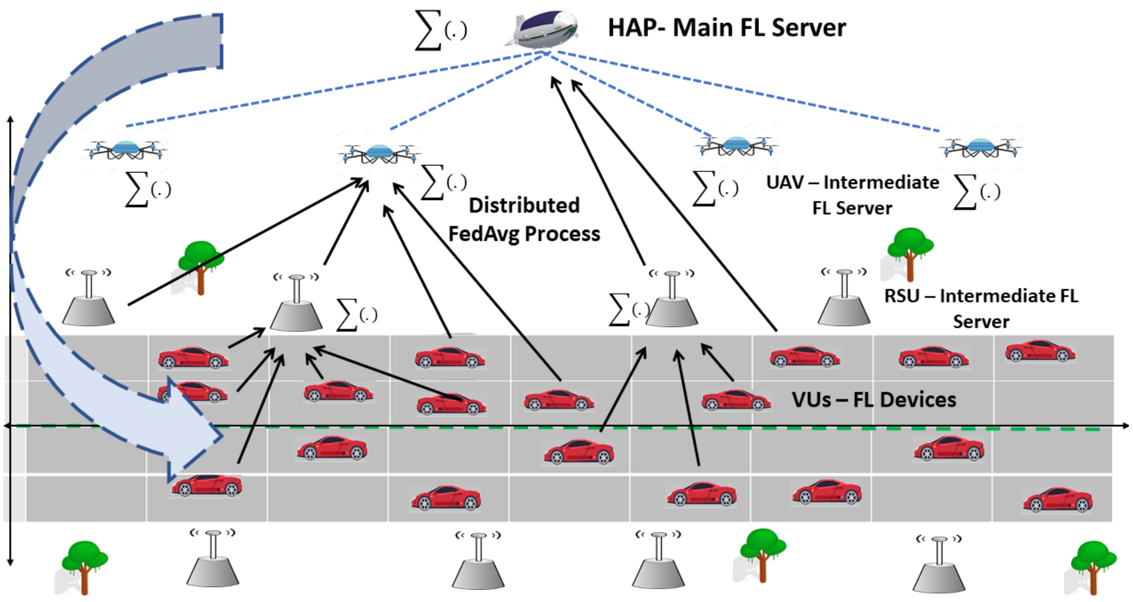

HAP- Main FL Server
$\sum(.)$
UAV – Intermediate FL Server
Distributed FedAvg Process
RSU – Intermediate FL Server
VUs – FL Devices

FIGURE 6.14: Distributed FL Platform.

6.3.1.2 Distributed FL Platform for Vehicular Applications

In order to solve the VN management through the proposed air-ground network architecture, we propose a VUs service-based distributed FL platform (Fig. 6.14).

The federated training operation depends on the service request ν, where ν can be any vehicular service requested by VUs, such as computation offloading towards edge servers, path planning, streaming-related services, etc. Each service ν requires a unique FL model $\mathcal{F}_\nu$. In the considered FL platform, VUs (i.e., FL client devices) with local datasets $\mathcal{D}_{\nu_m}$ can perform the local training for the FL model based upon the requested service ν. Since different VUs can request different services over time, a group of VUs randomly located in the coverage space of the HAP requesting the same service ν will participate collaboratively to train the FL model corresponding to the service ν. The number of VUs participating in the training process of the νth

FL model is given by:

$$M_v^{\mathrm{FL}} = \{v_m | v_m \Leftrightarrow v, v_m \in M\} \quad \text{with } M_v^{\mathrm{FL}} \subseteq M \tag{6.57}$$

In each itth FL iteration, after the local training operation, we assume that a data vector $w_{v,m}^{it,v}(\tau_i)$ (i.e., model updates embedded into IP packet) is generated where an information header is added indicating the VUs service $(\chi_v \Leftrightarrow v)$. Here, χ_v can be a unique sequence of bits indicating the vth service[5]. Such processed data will be sent towards an upper layer edge computing node based upon the network selection strategy embraced by the VUs, as will be discussed in the following.

After receiving the data from the lower layer entities, each Edge Node (EN) will perform the FedAvg process creating the new set of updates $w_{r,n}^{it,v}/w_{u,l}^{it,v}/w_h^{it,v}$, where $w_{r,n}^{it,v}$ is the aggregated FL model updates associated with the vth service generated by the intermediate RSU node n, and $w_{u,l}^{it,v}$ and $w_h^{it,v}$ are the FL model updates after the averaging process (i.e., FedAvg) performed at lth UAV node and hth HAP, respectively. With post-processing operation, the header information χ_v is again inserted into the aggregated data $(w_{r,n}^{it,v}/w_{u,l}^{it,v}/w_h^{it,v})$ for the next layer processing. In the end, data vector corresponding to the $w_{r,n}^{it,v}/w_{u,l}^{it,v}/w_h^{it,v}$ is transmitted towards the next platform or VUs based upon the network selection strategy. Though it is beyond the scope of this work, insertion/processing of the header information associated with the specific service request allows the proposed FL platform to train multiple service-based FL models simultaneously.

After receiving data from the ENs, VUs use them in the next iteration of the FL process. The process continues for several FL iterations until a certain confidence

[5]Though it is beyond the scope of this work, χ_v can easily be modeled into a more sophisticated vector that can further enhance the security of the FL platform to tackle the various security/privacy related challenges discussed in the related works section.

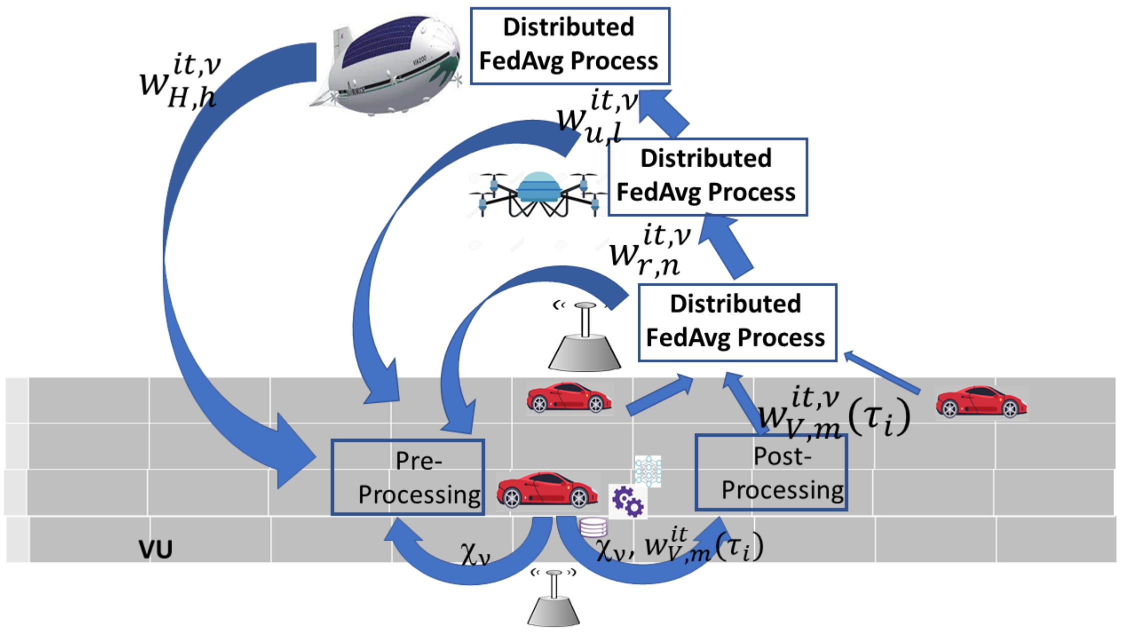


Figure 6.15: Distributed FL Process

interval is reached. Fig. 6.15 shows the steps of each single iteration of the proposed distributed FL process.

6.3.1.3 Network Selection Parameters

FL performance is a function of the number of participating VUs to the FL process, the number of FL iterations performed by VUs, the communication and computing latency, and the energy cost of each FL iteration. The FL process cost depends also on the network selection strategy adopted by different networking layers given their limited computing and communication resources.

To better clarify this point, if a VU selects the HAP node direct link for the FL data transmission, it can potentially save the processing latency and the cost required to perform the FedAvg process at the intermediate layers; however, it can increase the VUs data transmission cost in terms of transmission latency and energy, mainly due to the limited resources of VUs and the long-distance communication links between VU and HAP. Also, due to long-distance communication links, the link failure probability can be higher, resulting in a possibly high number of dropouts (i.e., VUs not participating in the FL training process). On the other hand, if VU decides to select RSU or UAV nodes for distributed FL data communication, it can potentially save communication time and energy. However, an additional burden of processing latency over these intermediate layers needs to be considered. Similar analysis can be applied to the RSU and UAV nodes when selecting the possible higher networking layers for the data communication. Therefore, there is a clear tradeoff between the different network selection strategies adapted by the VUs and the intermediate layers. A proper network selection strategy guaranteeing the optimal training latency and energy performance is required.

Based on their limited coverage ranges, each VU can be covered by set of RSUs, UAVs, and one HAP node. Focusing on the mth VU, $1 \leq N_{v,m}^{R} \leq R_{max}$, $1 \leq N_{v,m}^{U} \leq U_{max}$, and $1 \leq N_{v,m}^{H} \leq H_{max}$ represent the number of RSUs, UAVs and HAP nodes available for selection, respectively. Without loss of generality, we assume that $N_{v,m}^{R}$, $N_{v,m}^{U}$, and $N_{v,m}^{H}$ are lower bounded by 1 (i.e., every VU can be covered by at least one RSU, UAV and HAP), while R_{max}, U_{max}, and H_{max} are the upper bounds on RSUs, UAVs, and HAP nodes covering it. Here, we define the following three decision variables modeling the network selection behavior of VUs, RSUs, and UAVs.

VUs Network Selection Decision

For the case of mth VU, we define

$$\mathbf{a}_{v,m}(\tau_i) = \left[(0,1)_{(1\times N_{v,m}^{R})}, (0,1)_{(1\times N_{v,m}^{U})}, (0,1)_{(1\times N_{v,m}^{H})}\right]$$

with dimension $1\times(N^R_{v,m}+N^U_{v,m}+N^H_{v,m})$ modeling the available nodes for selection. VU can either select RSU, UAV, or HAP for communicating the FL model parameter updates. If $\mathbf{a}_{v,m}(\tau_i) = \{0\}_{(1\times(N^R_{v,m}+N^U_{v,m}+N^H_{v,m}))}$, the mth VU does not participate in the FL process. Also, for avoiding the additional complexity, we consider that each VU can be assigned to only one EN which can be RSU, UAV, or HAP during the FL process. Thus,

$$\sum \mathbf{a}_{v,m}(\tau_i) \leq 1 \tag{6.58}$$

RSUs Network Selection Decision For the case of nth RSU, we define

$$\mathbf{b}_{r,n}(\tau_i) = \left[(0,1)_{(1\times N^U_{r,n})}, (0,1)_{(1\times N^H_{r,n})}\right]$$

with dimension $1\times(N^U_{r,n}+N^H_{r,n})$ modeling the available nodes for selection. RSU node can either select UAV, or HAP for communicating the FL model parameter updates. If $\mathbf{b}_{r,n}(\tau_i) = \{0\}_{(1\times(N^U_{r,n}+N^H_{r,n}))}$, the nth RSU node does not communicate with higher layers and broadcasts back the model parameters towards VU. For avoiding the additional complexity we consider that each RSU can be assigned to only one EN which can be UAV, or HAP during the FL process. Thus,

$$\sum \mathbf{b}_{r,n}(\tau_i) \leq 1 \tag{6.59}$$

UAVs network selection decision For the case of lth UAV, we define

$$\mathbf{c}_{u,l}(\tau_i) = \left[(0,1)_{\left(1\times N^H_{u,l}\right)}\right]$$

with dimension $1\times N^H_{u,l}$ modeling the available nodes for selection. UAV can select HAP for communicating the FL model parameter updates or broadcast back the results towards VUs. If $\mathbf{c}_{u,l}(\tau_i) = \{0\}_{(1\times N^H_{u,l})}$, the lth UAV broadcast back the model parameters towards VU. For avoiding the additional complexity we consider that each UAV can be assigned to only one EN during the FL process. Thus,

$$\sum \mathbf{c}_{u,l}(\tau_i) \leq 1 \tag{6.60}$$

6.3.1.4 FL Process Cost Analysis

In general, FL is an iterative learning process where each FL iteration includes several steps adding latency and energy costs. Local on-device ML model training, data communication between VUs and FL servers, pre- and post-processing of FL model data, FedAvg process performed at FL servers are the main steps involved during FL iteration. In the following, we analyze the latency and energy cost of each of these operations.

FL Local training Model The FL computation corresponds to the local training of the ML model based on the on-device dataset. In local device training, the mth VU with service request v has to compute the local parameter set $w^{it,v}_{v,m}$ through the dataset having size $\mathcal{K}_{v_m}$ data samples; if we assume that, for every iteration, the total number of FLOPs required for each data sample d is ψ_d, the time and energy consumed during the FL training process by the mth device is [84]:

$$T^{\mathrm{FL,c}}_{v_m} = \frac{\sum_{d=1}^{\mathcal{K}_{v_m}} \psi_d}{c_{v,m} f_{v,m}}, \quad E^{\mathrm{FL,c}}_{v_m} = P^{\mathrm{c}}_{v,m} \cdot T^{\mathrm{FL,c}}_{v_m}.$$

where $P^{\mathrm{c}}_{v,m}$ is the power consumed by the mth VU for the data processing. We suppose for simplicity that the on-device FL processing time and energy is the same for every iteration

FL Data Pre-/Post-processing For each FL iteration, pre- and post-processing operations are performed for detecting and adding the header information χ_ν associated with the service (ν) requested by the vehicular nodes. The latency and energy of these operations are:

$$T_i^{\text{FL,hp}} = T_i^{\text{FL,pre}} + T_i^{\text{FL,post}}, \quad E_i^{\text{FL,hp}} = P_i^{\text{c}} \cdot T_i^{\text{FL,hp}}.$$

where, $T_i^{\text{FL,pre}} = \frac{N_i \cdot \psi_{pre}}{c_i \cdot f_i}$ is the time required to detect and remove the header information from FL data at the ith node, function of its computation resources, the number of FLOPs required to process the FL data (i.e., model parameters embedded in the IP packets) from node i given as ψ_{pre} and a number of VUs/servers sending the updates towards the server i given by N_i. Also, $T_i^{\text{FL,post}} = \frac{\cdot \psi_{post}}{c_i \cdot f_i}$ is the post-processing operation time required to insert the header information on model update vectors with ψ_{post} being a number of FLOPs required to process the updated model data after FedAvg.

FL FedAvg Process In the proposed FL infrastructure, intermediate FL server (i.e., RSUs, UAVs, HAP) perform the FedAvg process on the data received from any of the lower layers. The latency and the energy required to perform the FedAvg process is given by:

$$T_i^{\text{FL,FA}} = \frac{N_i \cdot \psi_{FA}}{c_i \cdot f_i}, \quad E_i^{\text{FL,FA}} = P_i^{\text{c}} \cdot T_i^{\text{FL,FA}}$$

where ψ_{FA} is the number of FLOPs required to process the individual nodes parameter vectors over ith server

FL Data Communication Model The data rate between ith and jth node is a function of the mutual distance, hence:

$$r^{\mathrm{it}}_{i,j}(B_i, d_{i,j}) = B_i \log_2\left(1 + \frac{P^{\mathrm{tx}}_i \cdot h(d_{i,j})}{N_0}\right) \tag{6.61}$$

where P^{tx}_i is the transmission power of the generic ith device, $h(d_{i,j})$ is the channel gain at a distance $d_{i,j}$ between the ith device and the jth device, and $N_0 = N_T B_i$ is the noise power, where N_T and B_i are the noise power spectral density and bandwidth associated to the ith device during communication.

During the FL processing, at each iteration it, the ith FL-device sends the parameters set w^{it}_i to the higher layers. Supposing that $|w^{it}_i|$ represents the data size of the parameters set expressed in bits [126], the uplink transmission time and energy for the FL parameters in the itth iteration is:

$$T^{\mathrm{FL,tx}}_{ij,it} = \frac{|w^{it}_i|}{r^{it}_{i,j}(B^j_i, d_{i,j})}, \quad E^{\mathrm{FL,tx}}_{ij,it} = P^{\mathrm{tx}}_i \cdot T^{\mathrm{FL,tx}}_{ij,it},$$

where, $r^{it}_{i,j}$ is the uplink transmission rate between ith and the jth FL node, during the itth iteration, which is a function of the bandwidth ($B^j_{i,m}$), and the distance ($d_{i,j}$) between the two nodes, modeled through the Shannon capacity formula in (6.61). Since FL-servers are accessed by multiple VUs/lower layer nodes, we assume for simplicity that the jth node bandwidth is equally shared among the connected VUs and lower layer nodes, i.e., if $u_j = u_l \in \mathcal{U}$, the bandwidth resources of u_l, $B^{l\to(v,r)}_{u,l}$ is shared among all VUs and RSUs connected to it. Also, P^{tx}_i is the ith device transmission power. Similarly, the reception time required to receive data from the jth node by the ith node is given by,

$$T^{\mathrm{FL,rx}}_{ij,it} = \frac{|w^{it}_j|}{r^{it}_{i,j}(B^j_i, d_{i,j})}, \quad E^{\mathrm{FL,rx}}_{ij,it} = P^{\mathrm{rx}}_i \cdot T^{\mathrm{FL,rx}}_{ij,it},$$

Each FL server needs to wait for receiving the data from all the connected VUs and lower layer nodes before performing the FedAvg process. The data reception latency and energy at the jth FL server are given by:

$$T_{j,it}^{\mathrm{FL,rx}} = \max_{i} \left\{ T_{i,it}^{\mathrm{FL,tx}} \right\}, \quad E_{j,it}^{\mathrm{FL,rx}} = \sum_{i} P_{j}^{\mathrm{rx}} \cdot T_{ji,it}^{\mathrm{FL,rx}},$$

With these basic latency and energy elements in hand, we can now define the FL iteration cost in terms of total latency and energy requirements.

FL Iteration Cost

The mth VU FL process cost includes the local computation cost, header processing operation cost at VU and the additional cost depending on the network selection strategy. Thus, for the mth VU, the total FL process cost (in terms of latency and energy consumed) for a single iteration is:

$$\begin{aligned} T_{v,m,it}^{\mathrm{FL}}(\mathbf{a}_{v,m}(\tau_i), \mathbf{b}_{r,n}(\tau_i), \mathbf{c}_{u,l}(\tau_i)) &= T_{v_m}^{\mathrm{FL,c}} + T_{v_m}^{\mathrm{FL,hp}} + \\ \mathbf{a}_{v,m}(\tau_i) \times & \begin{bmatrix} \left[T_{v_m,r_n,it}^{\mathrm{FL,tx}} + T_{r,n,it}^{\mathrm{FL}}(\mathbf{b}_{r,n}(\tau_i), \mathbf{c}_{u,l}(\tau_i)) \right]_{(N_{v,m}^{R} \times 1)} \\ \left[T_{v_m,u_l,it}^{\mathrm{FL,tx}} + T_{u,l,it}^{\mathrm{FL}}(\mathbf{c}_{u,l}(\tau_i)) \right]_{(N_{v,m}^{U} \times 1)} \\ \left[T_{v_m,h,it}^{\mathrm{FL,tx}} + T_{h,it}^{\mathrm{FL}} \right]_{(N_{v,m}^{H} \times 1)} \end{bmatrix} \\ E_{v,m,it}^{\mathrm{FL}}(\mathbf{a}_{v,m}(\tau_i), \mathbf{b}_{r,n}(\tau_i), \mathbf{c}_{u,l}(\tau_i)) &= E_{v_m}^{\mathrm{FL,c}} + E_{v_m}^{\mathrm{FL,hp}} + \\ \mathbf{a}_{v,m}(\tau_i) \times & \begin{bmatrix} \left[E_{v_m,r_n,it}^{\mathrm{FL,tx}} + E_{r,n,it}^{\mathrm{FL}}(\mathbf{b}_{r,n}(\tau_i), \mathbf{c}_{u,l}(\tau_i)) \right]_{(N_{v,m}^{R} \times 1)} \\ \left[E_{v_m,u_l,it}^{\mathrm{FL,tx}} + E_{u,l,it}^{\mathrm{FL}}(\mathbf{c}_{u,l}(\tau_i)) \right]_{(N_{v,m}^{U} \times 1)} \\ \left[E_{v_m,h,it}^{\mathrm{FL,tx}} + E_{h,it}^{\mathrm{FL}} \right]_{(N_{v,m}^{H} \times 1)} \end{bmatrix} \end{aligned}$$

where, for the n-th RSU, the FL process cost for a single iteration is a function of the time/energy required to receive model updates from VUs, the header processing cost, the FedAvg process cost, and the additional cost based upon the network selection strategy adopted by it. Thus, for the case of nth RSU,

$$\begin{aligned}
T^{\mathrm{FL}}_{r,n,it}(\mathbf{b}_{r,n}(\tau_i), \mathbf{c}_{u,l}(\tau_i)) =& T^{\mathrm{FL,rx}}_{r_n,it} + T^{\mathrm{FL,hp}}_{r_n} + T^{\mathrm{FL,FA}}_{r_n} \\
&+ \mathbf{b}_{r,n}(\tau_i) \times \begin{bmatrix} \left[T^{\mathrm{FL,tx}}_{r_n,u_l,it} + T^{\mathrm{FL}}_{u,l,it}(\mathbf{c}_{u,l}(\tau_i))\right]_{(N^U_{r,n}\times 1)} \\ \left[T^{\mathrm{FL,tx}}_{r_n,h,it} + T^{\mathrm{FL}}_{h,it}\right]_{(N^H_{r,n}\times 1)} \end{bmatrix} \\
&+ \left(1 - \sum \mathbf{b}_{r,n}(\tau_i)\right) T^{\mathrm{FL,tx}}_{r,v,it} \\
E^{\mathrm{FL}}_{r,n,it}(\mathbf{b}_{r,n}(\tau_i), \mathbf{c}_{u,l}(\tau_i)) =& E^{\mathrm{FL,rx}}_{r_n,it} + E^{\mathrm{FL,hp}}_{r_n} + E^{\mathrm{FL,FA}}_{r_n} \\
&+ \mathbf{b}_{r,n}(\tau_i) \times \begin{bmatrix} \left[E^{\mathrm{FL,tx}}_{r_n,u_l,it} + E^{\mathrm{FL}}_{u,l,it}(\mathbf{c}_{u,l}(\tau_i))\right]_{(N^U_{r,n}\times 1)} \\ \left[E^{\mathrm{FL,tx}}_{r_n,h,it} + E^{\mathrm{FL}}_{h,it}\right]_{(N^H_{r,n}\times 1)} \end{bmatrix} \\
&+ \left(1 - \sum \mathbf{b}_{r,n}(\tau_i)\right) E^{\mathrm{FL,tx}}_{r,v,it}
\end{aligned}$$

Similarly, for the l-th UAV, the FL process cost is based upon data reception, header processing, FedAvg process, and the additional cost due to the network selection strategy. Thus, for the case of lth UAV, a single iteration cost is,

$$\begin{aligned}
T^{\mathrm{FL}}_{u,l,it}(\mathbf{c}_{u,l}(\tau_i)) =& T^{\mathrm{FL,rx}}_{u_l,it} + T^{\mathrm{FL,hp}}_{u_l} + T^{\mathrm{FL,FA}}_{u_l} \\
&+ \mathbf{c}_{u,l}(\tau_i) \times \left[\left(T^{\mathrm{FL,tx}}_{u_l,h,it} + T^{\mathrm{FL}}_{h,it}\right)_{(N^H_{u,l}\times 1)}\right] \\
&+ \left(1 - \sum \mathbf{c}_{u,l}(\tau_i)\right) \cdot T^{\mathrm{FL,tx}}_{l,v,it} \\
E^{\mathrm{FL}}_{u,l,it}(\mathbf{c}_{u,l}(\tau_i)) =& E^{\mathrm{FL,rx}}_{u_l,it} + E^{\mathrm{FL,hp}}_{u_l} + E^{\mathrm{FL,FA}}_{u_l} \\
&+ \mathbf{c}_{u,l}(\tau_i) \cdot \left[\left(E^{\mathrm{FL,tx}}_{u_l,h,it} + E^{\mathrm{FL}}_{h,it}\right)_{(N^H_{u,l}\times 1)}\right] \\
&+ \left(1 - \sum \mathbf{c}_{u,l}(\tau_i)\right) \cdot E^{\mathrm{FL,tx}}_{l,v,it}.
\end{aligned}$$

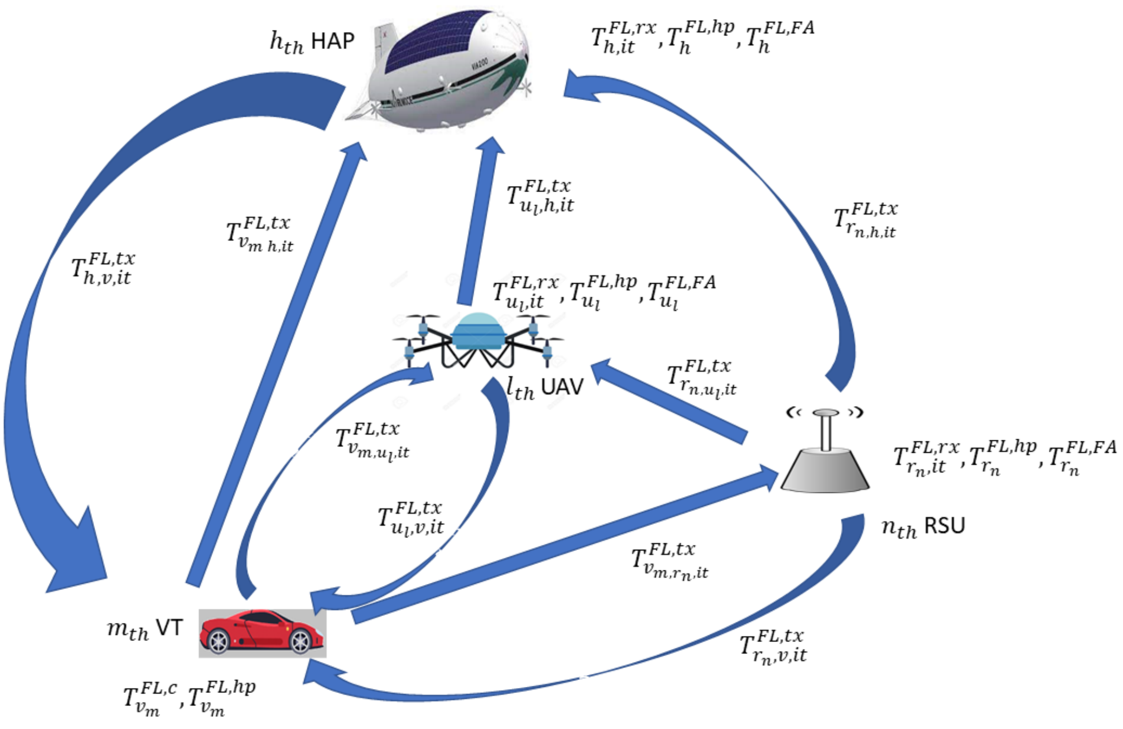


FIGURE 6.16: Distributed FL Process Latency Analysis.

Finally, for the HAP node, the FL process cost for a single iteration is:

$$T_{h,it}^{\text{FL}} = T_{h,it}^{\text{FL,rx}} + T_{h}^{\text{FL,hp}} + T_{h}^{\text{FL,FA}} + T_{h,v_m,it}^{\text{FL,tx}}$$
$$E_{h,it}^{\text{FL}} = E_{h,it}^{\text{FL,rx}} + E_{h}^{\text{FL,hp}} + E_{h}^{\text{FL,FA}} + E_{h,v_m,it}^{\text{FL,tx}}.$$

Fig. 6.16, presents the different latency components considered during the modeling of the FL latency over different nodes. We have avoided including the energy elements for simplicity.

In the end, for each FL iteration the required latency and energy cost for the mth VU is given by[6]:

$$T_{it}^{\mathrm{FL}}(d(v_m, r_n, u_l, \tau_i)) = \max_{m} \left\{T_{v,m,it}^{\mathrm{FL}}(d(v_m, r_n, u_l, \tau_i))\right\}$$
$$E_{it}^{\mathrm{FL}}(d(v_m, r_n, u_l, \tau_i)) = E_{v,m,it}^{\mathrm{FL}}(d(v_m, r_n, u_l, \tau_i))$$

6.3.1.5 Number of FL Iterations Performed

Each FL iteration adds cost in terms of required latency and energy consumed over different platforms. However, it is important to perform a sufficient number of FL iterations for generating the FL model with sufficient accuracy over the real world data. The number of FL iterations performed by VUs depends on the adopted network selection strategy and the sojourn time within each EN coverage area. It is supposed that each VU can participate in the FL process till it belongs to the considered ENs coverage area. Thus,

$$\rho\left(d(v_m, r_n, u_l, \tau_i)\right) \leq \frac{T_{v_m,j}^{soj}(\tau_i)}{T_{it}^{\mathrm{FL}}(d(v_m, r_n, u_l, \tau_i))} \tag{6.62}$$

where, $\rho(d(v_m, r_n, u_l, \tau_i))$ is the number of FL iterations performed by the mth VU whose value is upper bounded by the ratio between the jth ENs sojourn time, $T_{v_m,r_n}^{soj}(\tau_i)$, and the FL iteration time. Here, the jth node corresponds to any RSU, UAV or HAP based upon the network selection strategy adapted by the mth VU. It should be noted that the jth node corresponds to the FL server node that transmit back the global model parameters towards the VU.

[6]For notational simplicity hereafter, we use $d(v_m, r_n, u_l, \tau_i)$ as a decision vector notation indicating the three decision vectors $\mathbf{a}_{v,m}(\tau_i)$, $\mathbf{b}_{r,n}(\tau_i)$,and $\mathbf{c}_{u,l}(\tau_i)$ together

In general the FL process can be stopped if it achieves some predefined stopping criteria, such as the number of FL iterations performed, predefined loss function value, etc. [76, 122]. Therefore, without loss of generality, we introduce ϵ_ν as a convergence parameter in terms of FL global model loss function value, for the FL model corresponding to the service ν. In the past, it has been shown that, in certain environments, it is possible to limit the number of FL iterations required to be performed to achieve the predefined loss function value [76, 124, 125]. However, the maximum number of FL iterations required to be performed can depend upon several parameters such as local environment scenarios, number of VUs participating in the training process, quality of VUs data, etc. Here, we assume that the number of FL iterations required to achieve the FL performance is

$$\rho_\nu^{\max} = \frac{C}{\sqrt{M_\nu^{FL}}} \tag{6.63}$$

function of the number of VUs participating in the training process of the FL model of the νth service, where the square-root models the reduced impact when a higher number of VUs participate to the FL training process. Here, C is a constant representing the maximum number of iterations required for a single VU to achieve the FL model convergence. If VUs have participated to a reduced number of FL iterations when using the FL model for their applications, the performance can be sub-optimal. In such cases VUs might need to pay additional penalty in terms of performance degradation or reduced quality of service.

Here, we introduce a stochastic penalty function $\mathcal{P}_\nu^{\mathrm{FL}}(\rho\left(d(v_m, r_n, u_l, \tau_i)\right))$ for measuring the impact of the number of FL iterations performed over an FL model performance. This analysis is motivated by the work done in [76], for the joint computation offloading and the FL process optimization over VN. If the mth VU requesting the service ν is using the FL process to estimate the parameter x_ν with

$x_{min,v} \leq x_v \leq x_{max,v}$, and the estimated value is given by $\hat{x}_v(\rho\left(d(v_m, r_n, u_l, \tau_i)\right))$, the FL penalty is:

$$\mathcal{P}_v^{\mathrm{FL}}\left(\rho\left(d(v_m, r_n, u_l, \tau_i)\right)\right) = \sqrt{\left(x_v - \hat{x}_v\left(\rho\left(d(v_m, r_n, u_l, \tau_i)\right)\right)\right)^2} \tag{6.64}$$

where $\hat{x}_v$ is estimated by using a stochastic function with truncated normal distribution with probability density function $f_{\hat{x}_v}(\cdot)$ of $\hat{x}_v$ as,

$$f_{\hat{x}_v}(\hat{x}_v; \bar{\mu}, \bar{\sigma}) = \begin{cases} \dfrac{1}{\bar{\sigma}} \dfrac{\xi\left(\frac{\hat{x}_v - \bar{\mu}}{\bar{\sigma}}\right)}{\bar{\Delta}\left(\frac{x_{max,v} - \bar{\mu}}{\bar{\sigma}}\right) - \bar{\Delta}\left(\frac{x_{min,v} - \bar{\mu}}{\bar{\sigma}}\right)} & \\ & \text{if } x_{min,v} \leq \hat{x}_v \leq x_{max,v} \\ 0 & \text{otherwise} \end{cases}$$

and $\xi(\cdot)$ and $\bar{\Delta}(\cdot)$ are, respectively, the probability density function of the related standard normal distribution and its cumulative distribution function, i.e.,

$$\xi(\omega) = \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{\omega^2}{2}\right)}, \quad \bar{\Delta}(\kappa) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{\kappa}{\sqrt{2}}\right)\right].$$

In this work we assume that the mean value of the distribution of x, i.e., $\bar{\mu}$, and its variance, $\bar{\sigma}^2$, are equal to

$$\bar{\mu} = x_v, \quad \bar{\sigma}^2 = \left[\bar{\gamma} \cdot \frac{\rho_v^{\max} - \rho\left(d(v_m, r_n, u_l, \tau_i)\right)}{\rho_v^{\max}}\right]^2$$

where $\bar{\gamma}$ is a numerical constant, used for controlling the variance of the model. The interested reader can have a look to [76] where the same authors considered the above model for estimating the FL iterations vs performance for the computation offloading application over VN.

In the end the total FL latency and energy cost is:

$$T^{\mathrm{FL}}(d(v_m, r_n, u_l, \tau_i)) = \rho T_{it}^{\mathrm{FL}}(d(v_m, r_n, u_l, \tau_i))$$
$$E^{\mathrm{FL}}(d(v_m, r_n, u_l, \tau_i)) = \rho E_{it}^{\mathrm{FL}}(d(v_m, r_n, u_l, \tau_i))$$

6.3.1.6 Problem Formulation

In this work, we aim to perform a communication-efficient FL process over a joint air-ground network. By adopting a proper network selection strategy over different platforms ($\mathcal{A} = \{d(v_m, r_n, u_l, \tau_i)\}, \forall m, n, l$), the aim is to maximize the FL process performance. Thus the main aim is to minimize the joint cost of latency, energy, and the penalty function value measuring the FL process performance:

$$\begin{aligned}\mathbf{P1}: \mathcal{A}^* = \underset{\mathcal{A}}{\operatorname{argmin}} \Bigg\{ \frac{1}{M_v^{FL}} \sum_{m=1}^{M_v^{FL}} \Big(\eta_1 T^{\mathrm{FL}}\left(d\left(v_m, r_n, u_l, \tau_i\right)\right) \\ + \eta_2 E^{\mathrm{FL}}\left(d\left(v_m, r_n, u_l, \tau_i\right)\right) + w_1 \mathcal{P}_v^{\mathrm{FL}}\left(\rho\left(d\left(v_m, r_n, u_l, \tau_i\right)\right)\right)\Big)\Bigg\} \quad (6.65)\end{aligned}$$

subject to the following constraints,

$$\mathbf{C1}: \text{Eq. } (6.58), (6.59), (6.60) \quad (6.66a)$$
$$\mathbf{C2}: \text{Eq. } (6.62) \quad (6.66b)$$
$$\mathbf{C3}: \text{Eq. } \sum B_i^j \leq B^j \quad \forall j \in \mathcal{R}, \mathcal{U}, h \quad (6.66c)$$
$$\mathbf{C4}: 0 \leq \eta_1, \eta_2 \leq 1;\ \eta_1 + \eta_2 = 1, w_1 \geq 0 \quad (6.66d)$$

where $\mathcal{A} = \{d(v_m, r_n, u_l, \tau_i)\}$ is the combined set of network selection decisions of all nodes involved during the FL process, η_1 and η_2 are weighting coefficients for balancing latency and energy consumption, and w_1 is a weighting coefficient for the penalty function. According to (6.66a), each VU, RSU and UAV can communicate with only one EN from the upper layers. Eq. (6.66b) limits the number of iterations performed by each VU depending on the available sojourn time considering the limited HAP coverage. Eq. (6.66c) shows the upper limit on the bandwidth resources of any jth EN, among any RSU, UAV or HAP. The total bandwidth available for the VUs and other nodes connected to any jth EN will be upper bounded by the bandwidth of the jth node, i.e., B^j. According to (6.66d), weighting coefficients η_1 and η_2 can have any value between zero and one with their sum equal to one. Also w_1 can have any positive value.

Proposed Solutions

For solving (6.65), we aim at finding a proper EN selection strategy for creating a highly reliable FL model with reduced latency and energy costs. With multiple edge computing layers and a large number of VUs along the road, the considered problem can be hard to solve. Here, we propose a MDP-based RL approach for finding proper assignment strategies for different nodes. The basic elements of the MDP model include the state space ($\mathcal{S}$), the action space ($\mathcal{A}$), the reward function ($\mathcal{R}$), the discount factor (γ), and the proper environment dynamics, or state transition, probability model ($\mathcal{P}$). Thus, the MDP process can be defined as a tuple given by $\{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma\}$. In order to analyze the performance of the proposed MDP model, we present two multi-dimensional MDP approaches based on the VUs' local environment, as well three benchmark methods for comparison purposes.

6.3.2 MDP Based SOlution Approach

6.3.2.1 Local Environment based Multi-dimensional MDP Model

In the considered network architecture, each VU can be covered by one or more RSUs, UAVs, and one HAP. Thus, different VUs/ENs can have different number of nodes available for communicating the FL updates. In order to properly select the EN for performing the FL, and setting up the MDP parameters, we assume that each VU is able to acquire the local environment parameters through V2X communication links, allowing more personalized MDP models with better accuracy. In particular, we classify the VUs into different groups, based on their local environments, where each group can have a separate state space and action space.

For the mth VU, the main parameters include the number of RSUs ($N_{v,m}^R$), UAVs ($N_{v,m}^U$), and HAP ($N_{v,m}^H$) nodes available for the FL process. In addition, we define three vectors $V_{v_m}^R = \left\{V_{v_m,r_1}^R, \cdots, V_{v_m,r_{\left(N_{v,m}^R\right)}}^R\right\}$ with $V_{v_m,r_n}^R \leq V_{max}^R$, $V_{v_m}^U = \left\{V_{v_m,l_1}^U, \cdots, V_{v_m,l_{\left(N_{v,m}^U\right)}}^U\right\}$ with $V_{v_m,u_l}^U \leq V_{max}^U$, and $V_{v_m}^H = \left\{V_{v_m,h_1}^H, \cdots, V_{v_m,h_{\left(N_{v,m}^H\right)}}^H\right\}$ with $V_{v_m,h}^H \leq V_{max}^H$, corresponding to the number of nodes (i.e., VUs, RSUs, and UAVs) already connected to the each RSU, UAV and HAP node, respectively, covering the mth VU. Here, V_{max}^R, V_{max}^U and V_{max}^H stand for the maximum number of devices that can be served by each RSU, UAV and HAP nodes. Thus, a tuple $\kappa = \{N_{v,m}^R, N_{v,m}^U, N_{v,m}^H, V_{v_m}^R, V_{v_m}^U, V_{v_m}^H\}$ can represent the mth VU local environment. The number of possible κ values, i.e., $\bar{K}$, can depend upon V_{max}^R, V_{max}^U and V_{max}^H. Through V2X communication links, VUs can determine the number of nodes around them. However, since all VUs participate in the FL process simultaneously, their assignment parameters in advance is unknown. Therefore, some assumptions are required. Here, we consider the following two approaches for generating $V_{v_m}^R$, $V_{v_m}^U$, and $V_{v_m}^H$ vectors that can be used to improve the MDP models accuracy.

Minimum Distance Based Assignment Approach In the case of a minimum distance-based approach, each node is assigned to the upper layer node with the minimum possible distance. For example, mth VU is assigned to the nearest RSU, nth RSU is assigned to the nearest UAV, and lth UAV is assigned to the nearest HAP. Thus in general,

$$a_{v,m}(\tau_i) = 1 \Longleftrightarrow n = \underset{n \in N^R_{v,m}}{\operatorname{argmin}}\{d_{v_m,r_n}(\tau_i)\} \tag{6.67a}$$

$$b_{r,n}(\tau_i) = 1 \Longleftrightarrow l = \underset{l \in N^U_{v,m}}{\operatorname{argmin}}\{d_{r_n,u_l}(\tau_i)\} \tag{6.67b}$$

$$c_{u,l}(\tau_i) = 1 \Longleftrightarrow h = \underset{h \in N^H_{v,m}}{\operatorname{argmin}}\{d_{u_l,h}(\tau_i)\} \tag{6.67c}$$

Random Assignment Approach In this approach, each node is assigned to any of the higher layer nodes with a probabilistic rule. We have considered the uniform assignment approach where the probability of assigning the ith node towards the jth upper layer node is given by

$$p(i \rightarrow j) = \frac{1}{U_i^{max}} \tag{6.68}$$

where U_i^{max} indicate the total number of upper layer nodes covering the ith node which can be VU, RSU or UAV.

With these two approaches in hand, different sets of $V^R_{v_m}$, $V^U_{v_m}$, and $V^H_{v_m}$ can be generated, helping to select proper ENs. The two different MDP approaches resulting from these methods are denoted as MDP with minimum distance based assignment approach (MDP-MD), and MDP with random assignment approach (MDP-RA). Later, the performance of these two schemes is compared in the simulation results section.

State Space ($\mathcal{S}$) In general, MDP state space is constituted by all possible states in which MDP agents can find themselves during the exploration of the environment. Finding an appropriate network of ENs, i.e., vehicle to HAP ($v_m \rightarrow h$), vehicle to nth RSU to HAP ($v_m \rightarrow r_n \rightarrow h$), vehicle to lth UAV to HAP ($v_m \rightarrow u_l \rightarrow h$), vehicle to nth RSU to lth UAV to HAP ($v_m \rightarrow r_n \rightarrow u_l \rightarrow h$) can potentially save the FL iteration latency and energy cost and allow VUs to participate to a large number of FL iterations resulting into a better FL model generation. In this work, $\mathcal{S}$ is constituted by multiple number of binary variables corresponding to all $n \in N^R_{v,m}$, $l \in N^U_{v,m}$ and h. In particular, we define

$$S^{r_n}_R(d(v_m, r_n, u_l, \tau_i)) = \begin{cases} 1 & \text{if} \quad v_m \rightarrow r_n \\ & \text{and} \quad \frac{\rho(d(v_m,r_n,u_l,\tau_i))}{\rho_v^{\max}} > \zeta^R_\rho \\ 0 & \text{otherwise} \end{cases}$$

as a binary variable related to $r_n \in N^R_m$, which takes value 1 if the mth VU is assigned to the nth RSU and able to perform a sufficient number of FL iterations, where $0 < \zeta^R_\rho \leq 1$ is a parameter indicating the FL accuracy level that can be based on the service type requested by the users. For example, in case of a critical safety-related service, the FL model accuracy should be high for avoiding possible fatal car crashes due to the failure of the FL models. In this case, ζ^R_ρ should be closer to 1 or even 1. On the other hand, if it is not a high priority/safety-related service, a moderate FL accuracy can be sufficient to serve the user. In such cases ζ^R_ρ can be smaller. Similarly,

$$S^{u_l}_U(d(v_m, r_n, u_l, \tau_i)) = \begin{cases} 1 & \text{if} \quad (v_m \rightarrow u_l \quad \text{or} \quad v_m \rightarrow r_n \rightarrow u_l) \\ & \text{and} \quad \frac{\rho(d(v_m,r_n,u_l,\tau_i))}{\rho_v^{\max}} > \zeta^U_\rho \\ 0 & \text{otherwise} \end{cases}$$

is a binary variable related to $u_l \in N_m^U$, which takes 1 if the mth VU is assigned to the lth UAV and able to perform a sufficient number of FL iterations. Here, $0 < \zeta_\rho^U \leq 1$ is the parameter indicating the FL accuracy level as a function of the service type requested by the users. Finally,

$$S_H^h(d(v_m, r_n, u_l, \tau_i)) = \begin{cases} 1 & \text{if } (v_m \rightarrow h \text{ or } v_m \rightarrow r_n \rightarrow h \text{ or } v_m \rightarrow u_l \rightarrow h, \\ & \quad \text{or } v_m \rightarrow r_n \rightarrow u_l \rightarrow h) \text{ and } \frac{\rho(d(v_m,r_n,u_l,\tau_i))}{\rho_v^{\max}} > \zeta_\rho^H \\ 0 & \text{otherwise} \end{cases}$$

is a binary variable related to h, which takes value 1 if the mth VU is assigned to the HAP h and able to perform a sufficient number of FL iterations. Here, $0 < \zeta_\rho^H \leq 1$ is the parameter indicating the FL accuracy level based upon the service type requested by the users.

In the end, if the mth VUs local environment is modeled through the tuple κ, the complete state vector is given as,

$$S_\kappa = \left\{ S_R^1, \cdots, S_R^{N_{v,m}^R}, S_U^1, \cdots, S_U^{N_m^U}, S_H^h \right\}$$

Action Space ($\mathcal{A}$) If the mth VU local environment is modeled through the tuple κ, the action space ($\mathcal{A}_\kappa = \{a_\kappa(\tau_i)\}$) includes all possible actions a_κ that can be taken by the MDP agent corresponding to the κ. In the considered FL network selection problem, agents can select ENs belonging to the different networking layers. Therefore, the generic action space is defined as,

$$\mathcal{A}_\kappa = \big\{ (v_m \rightarrow r_n), (v_m \rightarrow u_l), (v_m \rightarrow h), (v_m \rightarrow r_n \rightarrow u_l), (v_m \rightarrow u_l \rightarrow h), \\ (v_m \rightarrow r_n \rightarrow h,), (v_m \rightarrow r_n \rightarrow u_l \rightarrow h) \big\} \ \forall m, l \quad (6.69)$$

Reward Function ($\mathcal{R}$) MDP agents can receive a positive or negative reward based upon the current state and the action taken. Here, we consider the weighted sum of latency, energy and penalty functions cost required to complete a single FL iteration as a reward received by the agent based upon its state and action. Thus,

$$\mathcal{R}_{v,\kappa}(s,a) = \eta_1 T^{\mathrm{FL}}(s_\kappa, a_\kappa) + \eta_2 E^{\mathrm{FL}}(s_\kappa, a_\kappa) + w_1 \mathcal{P}_v^{\mathrm{FL}}(\rho\,(s_\kappa, a_\kappa))$$

MDP Environment Dynamics ($\mathcal{P}$) MDP environment dynamics model the behavior of the MDP environment in terms of state transition probabilities based upon the agents' current state and the actions performed. The probability of MDP agent finding itself into state s' when it performs the action a from state s is given as $P(s'|s,a)$. Modeling such state transition probability over dynamic vehicular environments can be challenging. We propose a time-dependent state transition probability equation based upon the MDP agent's local environment. In general, for scenario κ, the state transition probability at τ_i is given by

$$\begin{aligned} P\left(s_\kappa(\tau+\delta)|s_\kappa(\tau), a_\kappa(\tau)\right) = P\big(&\left\{S_R^{r_n}(\tau+\delta), S_U^{u_l}(\tau+\delta), S_H^{h}(\tau+\delta)\right\} \\ &| \left\{S_R^{r_n}(\tau), S_U^{u_l}(\tau), S_H^{h}(\tau)\right\}, a_\kappa(\tau)\big) \end{aligned}$$

which represents the state transition probability for state $s_\kappa(\tau+\delta)$ for the MDP agent from current state $s_\kappa(\tau)$ taking action $a_\kappa(\tau)$. Here, δ is the MDP time step. Since VU can connect to only one node in a given time interval, the events $S_R^{r_n}$, $S_U^{u_l}$ and S_H^{h} can be considered as an independent events, which results into,

$$\begin{aligned} P(s_\kappa(\tau+\delta)|s_\kappa(\tau), a_\kappa(\tau)) = &P\left(S_R^{r_n}(\tau+\delta)| \left\{S_R^{r_n}(\tau), S_U^{u_l}(\tau), S_H^{h}(\tau)\right\}, a_\kappa(\tau)\right) \\ &\cdot P\left(S_U^{u_l}(\tau+\delta)| \left\{S_R^{r_n}(\tau), S_U^{u_l}(\tau), S_H^{h}(\tau)\right\}, a_\kappa(\tau)\right) \end{aligned}$$

$$\cdot P\left(S_H^h(\tau+\delta)\}|\left\{S_R^{r_n}(\tau), S_U^{u_l}(\tau), S_H^h(\tau)\right\}, a_\kappa(\tau)\right) \quad (6.70)$$

In particular, with various communication links, e.g., V2V, V2R, V2I, vehicular-based MDP agents can acquire useful information about the surrounding environment (i.e., tuple κ), which can be used to model the state transition probabilities. The transition probability expressions are modeled as exponential functions based upon various local environment parameters. The state transition probability expressions for $S_R^{r_n}$ for the κth MDP agent with current state $s_\kappa(\tau)$ and performing action $a_\kappa(\tau)$ is defined as,

$$P\left(\left\{S_R^{r_n}(\tau+\delta)=1\right\}|s_\kappa(\tau), a_\kappa(\tau)\right) = \begin{cases} \exp(-\lambda_n^R(\tau_i)) & \text{if } a_\kappa(\tau) \in v_m \to r_n \\ 0 & \text{else} \end{cases} \quad (6.71)$$

$$P\left(\left\{S_R^{r_n}(\tau+\delta)=0\right\}|s_\kappa(\tau), a_\kappa(\tau)\right) = 1 - P\left(\left\{S_R^{r_n}(\tau+\delta)=1\right\}|s_\kappa(\tau), a_\kappa(\tau)\right) \quad (6.72)$$

corresponding, respectively, to the probability that the MDP agent will be in state with $S_R^{r_n}(\tau+\delta)=1$ and $S_R^{r_n}(\tau+\delta)=0$ by taking action $a_\kappa(\tau)$ from current state $s_\kappa(\tau)$. Also, $\lambda_n(\tau_i) = \delta_1^R \cdot V_{v_m,r_n}^R(\tau_i) + \delta_2^R \cdot d_{v_m,r_n}(\tau_i) + \delta_3^R / T_{v_m,r_n}^{soj}(\tau_i)$ models the impact of VUs local environment over the state transition probability values. According to $\lambda_n(\tau_i)$, if the mth VU through action $a_\kappa(\tau)$ selects the RSU n with high V_{v_m,r_n}^R and $d_{v_m,r_n}(\tau_i)$, the VUs might not be able to perform the required number of FL iterations. Also if VU selects the RSU with a high sojourn time value, it can perform a sufficient number of FL iterations, resulting in a higher probability that occurs $S_R^{r_n}(\tau+\delta)=1$; δ_1^R, δ_2^R and δ_3^R are the weighing coefficients used to associate proper weights towards each parameters. Next, for $S_U^{u_l}$,

$$P\left(\left\{S_U^{u_l}(\tau+\delta)=1\right\}|s_\kappa(\tau),a_\kappa(\tau)\right)=\begin{cases}\exp(-\lambda_{l,1}^U(\tau_i)) & \text{if } a_\kappa(\tau)\in v_m\longrightarrow u_l\\ \exp(-\lambda_{l,2}^U(\tau_i)) & \text{if } a_\kappa(\tau)\in v_m\longrightarrow r_n\longrightarrow u_l\\ 0 & \text{else}\end{cases}$$

$$P\left(\left\{S_U^{u_l}(\tau+\delta)=0\right\}|s_\kappa(\tau),a_\kappa(\tau)\right)=1-P\left(\left\{S_U^{u_l}(\tau+\delta)=1\right\}|s_\kappa(\tau),a_\kappa(\tau)\right))$$

corresponding to the probabilities that MDP agent in state $s_\kappa(\tau)$ by taking action $a_\kappa(\tau)$ will find itself in state with $S_U^{u_l}(\tau+\delta)=1$ and $S_U^{u_l}(\tau+\delta)=0$, respectively.

Since VU can be connected to the UAV directly, or through RSU as a intermediate node, two separated cases are provided for increasing the accuracy of the MDP framework. Here, $\lambda_{l,1}^U(\tau_i) = \delta_1^U V_{v_m,u_l}^U(\tau_i)+\delta_2^U d_{v_m,u_l}(\tau_i)+\delta_3^U/T_{v_m,u_l}^{soj}(\tau_i)$ and $\lambda_{l,2}^U(\tau_i)=\delta_4^U V_{v_m,r_n}^R(\tau_i)+\delta_1^U V_{v_m,u_l}^U(\tau_i)+\delta_5^U(d_{v_m,r_n}(\tau_i)+d_{r_n,u_l}(\tau_i))+\delta_3/T_{v_m,u_l}^{soj}(\tau_i)$ are the two parameters measuring the impact of surrounding environment over the transition probabilities, while $\delta_1^U,\cdots,\delta_5^U$ correspond to the weighting coefficients for properly balancing the impact of various environment parameters. In the end, for S_H^h:

$$P\left(\left\{S_H^h(\tau+\delta)=1\right\}|s_\kappa(\tau),a_\kappa(\tau)\right)=\begin{cases}\exp(-\lambda_{h,1}^H(\tau_i)) & \text{if } a_\kappa(\tau)\in v_m\longrightarrow h\\ \exp(-\lambda_{h,2}^H(\tau_i)) & \text{if } a_\kappa(\tau)\in v_m\longrightarrow r_n\longrightarrow h\\ \exp(-\lambda_{h,3}^H(\tau_i)) & \text{if } a_\kappa(\tau)\in v_m\longrightarrow u_l\longrightarrow h\\ \exp(-\lambda_{h,4}^H(\tau_i)) & \text{if } a_\kappa(\tau)\in v_m\longrightarrow r_n\longrightarrow u_l\longrightarrow h\\ 0 & \text{else}\end{cases} \tag{6.73}$$

$$P\left(\left\{S_H^h(\tau+\delta)=0\right\}|s_\kappa(\tau),a_\kappa(\tau)\right)=1-P\left(\left\{S_H^h(\tau+\delta)=1\right\}|s_\kappa(\tau),a_\kappa(\tau)\right) \tag{6.74}$$

are the state transition probabilities corresponding to $S_H^h(\tau+\delta) = 1$ and $S_H^h(\tau+\delta) = 0$, respectively. Since VUs can communicate with the HAP node through various links, different cases are presented based upon the action taken. In (6.73),

$$\begin{aligned}
\lambda_{h,1}^H(\tau_i) =& \delta_1^H V_{v_m,h}^H(\tau_i) + \delta_2^H d_{v_m,h}(\tau_i) + \delta_3^H / T_{v_m,h}^{soj}(\tau_i) \\
\lambda_{h,2}^H(\tau_i) =& \delta_4^H V_{v_m,r_n}^R(\tau_i) + \delta_1^H V_{v_m,h}^H(\tau_i) + \delta_5^H (d_{v_m,r_n}(\tau_i) + d_{r_n,h}(\tau_i)) + \delta_3^H / T_{v_m,h}^{soj}(\tau_i) \\
\lambda_{h,3}^H(\tau_i) =& \delta_6^H V_{v_m,u_l}^U(\tau_i) + \delta_1^H V_{v_m,h}^H(\tau_i) + \delta_7^H (d_{v_m,u_l}(\tau_i) + d_{u_l,h}(\tau_i)) + \delta_3^H / T_{v_m,h}^{soj}(\tau_i) \\
\lambda_{h,4}^H(\tau_i) =& \delta_4^H V_{v_m,r_n}^R(\tau_i) + \delta_6^H V_{v_m,u_l}^U(\tau_i) + \delta_1^H V_{v_m,h}^H(\tau_i) + \delta_8^H (d_{v_m,r_n}(\tau_i) + d_{r_n,u_l}(\tau_i) \\
& + d_{u_l,h}(\tau_i)) + \delta_3^H / T_{v_m,h}^{soj}(\tau_i)
\end{aligned}$$

are the parameters modeling the surrounding environments impact over the state transitions. In the end, by using (6.70), (6.74) can be used to find the state transition probability in any interval τ.

6.3.2.2 MDP-Based FL Network Selection Strategy

For the MDP model corresponding to the κth agent, the solutions' set can be defined as a policy function $\pi_\kappa = \{\pi_\kappa(s_\kappa(\tau_i + \delta)), \forall \delta\}$ that maps every state $s_\kappa \in \mathcal{S}$ to action $a_\kappa \in \mathcal{A}$. Selecting different actions can result in different policy functions, where the aim is to find an optimal policy that corresponds to the minimum cost in terms of delay, energy and FL process penalty value. For every policy π_κ, a value function $V_{\pi_\kappa}(s_\kappa(\tau_i))$, corresponding to a state $s_\kappa(\tau_i)$ can be defined for analyzing its performance. In general, $V_{\pi_\kappa}(s_\kappa(\tau_i))$ corresponds to an expected value of a discounted sum of total reward received by following the policy π_κ from state $s_\kappa(\tau_i)$, and can be defined as:

$$V_{\pi_\kappa}(s_\kappa(\tau_i)) = \mathbb{E}\left\{\sum_{\delta=0}^{\Delta} \gamma^\delta R\left(s_\kappa(\tau_i + \delta), \pi_\kappa(s_\kappa(\tau_i + \delta))\right)\right\}$$

where $\gamma \in [0,1]$ is the discount factor, $R(s_\kappa(\tau_i+\delta), \pi_\kappa(s_\kappa(\tau_i+\delta)))$ is the immediate reward received for following a policy π_κ at time $\tau_i+\delta$ from the state $s_\kappa(\tau_i+\delta)$, Δ is the maximum number of steps considered during the MDP evaluation, i.e., episode length, and $\mathbb{E}(\cdot)$ corresponds to the expected value. Thus, the value function analyzes the particular policy function by assigning a numeric value to each state and can be utilized to compare the performance of different policies. In the end, the following optimization problem can be formulated in order to be able to find the best possible policy function associated with state $s_\kappa(\tau_i)$:

$$V(s_\kappa(\tau_i)) = \min_{\pi_\kappa \in \Pi_\kappa} V_{\pi_\kappa}(s_\kappa(\tau_i)) \tag{6.75}$$

where Π_κ corresponds to the set of policy functions that can be explored.

As shown by many works (e.g., [88, 89]), the problem defined in (6.75), can converge into a Bellman optimality equation given by:

$$\begin{aligned} V(s_\kappa(\tau_i)) = \min_{a_\kappa(\tau_i)\in A_\kappa(\tau_i)} \Bigg\{ & R(s_\kappa(\tau_i), a_\kappa(\tau_i)) + \\ & \gamma \sum_{s_\kappa(\tau_i+\delta)\in \mathcal{ST}} Pr\left\{s_\kappa(\tau_i+\delta) \mid s_\kappa(\tau_i), a_\kappa(\tau_i)\right\} V(s_\kappa(\tau_i+\delta)) \Bigg\} \end{aligned} \tag{6.76}$$

Different approaches can be used to solve the problem in (6.76); however, the value iteration approach is widely known for its fast convergence and easy implementation. Therefore, below we present a value iteration approach aimed at solving the MDP designed in the previous section for finding an optimal policy that corresponds to the minimization of a FL process time and energy over VN.

The value iteration method allows finding an optimal policy and value function for the MDP models. The Algorithm 15 describes the steps involved during the value iteration process. For every agent κ, the process begins by initializing the values of each state to ∞ and iteration count (it) to 0 (Line 2). For each state-action pair, the state value is determined by using (6.77) (Line 5). The state value and a corresponding optimal policy $(\pi_{\kappa}^{*}(s_{\kappa}(\tau_i)))$ associated with state s_{κ} is determined by using (6.78) and (6.79) (Lines 7-8). The iterative process continues till the change in the all states values becomes less than the predefined convergence parameter ϵ (Lines 10-13). In the end, the algorithm returns the set of optimal policy functions $\{\pi_{\kappa}^{*}\}$ associated with all possible scenarios in which VUs can find themselves over the road (Line 15).

Algorithm 15 MDP Value Iteration

```
Input: $\epsilon, \gamma, S_\kappa, A_\kappa, Pr, \bar{K}, \Delta$
Output: $\{\pi^*_\kappa\}$

 1: for $\kappa \in \bar{K}$ do
 2:     Initialize $it = 0$, $V^0(s_\kappa(\tau_i)) = \infty, \forall s_\kappa(\tau_i)$
 3:     for $s_\kappa(\tau_i) \in S_\kappa$ do
 4:         for $a_\kappa(\tau_i) \in A_\kappa$ do
 5:
                $$V^{it+1}(s_\kappa(\tau_i), a_\kappa(\tau_i)) \leftarrow R(s_\kappa(\tau_i), a_\kappa(\tau_i)) + \gamma \sum_{s_\kappa(\tau_i+\delta)\in s_\kappa} Pr(s_\kappa(\tau_i+\delta) \mid s_\kappa(\tau_i), a_\kappa(\tau_i)) v^{it}(s_\kappa(\tau_i+\delta)) \quad (6.77)$$
 6:         end for
 7:
            $$V^{it+1}(s_\kappa(\tau_i)) = \min_{a_\kappa(\tau_i)} V^{it+1}(s_\kappa(\tau_i), a_\kappa(\tau_i)) \quad (6.78)$$
 8:
            $$\pi^*_\kappa(s_\kappa(\tau_i)) = \operatorname{argmin}_{a_\kappa(\tau_i)} V^{it+1}(s_\kappa(\tau_i), a_\kappa(\tau_i)) \quad (6.79)$$
 9:     end for
10:     if any $|v^{it+1}(s_\kappa(\tau_i)) - v^{it}(s_\kappa(\tau_i))| > \epsilon$ then
11:         $it = it + 1$
12:     else
13:         $\pi^*_\kappa = \{\pi^*_\kappa(s_\kappa(\tau_i))\}$
14:     end if
15: end for
16: return $\{\pi^*_\kappa\}$
```

The time complexity of the traditional value iteration process can be analyzed as $O(\Delta|\mathcal{S}| \cdot |\mathcal{A}|)$ with Δ being the maximum number of time steps considered, $|\mathcal{S}|$ state space dimension, and $|\mathcal{A}|$ representing the action space. With the involvement of $\bar{K}$ scenarios, the time complexity expression becomes $O(\bar{K}\Delta|\mathcal{S}| \cdot |\mathcal{A}|)$. The scenario-based modeling can reduce the state and action space dimensions significantly. Additionally, time-dependent state transition probabilities can reduce the overall uncertainty in the MDP process.

6.3.3 Benchmark Methods

For analyzing the performance of the proposed MDP model, we have considered the following benchmark methods.

6.3.3.1 Conventional Centralized FL Process (C-FL)

In the case of a conventional centralized FL process, each VU transmits its model updates to the centralized HAP server. Thus, $v_m \rightarrow h,\ \forall v \in \mathcal{V}$. This approach can reduce the overall processing costs in terms of intermediate layer processing and averaging operations performed over RSUs and UAVs. However, possible long-distance communication links between VUs and HAP can limit the performance in terms of link failures, high energy costs, limited users participating in the FL process, etc.

6.3.3.2 Minimum Distance Based FL Process (MD-FL)

In this case, the FL process assumes that each node communicates with the nearest nodes from the upper layer. Thus, each participating VU can select the shortest distance RSU node for transmitting its update, which then process and transmit the aggregated update vectors towards the nearest UAV for further processing. In the end, HAP collects data from all the participating UAV terminals for generating the global model, which it then broadcasts back towards VUs. Eqs. (6.67a)-(6.67c) can be used to determine the minimum distance assignment vectors for different nodes.

6.3.3.3 Random Assignment Based FL Process (RA-FL)

In this approach, nodes involved in the FL process (i.e., VUs, RSUs, UAVs, and HAP) follow the random assignment strategy in (6.68). Thus, each VU selects the one EN from a set of RSUs, UAVs, and HAP covering it. Similarly, RSUs can either be connected to the UAV/HAP or can also communicate back the results to VUs. UAVs also followed the same strategy, where they can either send their data to HAP or return it to the VUs for the next round of the FL process.

6.3.3.4 FedCPF inspired RSU-based benchmark solution for the considered scenario (FedR-FL)

In [131], authors have proposed a FedCPF approach based upon a customized local training strategy, partial client participation, and flexible aggregation strategies. Here we considered a FedCPF-inspired, RSU-based benchmark approach where VUs are performing the local training process and transmitting the model parameters to the nearest RSU node. The client selection strategy of the FedCPF approach is considered where participation of each VU in the FL process is based upon a probability P_{sel} (i.e., P_{sel} is the probability of the client being a part of FL training, while $(1 - P_{sel})$ is the probability that the client will opt out from the LF training).

The other two strategies of customized local training strategy and the deadline-based server aggregation strategy are based upon the RSU sojourn time constraint. In particular, VUs' participation in the FL process is limited by its dynamicity and the RSU coverage range.

6.3.4 Performance Evaluation

The value iteration algorithm for solving the MDP model and the benchmark methods previously described are simulated over a Python-based simulator, using ML-related libraries such as NumPy, Pandas, Matplotlib. In Table 6.2, the main simulation parameters are shown for the considered network architecture. The service area is under the coverage of one HAP, 20 UAVs and 40 RSUs. A variable number of VUs between 200 and 700 are considered, assuming that each one is requesting service v with a probability equal to 0.2. Each VU is traveling with a variable speed as modeled in (6.54) with μ=10 m/s and σ=1. The maximum number of FL iterations required to achieve the proper performance, as defined in (6.63), consider that $C = 1000$. Also $P_{sel} = 0.7$ is used for the FedCPF-inspired RSU-based benchmark solution approach. Each VU has a FL dataset of size $|\mathcal{D}_{v_m}| = 5,000$ samples. During the FL training process, $\psi_d = 1500$ and $T_{i,v}^{\mathrm{FA}} = 1$ ms. The maximum number of nodes covering any VU is given by $R_{max} = 3$, $U_{max} = 2$ and $H_{max} = 1$. Additionally a maximum number of nodes served by each RSU $v_{max}^{R} = 8$, each UAV $v_{max}^{U} = 16$, and HAP $v_{max}^{H} = 32$ are considered. With the multi-core processing hardware of ENs, these users can be grouped into different levels based on the number of cores.

The weighting coefficients used for modeling the transition probabilities are defined as, $[\delta_1^R, \delta_2^R, \delta_3^R] = [0.025, 0.004, 0.5]$, $[\delta_1^U, \delta_2^U, \delta_3^U, \delta_4^U, \delta_5^U] = [0.2, 0.0125, 0.1, 0.2, 0.005]$, $[\delta_1^H, \delta_2^H, \delta_3^H, \delta_4^H, \delta_5^H, \delta_6^H, \delta_7^H, \delta_8^H] = [100, 0.4, 100, 200, 1.25, 100, 0.4, 1.25] \cdot 10^{-3}$.

TABLE 6.2: Simulation parameters

HAP Coverage (R_h)	1.2 km
UAV Coverage ($R_{u,l}$)	100 m
RSU Coverage (($R_{r,n}$))	50 m
VU Computation Cap. ($c_{v,m} \cdot f_{v,m}$)	10 GFLOPS
RSU Computation Cap. ($c_{r,n} \cdot f_{r,n}$)	20 GFLOPS
UAV Computation Cap. ($c_{u,l} \cdot f_{u,l}$)	20 GFLOPS
HAP Computation Cap. ($c_h \cdot f_h$)	40 GFLOPS
HAP Altitude ($\bar{h}_h$)	10 km
UAV Altitude ($\bar{h}_{u,l}$)	1 km
HAP Bandwidth ($B_h^{h\to(v,r,l)}$)	250 MHz
UAV Bandwidth ($B_{u,l}^{l\to(v,r)}$)	75 MHz
RSU Bandwidth ($B_{r,n}^{r\to v}$)	25 MHz
VU Speed Range ($\vec{v}_{\min}, \vec{v}_{\max}$).	(8 m/s m/s, 14 m/s)
HAP Power (Pt_h, Pr_h)	(1.1, 0.9) W
UAV Power ($Pt_{c,u}^l, Pr_{c,u}^l$)	(1.2, 1) W
RSU Power ($Pt_{c,r}^n, Pr_{c,r}^n$)	(1.3, 1.2) W
VU Power ($P_{c,v}^m, Pt_{c,v}^m, Pr_{c,v}^m$)	(1.1, 1.5, 1.3) W
Noise Power (N_T)	−110 dbm [132]
FLOPs Required ($\psi_{pre}, \psi_{post}, \psi_{FA}$)	$10^4, 10^4, 10^5$ FLOPs
Weighting Coefficients (η_1, η_2, w_1)	(0.5, 0.5, 1)

The weighting coefficients have been defined so that different local environment parameters, defined in the transition probability equations (6.72)-(6.74), have a value range consistent among them during the MDP state evaluation. For the case of FL penalty function $x = 0.5$ with $0 \le x \le 1$ is used along with $\bar{\lambda} = 0.3$. Additionally, ζ_ρ^R, ζ_ρ^U, and ζ_ρ^H are set to 0.7. During value iteration process $\gamma = 0.9$, $\epsilon = 0.01$, $\Lambda = 0.1$ and episode length $\Delta = 200$ are used.

6.3.4.1 Numerical Results

In the following, we present the main performance results including the FL cost, latency, energy, FL penalty, and the average number of FL iterations performed by VUs for different methods.

FL Process Cost The main objective of this work is to jointly reduce the overall latency, energy, and FL penalty. Fig. 6.17 shows the performance in terms of FL overall cost for the MDP schemes and the benchmark methods previously presented. It can be observed that both MDP methods outperform the benchmark approaches as the number of VUs increases.

With a reduced number of VUs, with a fully distributed FL process, the MD-FL approach requires higher cost mainly due to several processing operations performed at different layers. On the other hand, a fully centralized C-FL method has reduced costs due to the presence of a limited number of VUs requesting the resources from the centralized HAP node. However, if the number of VUs is higher, the overall cost of the C-FL approach grows fast mainly due to the higher communication distances and the limited resources of a HAP node. With this, the C-FL cost becomes higher than the other benchmark methods with the increasing density of VUs. Similar effects can be seen later in the latency and energy plots shown in Figs. 6.18 and 6.19. The other two benchmark approaches (RA-FL and FedCPF-inspired method) have a slightly better performance mainly due to the reduced communication distances and reduced processing operations compared to the fully distributed MD-FL and a fully centralized C-FL approaches. However, the imperfect/static edge node selection without considering the local environment parameters and the available resources, the performance of the benchmark approaches compared to the proposed MDP solutions.

On the other hand, the proposed MDP solutions, with network selection based upon the VUs local environments and the available resources of ENs, are able to keep the FL process cost under the limit. In particular, the MDP-RA method outperforms all other approaches. For the case of the MDP-MD, the VUs local environment is modeled through the assignments of the FL devices to the nearest nodes with less flexibility, i.e., VUs can be assigned to the RSUs, RSUs can be assigned to the UAVs, and UAVs are assigned to the HAP node. On the other hand, the MDP-RA approach is more flexible, where each node can select any higher layer entities, i.e., VUs can be assigned to the RSU, UAVs, or HAP. Therefore, MDP-RA method outperforms the MDP-MD in terms of overall cost, as well FL latency, and energy, as later shown in Figs. 6.18 and 6.19.

With the upcoming latency-constrained vehicular applications and services demanding ML models with high accuracy, it is important to perform the distributed learning process, such as FL, in a limited time and with reduced energy consumption. Thus, the proposed distributed learning framework with efficient network selection strategies allows a large number of VUs to participate in the training process with reduced cost and a huge advantage over the traditional methods.

FL Latency Performance For each FL iteration, the FL process is impacted by communication, training, and processing latency. In Fig. 6.18, we present the average FL iteration latency for various MDP and benchmark methods. In particular, MDP-MD and MDP-RA methods, with proper node selections can perform the FL process with reduced latency compared with other methods. As for the previous case, with fewer nodes requesting resources, the C-FL method performs better compared to the MD-FL and RA-FL methods. However, when VUs are more, C-FL performance in terms of latency requirements degrades drastically. On the other hand with a distributed FL process, the MD-FL approach induces higher latency with lower VUs. However, with higher VUs its performance is better than the C-FL method mainly due to the distribution of the FL process over the multiple edge nodes. The fedCPF-inspired approach selects the RSU nodes for the averaging operation limiting the latency costs in the beginning. However, with the limited sojourn

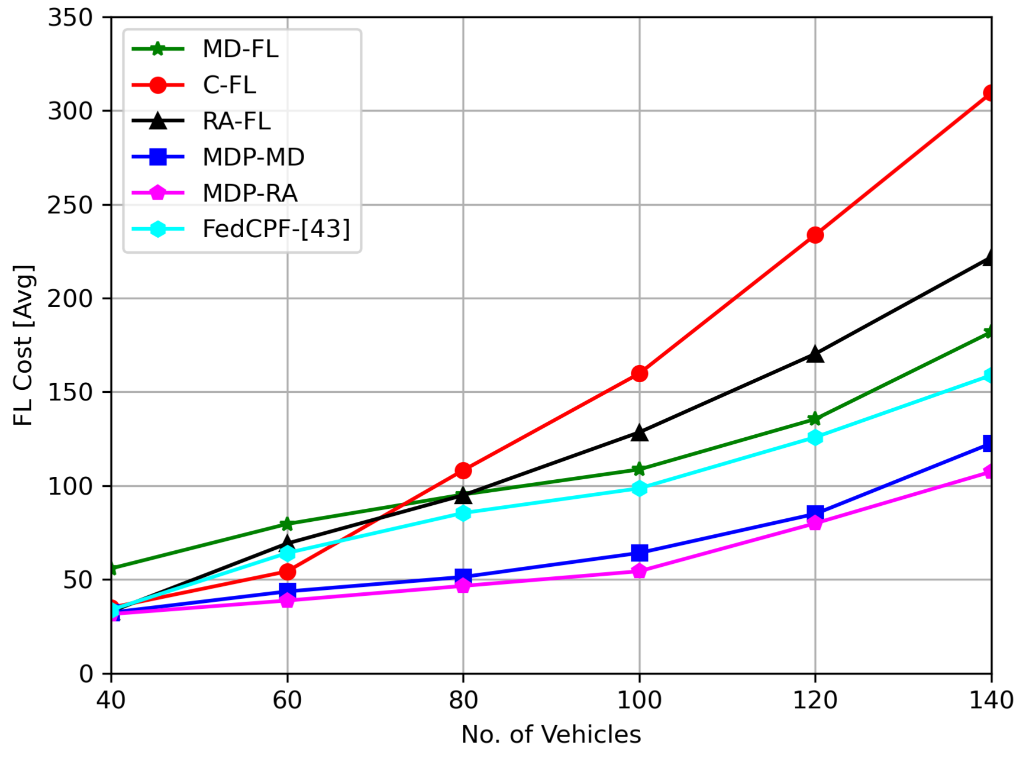


FIGURE 6.17: Performance results in terms of overall cost function with variable number of active vehicles.

time, VUs are unable to perform a sufficient number of iterations resulting in the higher FL penalty as shown later in Fig. 6.20. With a higher number of VUs, the performance of the FedCPF-inspired approach degrees mainly due to the limited RSU resources and the imperfect node selection.

The MDP approaches, especially the MDP-RA method, jointly reduce both communication and processing latency by distributing the FedAvg process over a sufficient number of edge nodes. Therefore, the proposed methods can efficiently train the FL model over the distributed multi-layered VN environments.

FL Energy Performance It is important to reduce the FL process energy cost given the involvement of different T/NT networking platforms (i.e., VUs, UAVs)

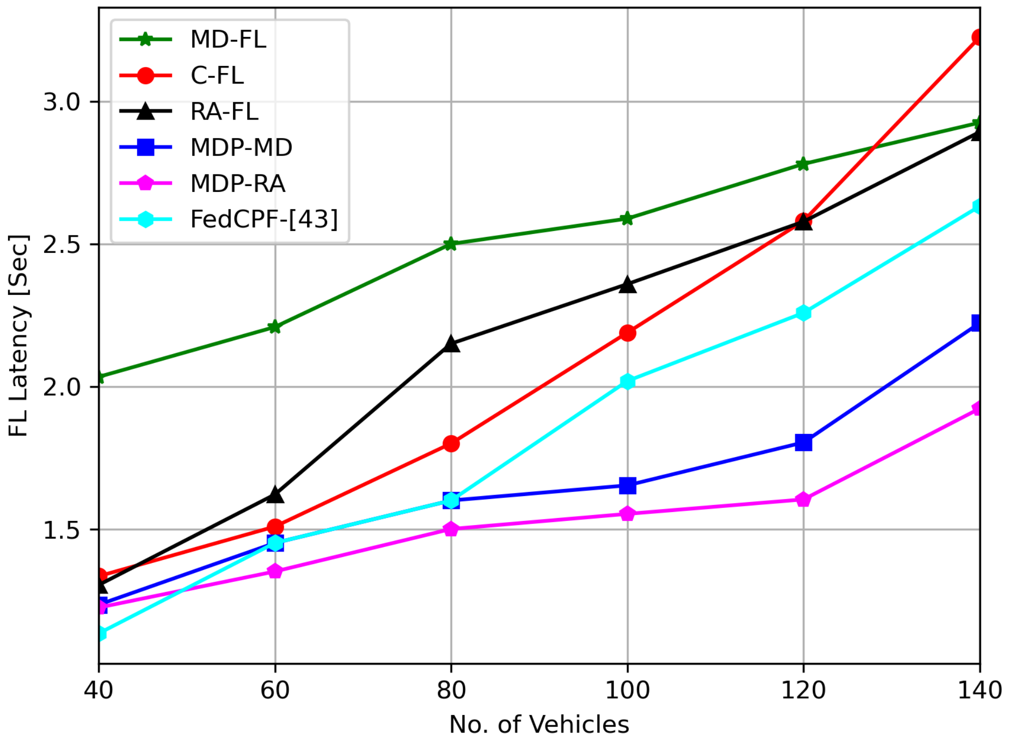


FIGURE 6.18: FL latency with variable number of active vehicles.

with scarce energy resources. The FL process consumes energy for the local training operations, data communication, and FL data processing over servers. In Fig. 6.19, we present the performance in terms of energy spent by the different methods. Similar to the latency performance, the C-FL process energy performance is better for a reduced number of VUs, due to the involvement of a limited number of VUs and reduced FedAvg process cost. However, when VUs are more, the VUs energy requirements become high mainly due to long-distance communication over limited bandwidth resources. Compared with the C-FL method, MD-FL has an advantage in terms of reduced communication distances/costs. However, with the repetition of the FedAvg process over each layer, the energy cost increases. On the other hand, the proposed MDP methods can reduce both communication and computation process energy requirements simultaneously by properly distributing the FL process over multi-layered VN. By utilizing the local environment knowledge, MDP methods can select proper ENs with sufficient resources and, as a result, are able to perform the FL process with reduced energy requirements.

It should also be noted that the energy performance of the C-FL method degrades quickly compared to its latency performance. This is mainly because every VUs involved in the FL process of the C-FL approach requires communication with the centralized HAP node. Due to this, the energy cost induced by the individual VUs can be higher compared to the other benchmark methods. This trend can also be seen in the overall cost performance in Fig. 6.17. The results from Figs. 6.18 and 6.19 can also highlight the issues of the well-known straggler effect in the FL framework with the traditional benchmark methods and the necessity to counter such effects with the new solutions. The proposed MDP-based methods can mitigate such effects as highlighted by the performance in Figs. 6.17-6.19.

FL Penalty Performance With the adopted network selection strategy, if VUs fail to perform a sufficient number of FL iterations, the FL model performance may not be adequate. We have modeled the impact of the number of FL iterations performed by VUs in terms of a stochastic penalty function presented in (6.64). In Fig. 6.20, we show the average FL penalty value for different sets of VUs for the proposed methods. The benchmarks, with inadequate FL process, fail to perform the required number of FL iterations resulting in the higher FL penalties. The FedCPF-inspired approach selects the nearby RSU node for limiting the FL communication cost, which as result limits the number of FL iterations performed by VUs, inducing the heavy FL penalty. The other two benchmark methods, MD-FL and RA-FL also suffer from a large penalty due to the reduced number of FL iterations performed mainly due to the high latency per FL iteration with constrained sojourn times.

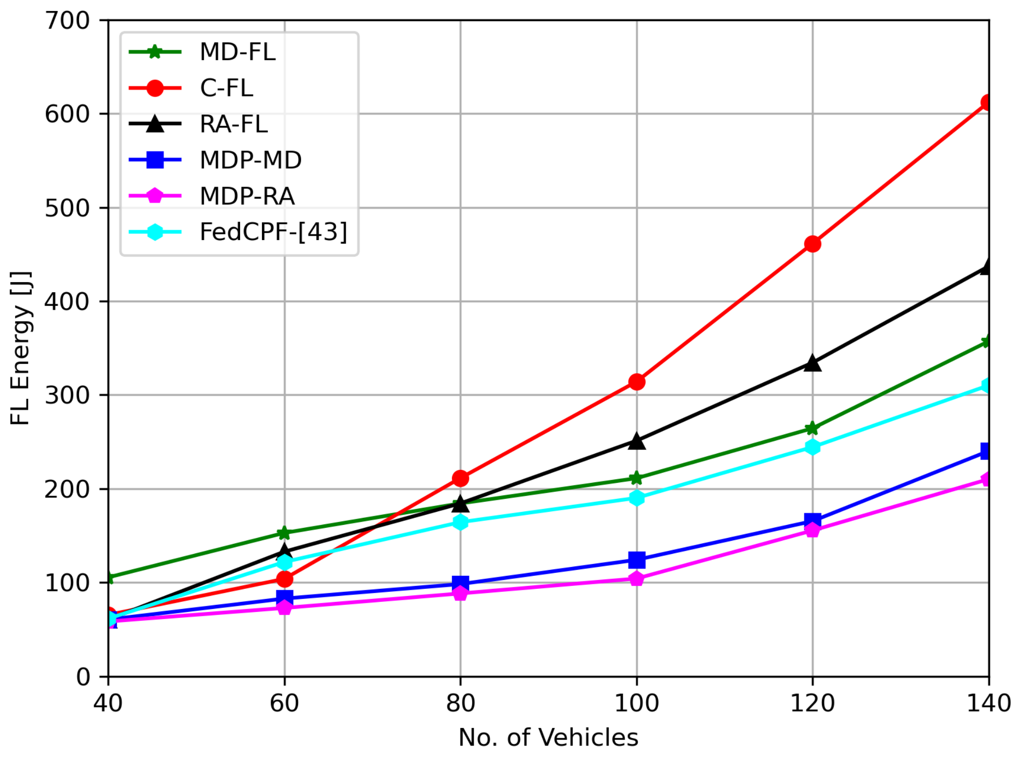


FIGURE 6.19: Performance results in terms of energy consumption for the FL process with variable number of active vehicles.

Although the C-FL method gains from a higher coverage range of the HAP node and with reduced FL latency, for a reduced number of VUs it is able to perform a large number of FL iterations with a reduced penalty, while, as the number of VUs increases, its performance decreases. For a reduced number of VUs, the penalty value for the MDP-RA process is high, mainly because of the low number of VUs participating in the FL process and its decisions to select the nearby edge nodes for reducing the overall FL cost. However, with a growing number of VUs, its performance increases with proper network selection strategies and an adequate number of VUs participating in the FL process. On the other hand, the MDP-MD method which suffers slightly in terms of latency and energy costs in the beginning can perform a high number of FL iterations reducing the LF penalty. Notice that these behaviors of MDP methods can also be impacted by the assumptions made over

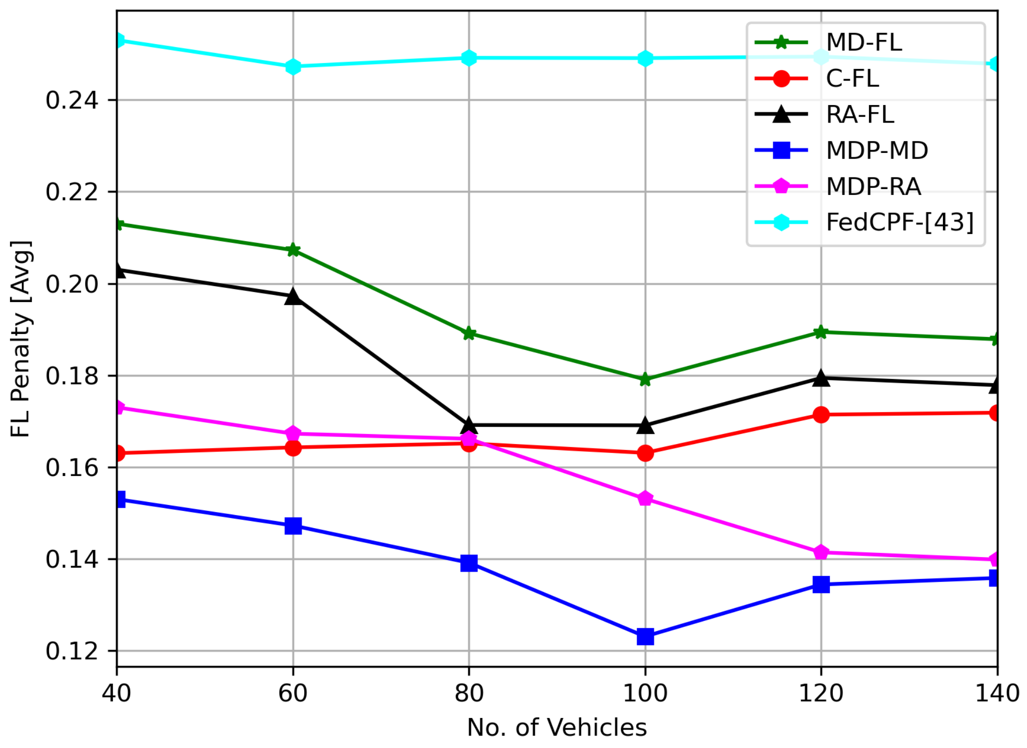


FIGURE 6.20: Performance results in terms of FL Penalty value with variable number of active vehicles.

the competing VUs decisions and can have different impacts in terms of individual costs. However, both the MDP methods are able to reduce the joint costs of latency, energy and penalties significantly compare to the traditional benchmark methods. Therefore, the proposed FL process with proper network selections can create reliable FL models with better performance.

Average Number of FL Iterations For having adequate performance FL nodes, VUs should be able to perform a sufficient number of FL iterations $(\rho\ (d(v_m, r_n, u_l, \tau_i)))$. The number of FL iterations performed by each VU is based upon the network selection strategy and the available sojourn time of the selected ENs, as given in (6.62).

A proper network selection strategy can reduce the FL iteration time. Also selecting proper ENs with a higher number of communication/computation resources allows VUs to participate in a larger number of iterations. To shade more light on the results presented in the previous figures, here we present the average number of FL iterations performed by different methods (Fig. 6.21). It can be seen that with a lower number of VUs, C-FL is able to perform a higher number of FL iterations, however, its performance reduces as more and more VUs participate in the process mainly due to the longer FL iteration time. It should also be noticed that though in the beginning, the C-FL approach can outperform one of the MDP solutions (MDP-RA), its joint performance is still not optimized due to the static FL process (Fig. 6.17). On the other hand, as described before in Fig. 6.20 the node selection strategies for the MDP-RA and MDP-MD methods are based upon a joint cost optimization and can be influenced by the competing VUs decisions. With FedCPF-inspired RSU-based benchmark solution, VUs can only perform a limited number of iterations only, mainly due to the limited coverage range of the RSU nodes. This also highlights the importance of considering the distributed NTN layers of networking platforms for supporting the FL process. With imperfect edge node selection strategies, the other two benchmark solutions (MD-FL and RA-FL), also suffer from limited FL iterations resulting in imperfect FL models with higher performance penalties (i.e., Fig. 6.20).

6.3.5 Conclusion

In this work, we have presented the communication-efficient, distributed FL platform over a joint T/NT-based VN. The proposed approach can be useful for creating cost-efficient, sustainable, and more reliable FL models for serving VUs applications. With proper analysis of the FL process cost, we formed the constrained

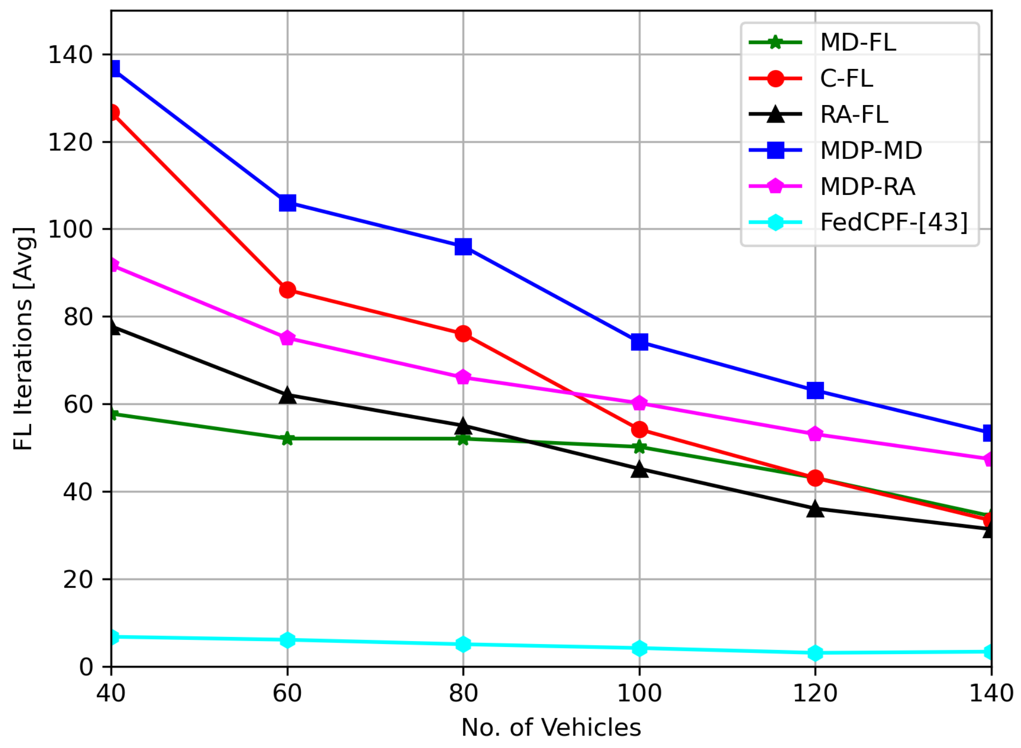


FIGURE 6.21: Performance results in terms of average number of FL iterations with variable number of active vehicles.

optimization problem for finding the optimal FL network selection strategy over multi-layered VNs. We further modeled the FL network selection problem as a sequential decision-making RL problem by adapting the MDP framework. A time-dependent environment dynamic model is created by utilizing the VUs environment parameters acquired through the V2X technology. In the end, the value iteration approach is used to solve the MDP model for finding suitable policies. The numerical results acquired over the Python-based simulation show the major advantages of the proposed FL approach over several other benchmark methods including the conventional centralized FL process. In the future, we expect to extend this work by analyzing the performance of proposed methods on realistic vehicular systems for enabling intelligent solutions at the edge.

Chapter 7

Conclusion

This dissertation highlights the importance of a future 6G enabled IoV technology and addresses two of the most challenging problems faced by vehicular users. It includes novel solutions for distributed data processing with the support of edge computing facilities enabled through joint T-NTN architecture. Next, we highlight the importance of an edge intelligence paradigm for vehicular cases and propose novel solutions for implementing distributed learning with the support of edge computing facilities in vehicular scenarios.

Several NTN layers are expected to play a key role in shaping the future wireless technology world including vehicular networks. Therefore, in this dissertation, we highlight the importance of different NTN layers and propose a novel framework for joint T-NTN-based vehicular systems with integrated edge computing facilities. This proposed architecture is then investigated to enable efficient solutions for distributed data processing and edge intelligence cases.

First, we analyze the case of terrestrial VEC systems enabled through eh RSUs and base stations. We address the problem of vehicular data processing with partial offloading case. We proposed novel RL-based solutions for the case of single-service and multi-service vehicular scenarios. In the first case of a single-service scenario, a joint network selection and offloading problem is formed to minimize the latency and energy costs from both the user and edge node sides. The problem is solved through the novel MDP framework with time-dependent state transition probabilities. A model-based value iteration algorithm is used to find the optimal policies. Next, we consider the case of a multi-service scenario and propose collaborative RL-based solutions.

The considered scenarios are then extended toward the case of a joint T-NTN framework with integrated edge computing facilities. Given the complexity of the considered scenario with heterogeneous edge computing facilities, we first propose an adaptive genetic algorithm-based meta-heuristic algorithm to address the data processing problem. Next, we evolved the considered problem to minimize the latency and energy costs and proposed HRL-based intelligent solutions. Next, we also address the problem of joint service placement, network selection, and offloading with multi-time scale decisions. In this case, we proposed multi-time scale optimization through MDP-based solutions.

Next for the case of distributed intelligence, we propose novel distributed learning frameworks with the support of joint T-NTN and edge computing facilities. We propose an FL-based solution for solving the data offloading problem in VEC systems. In this case, we take into account the limited vehicular resources and propose an optimized framework for FL and offloading processes. In particular, we proposed a joint optimization strategy for the efficient implementation of FL and offloading processes with limited vehicular resources through proper resource allocations. Next, with the support of a joint air-ground network, we proposed a distributed FL solution for enabling the intelligent vehicular system. In this work, we aimed to utilize the diverse nature of edge computing resources from different edge layers to enable an efficient FL solution for vehicular problems. In particular, we distribute the FL process over different edge layers according to the vehicular user's demands. Next, we formed an optimization problem for minimizing the FL process cost in terms of latency, energy, and performance penalty and solved it through a proper user-server allocation. We proposed MDP-based solutions for determining efficient server-selection policies based on the vehicular user's local environments.

To conclude, this dissertation highlights the key challenges faced by future IoV users in terms of distributed data processing with single/multi-service demands and edge intelligence. Several novel solutions with the support of joint T-NTN, edge computing, and distributed ML technologies are proposed, for different vehicular scenarios.

Scope for Future Work

The considered vehicular scenarios, corresponding network architecture, considered problems and proposed solutions can be evolved further in the future. For the case of distributed data processing with single/multi-service vehicular scenarios, the solutions can be evolved towards scenarios with different task dependencies. In a considered scenario, the vehicular users have a single processing task. This approach can evolve towards vehicular application demands with several processing tasks having interdependencies. In the future, it will be interesting to analyze such complex offloading cases with ML-related solutions. A multi-level data offloading scenario where user data can be processed with multiple high-level partitions can also be another future direction for a considered vehicular data processing problem.

For the case of distributed edge intelligence solutions in vehicular settings, the considered scenarios can be further evolved towards advanced distributed learning solutions with novel solutions. In recent times different learning tools such as split learning, transfer learning, etc have gained popularity. Integrating such tools with distributed learning solutions such as FL over resource-constrained vehicular networks can be an interesting future direction to be considered. Next, network socialization is another key trend that has evolved through the development of 5G systems. How to evolve the distributed learning solutions with the support of network softwarization techniques such as network slicing, software-defined networks, Network function virtualization, etc is still an open problem. The proposed distributed learning solutions can be evolved with the support of such network softwarization-related solutions to address the growing demand for intelligent solutions in vehicular networks.

References

[1] Walid Saad, Mehdi Bennis, and Mingzhe Chen. A vision of 6g wireless systems: Applications, trends, technologies, and open research problems. *IEEE network*, 34(3):134–142, 2019.

[2] Cheng-Xiang Wang, Xiaohu You, Xiqi Gao, Xiuming Zhu, Zixin Li, Chuan Zhang, Haiming Wang, Yongming Huang, Yunfei Chen, Harald Haas, et al. On the road to 6g: Visions, requirements, key technologies and testbeds. *IEEE Communications Surveys & Tutorials*, 2023.

[3] Haibo Zhou, Wenchao Xu, Jiacheng Chen, and Wei Wang. Evolutionary v2x technologies toward the internet of vehicles: Challenges and opportunities. *Proceedings of the IEEE*, 108(2):308–323, 2020.

[4] Ahmad Hammoud, Hani Sami, Azzam Mourad, Hadi Otrok, Rabeb Mizouni, and Jamal Bentahar. Ai, blockchain, and vehicular edge computing for smart and secure iov: Challenges and directions. *IEEE Internet of Things Magazine*, 3(2):68–73, 2020.

[5] Marco Giordani and Michele Zorzi. Non-terrestrial networks in the 6g era: Challenges and opportunities. *IEEE Network*, 35(2):244–251, 2020.

[6] Swapnil Sadashiv Shinde and Daniele Tarchi. Towards a novel air–ground intelligent platform for vehicular networks: Technologies, scenarios, and challenges. *Smart Cities*, 4(4):1469–1495, 2021.

[7] Ke Zhang, Yuming Mao, Supeng Leng, Yejun He, and Yan Zhang. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. 12(2):36–44, June 2017.

[8] Jun Zhang and Khaled B. Letaief. Mobile edge intelligence and computing for the Internet of Vehicles. 108(2):246–261, February 2020. doi: 10.1109/JPROC.2019.2947490.

[9] Lujie Tang, Bing Tang, Li Zhang, Feiyan Guo, and Haiwu He. Joint optimization of network selection and task offloading for vehicular edge computing. *Journal of Cloud Computing*, 10, 2021. Art. no. 23.

[10] Chao Yang, Yi Liu, Xin Chen, Weifeng Zhong, and Shengli Xie. Efficient mobility-aware task offloading for vehicular edge computing networks. *IEEE Access*, 7:26652–26664, 2019.

[11] Xiangwang Hou, Zhiyuan Ren, Jingjing Wang, Wenchi Cheng, Yong Ren, Kwang-Cheng Chen, and Hailin Zhang. Reliable computation offloading for edge-computing-enabled software-defined iov. 7(8):7097–7111, 2020. doi: 10.1109/JIOT.2020.2982292.

[12] Arash Bozorgchenani, Setareh Maghsudi, Daniele Tarchi, and Ekram Hossain. Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions. *arXiv preprint arXiv:2008.06302*, 2020.

[13] Zhenyu Zhou, Pengju Liu, Zheng Chang, Chen Xu, and Yan Zhang. Energy-efficient workload offloading and power control in vehicular edge computing. In *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 191–196, Barcelona, Spain, April 2018. doi: 10.1109/WCNCW.2018.8368975.

[14] Syed Adeel Ali Shah, Ejaz Ahmed, Muhammad Imran, and Sherali Zeadally. 5g for vehicular communications. *IEEE Communications Magazine*, 56(1): 111–117, 2018. doi: 10.1109/MCOM.2018.1700467.

[15] Juan Contreras-Castillo, Sherali Zeadally, and Juan Antonio Guerrero-Ibañez. Internet of vehicles: Architecture, protocols, and security. *IEEE Internet of Things Journal*, 5(5):3701–3709, 2018. doi: 10.1109/JIOT.2017.2690902.

[16] Surbhi Sharma and Baijnath Kaushik. A survey on internet of vehicles: Applications, security issues & solutions. *Vehicular Communications*, 20: 100182:1–100182:46, 2019. ISSN 2214-2096. doi: 10.1016/j.vehcom.2019.100182. URL `https://www.sciencedirect.com/science/article/pii/S2214209619302293`.

[17] Sarah Ali Siddiqui, Adnan Mahmood, Quan Z. Sheng, Hajime Suzuki, and Wei Ni. A survey of trust management in the internet of vehicles. *Electronics*, 10 (18):2223:1–2223:27, 2021. ISSN 2079-9292. doi: 10.3390/electronics10182223. URL `https://www.mdpi.com/2079-9292/10/18/2223`.

[18] Luca Cesarano, Andrea Croce, Leandro do C. Martins, Daniele Tarchi, and Angel A. Juan. A real-time energy-saving mechanism in internet of vehicles systems. *IEEE Access*, 2021. doi: 10.1109/ACCESS.2021.3130125. Early Access.

[19] Junhui Zhao, Qiuping Li, Yi Gong, and Ke Zhang. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(8):7944–7956, 2019. doi: 10.1109/TVT.2019.2917890.

[20] Jun Wang, Daquan Feng, Shengli Zhang, Jianhua Tang, and Tony Q. S. Quek. Computation offloading for mobile edge computing enabled vehicular networks. *IEEE Access*, 7:62624–62632, 2019. doi: 10.1109/ACCESS.2019.2915959.

[21] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, 26(3):1145–1168, 2021. doi: 10.1007/s11036-020-01624-1.

[22] Junfei Qiu, David Grace, Guoru Ding, Muhammad D Zakaria, and Qihui Wu. Air-ground heterogeneous networks for 5G and beyond via integrating high and low altitude platforms. 26(6):140–148, December 2019.

[23] Federica Rinaldi, Helka-Liina Maattanen, Johan Torsner, Sara Pizzi, Sergey Andreev, Antonio Iera, Yevgeni Koucheryavy, and Giuseppe Araniti. Non-terrestrial networks in 5g & beyond: A survey. *IEEE Access*, 8:165178–165200, 2020. doi: 10.1109/ACCESS.2020.3022981.

[24] Renchao Xie, Qinqin Tang, Qiuning Wang, Xu Liu, F. Richard Yu, and Tao Huang. Satellite-terrestrial integrated edge computing networks: Architecture, challenges, and open issues. *IEEE Network*, 34(3):224–231, 2020. doi: 10.1109/MNET.011.1900369.

[25] Fuhui Zhou, Rose Qingyang Hu, Zan Li, and Yuhao Wang. Mobile edge computing in unmanned aerial vehicle networks. *IEEE Wireless Communications*, 27(1):140–146, 2020. doi: 10.1109/MWC.001.1800594.

[26] Zhenyu Zhou, Junhao Feng, Lu Tan, Yejun He, and Jie Gong. An air-ground integration approach for mobile edge computing in iot. *IEEE Communications Magazine*, 56(8):40–47, 2018. doi: 10.1109/MCOM.2018.1701111.

[27] Hasan Ali Khattak, Haleem Farman, Bilal Jan, and Ikram Ud Din. Toward integrating vehicular clouds with iot for smart city services. *IEEE Network*, 33(2):65–71, 2019. doi: 10.1109/MNET.2019.1800236.

[28] Samad Ali, Walid Saad, Nandana Rajatheva, Kapseok Chang, Daniel Steinbach, Benjamin Sliwa, Christian Wietfeld, Kai Mei, Hamid Shiri, Hans-Jürgen Zepernick, Thi My Chinh Chu, Ijaz Ahmad, Jyrki Huusko, Jaakko Suutala, Shubhangi Bhadauria, Vimal Bhatia, Rangeet Mitra, Saidhiraj Amuru, Robert Abbas, Baohua Shao, Michele Capobianco, Guanghui Yu, Maelick Claes, Teemu Karvonen, Mingzhe Chen, Maksym Girnyk, and Hassan Malik. 6g white paper on machine learning in wireless communication networks, 2020.

[29] Yaohua Sun, Mugen Peng, Yangcheng Zhou, Yuzhe Huang, and Shiwen Mao. Application of machine learning in wireless networks: Key techniques and open issues. *IEEE Communications Surveys & Tutorials*, 21(4):3072–3108, 2019. doi: 10.1109/COMST.2019.2924243.

[30] Fengxiao Tang, Yuichi Kawamoto, Nei Kato, and Jiajia Liu. Future intelligent and secure vehicular network toward 6g: Machine-learning approaches. *Proceedings of the IEEE*, 108(2):292–307, 2020. doi: 10.1109/JPROC.2019.2954595.

[31] Nei Kato, Bomin Mao, Fengxiao Tang, Yuichi Kawamoto, and Jiajia Liu. Ten challenges in advancing machine learning technologies toward 6g. *IEEE Wireless Communications*, 27(3):96–103, 2020. doi: 10.1109/MWC.001.1900476.

[32] Zhaoyang Du, Celimuge Wu, Tsutomu Yoshinaga, Kok-Lim Alvin Yau, Yusheng Ji, and Jie Li. Federated learning for vehicular internet of things: Recent advances and open issues. *IEEE Open J. Comp. Soc.*, 1:45–61, 2020.

[33] Dongdong Ye, Rong Yu, Miao Pan, and Zhu Han. Federated learning in vehicular edge computing: A selective model aggregation approach. 8:23920–23935, 2020.

[34] Ahmet M Elbir, Burak Soner, and Sinem Coleri. Federated learning in vehicular networks. arXiv:2006.01412, 2020.

[35] Fengxiao Tang, Bomin Mao, Nei Kato, and Guan Gui. Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges. *IEEE Communications Surveys & Tutorials*, 23(3):2027–2057, 2021. doi: 10.1109/COMST.2021.3089688.

[36] Nelson Cardona, Estefanía Coronado, Steven Latré, Roberto Riggio, and Johann M. Marquez-Barja. Software-defined vehicular networking: Opportunities and challenges. *IEEE Access*, 8:219971–219995, 2020. doi: 10.1109/ACCESS.2020.3042717.

[37] Kai Wang, Hao Yin, Wei Quan, and Geyong Min. Enabling collaborative edge computing for software defined vehicular networks. *IEEE Network*, 32 (5):112–117, 2018. doi: 10.1109/MNET.2018.1700364.

[38] Edgesat - edge network computing capabilities for satellite remote terminals. URL `https://artes.esa.int/projects/edgesat`.

[39] Satis5 - demonstrator for satellite-terrestrial integration in the 5g context. URL `https://artes.esa.int/projects/satis5-0`.

[40] Expanse - leveraging big data concepts in future satcom networks. URL https://artes.esa.int/projects/expanse.

[41] 3gpp - release 17. URL https://www.3gpp.org/release-17.

[42] Rahman Imadur, Razavi Sara, Modarres, Liberg Olof, Hoymann Christian, Wiemann Henning, Tidestav Claes, Schliwa-Bertling Paul, Persson Patrik, and Gerstenberger Dirk. 5g evolution toward 5g advanced: An overview of 3gpp releases 17 and 18. *Ericsson Technology Review*, 2021. https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-evolution-toward-5g-advanced.

[43] Ning Zhang, Shan Zhang, Peng Yang, Omar Alhussein, Weihua Zhuang, and Xuemin Sherman Shen. Software defined space-air-ground integrated vehicular networks: Challenges and solutions. *IEEE Communications Magazine*, 55(7): 101–109, 2017. doi: 10.1109/MCOM.2017.1601156.

[44] Zhisheng Niu, Xuemin S. Shen, Qinyu Zhang, and Yuliang Tang. Space-air-ground integrated vehicular network for connected and automated vehicles: Challenges and solutions. *Intelligent and Converged Networks*, 1(2):142–169, 2020. doi: 10.23919/ICN.2020.0009.

[45] Changfeng Ding, Jun-Bo Wang, Hua Zhang, Min Lin, and Geoffrey Ye Li. Joint optimization of transmission and computation resources for satellite and high altitude platform assisted edge computing. *IEEE Transactions on Wireless Communications*, 2021. doi: 10.1109/TWC.2021.3103764. Early Access.

[46] Jinna Hu, Chen Chen, Lin Cai, Mohammad R. Khosravi, Qingqi Pei, and Shaohua Wan. UAV-assisted vehicular edge computing for the 6G internet of vehicles: Architecture, intelligence, and challenges. *IEEE Comm. Stand. Mag.*, 5(2):12–18, 2021. doi: 10.1109/MCOMSTD.001.2000017.

[47] Ibrahim Afolabi, Tarik Taleb, Konstantinos Samdanis, Adlen Ksentini, and Hannu Flinck. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials*, 20(3):2429–2453, 2018. doi: 10.1109/COMST.2018.2815638.

[48] Bo Yi, Xingwei Wang, Keqin Li, Sajal k. Das, and Min Huang. A comprehensive survey of network function virtualization. *Computer Networks*, 133:212–262, 2018. ISSN 1389-1286. doi: 10.1016/j.comnet.2018.01.021. URL `https://www.sciencedirect.com/science/article/pii/S1389128618300306`.

[49] Othman S. Al-Heety, Zahriladha Zakaria, Mahamod Ismail, Mohammed Mudhafar Shakir, Sameer Alani, and Hussein Alsariera. A comprehensive survey: Benefits, services, recent works, challenges, security, and use cases for sdn-vanet. *IEEE Access*, 8:91028–91047, 2020. doi: 10.1109/ACCESS.2020.2992580.

[50] Ousmane Sadio, Ibrahima Ngom, and Claude Lishou. Design and prototyping of a software defined vehicular networking. *IEEE Transactions on Vehicular Technology*, 69(1):842–850, 2020. doi: 10.1109/TVT.2019.2950426.

[51] Xiaohu Ge, Zipeng Li, and Shikuan Li. 5g software defined vehicular networks. *IEEE Communications Magazine*, 55(7):87–93, 2017. doi: 10.1109/MCOM.2017.1601144.

[52] Shiva Raj Pokhrel. Software defined internet of vehicles for automation and orchestration. *IEEE Transactions on Intelligent Transportation Systems*, 22(6):3890–3899, 2021. doi: 10.1109/TITS.2021.3077363.

[53] Adnan Mahmood, Wei Emma Zhang, and Quan Z. Sheng. Software-defined heterogeneous vehicular networking: The architectural design and open challenges. *Future Internet*, 11(3):70:1–70:17, 2019. ISSN 1999-5903. doi: 10.3390/fi11030070. URL `https://www.mdpi.com/1999-5903/11/3/70`.

[54] Claudia Campolo, Antonella Molinaro, Antonio Iera, and Francesco Menichella. 5g network slicing for vehicle-to-everything services. *IEEE Wireless Communications*, 24(6):38–45, 2017. doi: 10.1109/MWC.2017.1600408.

[55] Jie Mei, Xianbin Wang, and Kan Zheng. Intelligent network slicing for v2x services toward 5g. *IEEE Network*, 33(6):196–204, 2019. doi: 10.1109/MNET.001.1800528.

[56] Arash Bozorgchenani, Farshad Mashhadi, Daniele Tarchi, and Sergio A. Salinas Monroy. Multi-objective computation sharing in energy and delay constrained mobile edge computing environments. *IEEE Transactions on Mobile Computing*, 20(10):2992–3005, 2021. doi: 10.1109/TMC.2020.2994232.

[57] Jiao Zhang, Xiping Hu, Zhaolong Ning, Edith C.-H. Ngai, Li Zhou, Jibo Wei, Jun Cheng, Bin Hu, and Victor C. M. Leung. Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching. *IEEE Internet of Things Journal*, 6(3):4283–4294, 2019. doi: 10.1109/JIOT.2018.2875917.

[58] Ivana Kovacevic, Erkki Harjula, Savo Glisic, Beatriz Lorenzo, and Mika Ylianttila. Cloud and edge computation offloading for latency limited services. *IEEE Access*, 9:55764–55776, 2021. doi: 10.1109/ACCESS.2021.3071848.

[59] Arash Bozorgchenani, Setareh Maghsudi, Daniele Tarchi, and Ekram Hossain. Computation offloading in heterogeneous vehicular edge networks: On-line

and off-policy bandit solutions. 2021. doi: 10.1109/TMC.2021.3082927. Early Access.

[60] Baofeng Ji, Xueru Zhang, Shahid Mumtaz, Congzheng Han, Chunguo Li, Hong Wen, and Dan Wang. Survey on the Internet of Vehicles: Network architectures and applications. *IEEE Comm. Stand. Mag.*, 4(1):34–41, March 2020. doi: 10.1109/MCOMSTD.001.1900053.

[61] Mashael Khayyat, Ibrahim A. Elgendy, Ammar Muthanna, Abdullah S. Alshahrani, Soltan Alharbi, and Andrey Koucheryavy. Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. *IEEE Access*, 8:137052–137062, 2020. doi: 10.1109/ACCESS.2020.3011705.

[62] H. Zhou, W. Xu, J. Chen, and W. Wang. Evolutionary V2X technologies toward the internet of vehicles: Challenges and opportunities. *Proceedings of the IEEE*, 108(2):308–323, 2020. doi: 10.1109/JPROC.2019.2961937.

[63] Leandro do C. Martins, Daniele Tarchi, Angel A. Juan, and Alessandro Fusco. Agile optimization for a real-time facility location problem in internet of vehicles networks. *Networks*, 2021. doi: 10.1002/net.22067. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/net.22067`. Early View.

[64] Chia-Hung Lin, Yu-Chien Lin, Yen-Jung Wu, Wei-Ho Chung, and Ta-Sung Lee. A survey on deep learning-based vehicular communication applications. *Journal of Signal Processing Systems*, 93(4):369–388, 2021. doi: 10.1007/s11265-020-01587-2.

[65] Jason Posner, Lewis Tseng, Moayad Aloqaily, and Yaser Jararweh. Federated learning in vehicular networks: Opportunities and solutions. *IEEE Network*, 35(2):152–159, 2021. doi: 10.1109/MNET.011.2000430.

[66] Steve Chukwuebuka Arum, David Grace, and Paul Daniel Mitchell. A review of wireless communication using high-altitude platforms for extended coverage and capacity. *Computer Communications*, 157:232–256, 2020. ISSN 0140-3664. doi: 10.1016/j.comcom.2020.04.020. URL `https://www.sciencedirect.com/science/article/pii/S0140366419313143`.

[67] ITU-R. Methodology for determining the power level for high altitude platform stations ground terminals to facilitate sharing with space station receivers in the bands 47.2-47.5 ghz and 47.9-48.2 ghz. Recommendation SF.1843, ITU-R, 2007.

[68] Swapnil Sadashiv Shinde and Daniele Tarchi. A markov decision process solution for energy-saving network selection and computation offloading in vehicular networks. 2023. Early access, doi:10.1109/TVT.2023.3264504.

[69] Swapnil Sadashiv Shinde and Daniele Tarchi. Collaborative reinforcement learning for multi-service Internet of Vehicles. 10(3):2589–2602, February 2023. doi: 10.1109/JIOT.2022.3213993.

[70] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5G. White Paper 11, ETSI, September 2015. URL `https://www.etsi.org/images/files/etsiwhitepapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf`.

[71] Alisson Barbosa De Souza, Paulo A. L. Rego, Tiago Carneiro, Jardel Das C. Rodrigues, Pedro Pedrosa Rebouças Filho, José Neuman De Souza, Vinay

Chamola, Victor Hugo C. De Albuquerque, and Biplab Sikdar. Computation offloading for vehicular environments: A survey. *IEEE Access*, 8:198214–198243, 2020. doi: 10.1109/ACCESS.2020.3033828.

[72] Mushu Li, Jie Gao, Lian Zhao, and Xuemin Shen. Deep reinforcement learning for collaborative edge computing in vehicular networks. 6(4):1122–1135, December 2020. doi: 10.1109/TCCN.2020.3003036.

[73] Shupeng Wang, Jun Li, Guangjun Wu, Handi Chen, and Shihui Sun. Joint optimization of task offloading and resource allocation based on differential privacy in vehicular edge computing. 9(1):109–119, February 2022. doi: 10.1109/TCSS.2021.3074949.

[74] Huizi Xiao, Jun Zhao, Qingqi Pei, Jie Feng, Lei Liu, and Weisong Shi. Vehicle selection and resource optimization for federated learning in vehicular edge computing. 23(8):11073–11087, August 2022. doi: 10.1109/TITS.2021.3099597.

[75] Rahul Yadav, Weizhe Zhang, Omprakash Kaiwartya, Houbing Song, and Shui Yu. Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing. 69(12):14198–14211, December 2020. doi: 10.1109/TVT.2020.3040596.

[76] Swapnil Sadashiv Shinde, Arash Bozorgchenani, Daniele Tarchi, and Qiang Ni. On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems. 71(2):2041–2057, February 2022.

[77] Swapnil Sadashiv Shinde and Daniele Tarchi. Collaborative reinforcement learning for multi-service internet of vehicles. 10(3):2589–2602, February 2023. doi: 10.1109/JIOT.2022.3213993.

[78] Swapnil Sadashiv Shinde and Daniele Tarchi. Network selection and computation offloading in non-terrestrial network edge computing environments for vehicular applications. In *2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, Graz, Austria, September 2022. doi: 10.1109/ASMS/SPSC55670.2022.9914757.

[79] Yujiong Liu, Shangguang Wang, Qinglin Zhao, Shiyu Du, Ao Zhou, Xiao Ma, and Fangchun Yang. Dependency-aware task scheduling in vehicular edge computing. 7(6):4961–4971, June 2020. doi: 10.1109/JIOT.2020.2972041.

[80] Arash Bozorgchenani, Daniele Tarchi, and Giovanni Emanuele Corazza. Mobile edge computing partial offloading techniques for mobile urban scenarios. In *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, December 2018. doi: 10.1109/GLOCOM.2018.8647240.

[81] Yang Liu, Jianshan Zhou, Daxin Tian, Zhengguo Sheng, Xuting Duan, Guixian Qu, and Victor C. M. Leung. Joint communication and computation resource scheduling of a UAV-assisted mobile edge computing system for platooning vehicles. 23(7):8435–8450, July 2022. doi: 10.1109/TITS.2021.3082539.

[82] Mohammad Nekoui and Hossein Pishro-nik. Fundamental tradeoffs in vehicular ad hoc networks. In *Proceedings of the Seventh ACM International Workshop on VehiculAr InterNETworking*, page 91–96, Chicago, Illinois, USA, September 2010. ISBN 9781450301459.

[83] Kai Xiong, Supeng Leng, Xiaosha Chen, Chongwen Huang, Chau Yuen, and Yong Liang Guan. Communication and computing resource optimization for

connected autonomous driving. 69(11):12652–12663, 2020. doi: 10.1109/TVT.2020.3029109.

[84] Xiaopeng Mo and Jie Xu. Energy-efficient federated edge learning with joint communication and computation design. *Journal of Communications and Information Networks*, 6(2):110, June 2021. doi: j.issn.2096-1081.2021.02.02.

[85] Jie Zhang, Hongzhi Guo, Jiajia Liu, and Yanning Zhang. Task offloading in vehicular edge computing networks: A load-balancing solution. 69(2):2092–2104, February 2020. doi: 10.1109/TVT.2019.2959410.

[86] Arash Bozorgchenani, Setareh Maghsudi, Daniele Tarchi, and Ekram Hossain. Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions. 21(12):4233–4248, December 2022. doi: 10.1109/TMC.2021.3082927.

[87] Luca Cesarano, Andrea Croce, Leandro Do Carmo Martins, Daniele Tarchi, and Angel A. Juan. A real-time energy-saving mechanism in Internet of Vehicles systems. 9:157842–157858, 2021. doi: 10.1109/ACCESS.2021.3130125.

[88] Guisong Yang, Ling Hou, Xingyu He, Daojing He, Sammy Chan, and Mohsen Guizani. Offloading time optimization via markov decision process in mobile-edge computing. 8(4):2483–2493, February 2021. doi: 10.1109/JIOT.2020.3033285.

[89] Xuefei Zhang, Jian Zhang, Zhitong Liu, Qimei Cui, Xiaofeng Tao, and Shuo Wang. MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities. 69(3):3296–3309, March 2020. doi: 10.1109/TVT.2020.2965159.

[90] Nikhil Balaji, Stefan Kiefer, Petr Novotnỳ, Guillermo A Pérez, and Mahsa Shirmohammadi. On the complexity of value iteration. *arXiv:1807.04920*, 2018. doi:10.48550/arXiv.1807.04920.

[91] Zhaolong Ning, Peiran Dong, Xiaojie Wang, Joel JPC Rodrigues, and Feng Xia. Deep reinforcement learning for vehicular edge computing: An intelligent offloading system. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(6):1–24, November 2019. Art. no. 60.

[92] Yueyue Dai, Du Xu, Sabita Maharjan, and Yan Zhang. Joint load balancing and offloading in vehicular edge computing and networks. *IEEE Internet of Things Journal*, 6(3):4377–4387, 2018.

[93] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv:1706.05296*, 2017. doi:10.48550/arXiv.1706.05296.

[94] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob N Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21:1–51, 2020. Art. no. 178.

[95] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[96] Yi Liu, Huimin Yu, Shengli Xie, and Yan Zhang. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. 68(11):11158–11168, November 2019.

[97] Hongzhi Guo, Jiajia Liu, Ju Ren, and Yanning Zhang. Intelligent task offloading in vehicular edge computing networks. *IEEE Wireless Communications*, 27(4):126–132, 2020. doi: 10.1109/MWC.001.1900489.

[98] Xuefei Zhang, Jian Zhang, Zhitong Liu, Qimei Cui, Xiaofeng Tao, and Shuo Wang. MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities. *IEEE Transactions on Vehicular Technology*, 69(3):3296–3309, 2020. doi: 10.1109/TVT.2020.2965159.

[99] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

[100] Helin Yang, Zehui Xiong, Jun Zhao, Dusit Niyato, Liang Xiao, and Qingqing Wu. Deep reinforcement learning-based intelligent reflecting surface for secure wireless communications. *IEEE Transactions on Wireless Communications*, 20(1):375–388, 2021. doi: 10.1109/TWC.2020.3024860.

[101] Zhaolong Ning, Peiran Dong, Xiaojie Wang, Lei Guo, Joel J. P. C. Rodrigues, Xiangjie Kong, Jun Huang, and Ricky Y. K. Kwok. Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme. *IEEE Transactions on Cognitive Communications and Networking*, 5(4):1060–1072, 2019. doi: 10.1109/TCCN.2019.2930521.

[102] Marco Giordani and Michele Zorzi. Non-terrestrial networks in the 6G era: Challenges and opportunities. 35(2):244–251, 2021. doi: 10.1109/MNET.011.2000493.

[103] Alessandro Traspadini, Marco Giordani, and Michele Zorzi. UAV/HAP-assisted vehicular edge computing in 6G: Where and what to offload? In *2022*

Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), pages 178–183, Grenoble, France, June 2022. doi: 10.1109/EuCNC/6GSummit54941.2022.9815734.

[104] Jianan Sun, Qing Gu, Tao Zheng, Ping Dong, Alvin Valera, and Yajuan Qin. Joint optimization of computation offloading and task scheduling in vehicular edge computing networks. 8:10466–10477, 2020. doi: 10.1109/ACCESS.2020.2965620.

[105] Huizi Xiao, Jun Zhao, Qingqi Pei, Jie Feng, Lei Liu, and Weisong Shi. Vehicle selection and resource optimization for federated learning in vehicular edge computing. 2021. doi: 10.1109/TITS.2021.3099597. early access. doi: 10.1109/TITS.2021.3099597.

[106] Qingqing Tang, Zesong Fei, Bin Li, and Zhu Han. Computation offloading in LEO satellite networks with hybrid cloud and edge computing. 8(11):9164–9176, 2021. doi: 10.1109/JIOT.2021.3056569.

[107] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing.* Springer, Berlin, Heidelberg, 2 edition, 2015.

[108] Sangwon Hwang, Hanjin Kim, Hoon Lee, and Inkyu Lee. Multi-agent deep reinforcement learning for distributed resource management in wirelessly powered communication networks. 69(11):14055–14060, 2020. doi: 10.1109/TVT.2020.3029609.

[109] Tao Ren, Jianwei Niu, Bin Dai, Xuefeng Liu, Zheyuan Hu, Mingliang Xu, and Mohsen Guizani. Enabling efficient scheduling in large-scale UAV-assisted mobile-edge computing via hierarchical reinforcement learning. 9(10):7095–7109, 2022. doi: 10.1109/JIOT.2021.3071531.

[110] Shengheng Liu, Chong Zheng, Yongming Huang, and Tony Q. S. Quek. Distributed reinforcement learning for privacy-preserving dynamic edge caching. 40(3):749–760, 2022. doi: 10.1109/JSAC.2022.3142348.

[111] Xiaohui Gu and Guoan Zhang. Energy-efficient computation offloading for vehicular edge computing networks. *Computer Communications*, 166:244–253, 2021.

[112] Arash Bozorgchenani, Daniele Tarchi, and Walter Cerroni. On-demand service deployment strategies for Fog-as-a-Service scenarios. 25(5):1500–1504, May 2021. doi: 10.1109/LCOMM.2021.3055535.

[113] Le Thanh Tan and Rose Qingyang Hu. Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning. 67(11):10190–10203, November 2018. doi: 10.1109/TVT.2018.2867191.

[114] Swapnil Sadashiv Shinde and Daniele Tarchi. Joint air-ground distributed federated learning for intelligent transportation systems. 2023. Early access, doi:10.1109/TITS.2023.3265416.

[115] Tao Ouyang, Zhi Zhou, and Xu Chen. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. 36(10):2333–2345, October 2018. doi: 10.1109/JSAC.2018.2869954.

[116] Sun Mao, Shunfan He, and Jinsong Wu. Joint UAV position optimization and resource scheduling in space-air-ground integrated networks with mixed cloud-edge computing. 15(3):3992–4002, September 2021. doi: 10.1109/JSYST.2020.3041706.

[117] Hyeong Soo Chang, P.J. Fard, S.I. Marcus, and M. Shayman. Multitime scale markov decision processes. 48(6):976–987, June 2003. doi: 10.1109/TAC.2003.812782.

[118] Zhenyu Zhou, Junhao Feng, Zheng Chang, and Xuemin Shen. Energy-efficient edge computing service provisioning for vehicular networks: A consensus ADMM approach. 68(5):5087–5099, May 2019.

[119] Shichao Li, Siyu Lin, Lin Cai, Wenjie Li, and Gang Zhu. Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing. 69(3):3384–3398, March 2020.

[120] Qiqi Ren, Omid Abbasi, Gunes Karabulut Kurt, Halim Yanikomeroglu, and Jian Chen. Caching and computation offloading in high altitude platform station (HAPS) assisted intelligent transportation systems, 2021. arXiv:2106.14928.

[121] Arash Bozorgchenani, Farshad Mashhadi, Daniele Tarchi, and Sergio Salinas. Multi-objective computation sharing in energy and delay constrained mobile edge computing environments. 20(10):2992–3005, October 2021. doi: 10.1109/TMC.2020.2994232.

[122] Mingzhe Chen, Deniz Gündüz, Kaibin Huang, Walid Saad, Mehdi Bennis, Aneta Vulgarakis Feljan, and H. Vincent Poor. Distributed learning in wireless networks: Recent progress and future challenges. arXiv:2104.02151, 2021.

[123] Zhaohui Yang, Mingzhe Chen, Walid Saad, Choong Seon Hong, and Mohammad Shikh-Bahaei. Energy efficient federated learning over wireless communication networks. 20(3):1935–1949, March 2021. doi: 10.1109/TWC.2020.3037554.

[124] Canh T. Dinh, Nguyen H. Tran, Minh N. H. Nguyen, Choong Seon Hong, Wei Bao, Albert Y. Zomaya, and Vincent Gramoli. Federated learning over wireless networks: Convergence analysis and resource allocation. 29(1):398–409, February 2021. doi: 10.1109/TNET.2020.3035770.

[125] Hongda Wu and Ping Wang. Fast-convergent federated learning with adaptive weighting. 2021. doi: 10.1109/TCCN.2021.3084406. Early Access.

[126] Mingzhe Chen, Zhaohui Yang, Walid Saad, Changchuan Yin, H. Vincent Poor, and Shuguang Cui. A joint learning and communications framework for federated learning over wireless networks. 20(1):269–283, January 2021. doi: 10.1109/TWC.2020.3024629.

[127] N.C. Sagias, G.S. Tombras, and G.K. Karagiannidis. New results for the shannon channel capacity in generalized fading channels. 9(2):97–99, February 2005. doi: 10.1109/LCOMM.2005.02031.

[128] Gunes Karabulut Kurt, Mohammad G Khoshkholgh, Safwan Alfattani, Ahmed Ibrahim, Tasneem SJ Darwish, Md Sahabul Alam, Halim Yanikomeroglu, and Abbas Yongacoglu. A vision and framework for the high altitude platform station (HAPS) networks of the future. 23(2):729–779, 2021.

[129] Wenhan Zhan, Chunbo Luo, Jin Wang, Chao Wang, Geyong Min, Hancong Duan, and Qingxin Zhu. Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing. 7(6):5449–5465, June 2020. doi: 10.1109/JIOT.2020.2978830.

[130] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Hoboken, NJ, USA, 2005.

[131] Su Liu, Jiong Yu, Xiaoheng Deng, and Shaohua Wan. FedCPF: An efficient-communication federated learning approach for vehicular edge computing in 6G communication networks. 23(2):1616–1629, February 2022. doi: 10.1109/TITS.2021.3099368.

[132] Mashael Khayyat, Ibrahim A. Elgendy, Ammar Muthanna, Abdullah S. Alshahrani, Soltan Alharbi, and Andrey Koucheryavy. Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. *IEEE Access*, 8:137052–137062, 2020. doi: 10.1109/ACCESS.2020.3011705.

[133] Nancy Lyons and George Lăzăroiu. Addressing the COVID-19 crisis by harnessing internet of things sensors and machine learning algorithms in data-driven smart sustainable cities. *Geopolitics, History, and International Relations*, 12(2):65–71, 2020. ISSN 19489145, 23744383.

[134] Agachai Sumalee and Hung Wai Ho. Smarter and more connected: Future intelligent transportation system. *IATSS Research*, 42(2):67–71, 2018.

[135] Farshad Mashhadi, Sergio A Salinas Monroy, Arash Bozorgchenani, and Daniele Tarchi. Optimal auction for delay and energy constrained task offloading in mobile edge computing. *Computer Networks*, 183, 2020. Art. no. 107527.

[136] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. 21(4):3133–3174, Fourth quarter 2019.

[137] Deniz Gündüz, Paul de Kerret, Nicholas D Sidiropoulos, David Gesbert, Chandra R Murthy, and Mihaela van der Schaar. Machine learning in the air. 37 (10):2184–2199, October 2019.

[138] He Fang, Xianbin Wang, and Stefano Tomasin. Machine learning for intelligent authentication in 5G and beyond wireless networks. 26(5):55–61, October 2019.

[139] Manuel Eugenio Morocho-Cayamcela, Haeyoung Lee, and Wansu Lim. Machine learning for 5G/B5G mobile and wireless communications: Potential, limitations, and future directions. 7:137184–137206, 2019.

[140] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[141] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[142] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. URL `https://arxiv.org/abs/1610.05492`.

[143] Xianbin Cao, Peng Yang, Mohamed Alzenad, Xing Xi, Dapeng Wu, and Halim Yanikomeroglu. Airborne communication networks: A survey. 36(9):1907–1926, September 2018.

[144] Yohei Shibata, Noboru Kanazawa, Mitsukuni Konishi, Kenji Hoshino, Yoshichika Ohta, and Atsushi Nagate. System design of gigabit HAPS mobile communications. *IEEE Access*, 8:157995–158007, 2020.

[145] Salman Raza, Wei Liu, Manzoor Ahmed, Muhammad Rizwan Anwar, Muhammad Ayzed Mirza, Qibo Sun, and Shangguang Wang. An efficient task offloading scheme in vehicular edge computing. *Journal of Cloud Computing*, 9:1–14, 2020.

[146] Sihua Wang, Mingzhe Chen, Changchuan Yin, Walid Saad, Choong Seon Hong, Shuguang Cui, and H. Vincent Poor. Federated learning for task and resource allocation in wireless high altitude balloon networks. 2021. doi: 10.1109/JIOT.2021.3080078. Early Access.

[147] Mohammad Mohammadi Amiri, Deniz Gündüz, Sanjeev R. Kulkarni, and H. V. Poor. Update aware device scheduling for federated learning at the wireless edge. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2598–2603, Los Angeles, CA, USA, June 2020. doi: 10.1109/ISIT44484.2020.9173960.

[148] A. Bozorgchenani, D. Tarchi, and G. E. Corazza. Centralized and distributed architectures for energy and delay efficient fog network-based edge computing services. *IEEE Transactions on Green Communications and Networking*, 3(1): 250–263, 2019. doi: 10.1109/TGCN.2018.2885443.

[149] Hao Ye, Le Liang, Geoffrey Ye Li, JoonBeom Kim, Lu Lu, and May Wu. Machine learning for vehicular networks: Recent advances and application examples. 13(2):94–101, June 2018.

[150] Pengju Liu, Junluo Li, and Zhongwei Sun. Matching-based task offloading for vehicular edge computing. *IEEE Access*, 7:27628–27640, 2019.

[151] Kai Xiong, Supeng Leng, Chongwen Huang, Chau Yuen, and Yong Liang Guan. Intelligent task offloading for heterogeneous v2x communications. *IEEE Transactions on Intelligent Transportation Systems*, 22(4):2226–2238, 2020.

[152] Ali Shakarami, Mostafa Ghobaei-Arani, and Ali Shahidinejad. A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. *Computer Networks*, page 107496, 2020.

[153] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *CoRR*, abs/1811.11479, 2018. URL `http://arxiv.org/abs/1811.11479`.

[154] Wugedele Bao, Celimuge Wu, Siri Guleng, Jiefang Zhang, Kok-Lim Alvin Yau, and Yusheng Ji. Edge computing-based joint client selection and networking scheme for federated learning in vehicular IoT. *China Communications*, 18(6): 39–52, June 2021.

[155] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, 26:1145–1168, 2021.

[156] Rudzidatul Akmam Dziyauddin, Dusit Niyato, Nguyen Cong Luong, Ahmad Ariff Aizuddin Mohd Atan, Mohd Azri Mohd Izhar, Marwan Hadri Azmi, and Salwani Mohd Daud. Computation offloading and content caching and delivery in vehicular edge network: A survey. Art. no. 108228, 2021.

[157] Zhengxin Yu, Jia Hu, Geyong Min, Zhiwei Zhao, Wang Miao, and M Shamim Hossain. Mobility-aware proactive edge caching for connected vehicles using federated learning. 22(8):5341–5351, August 2021. doi: 10.1109/TITS.2020. 3017474.

[158] Xin Li, Yifan Dang, Mohammad Aazam, Xia Peng, Tefang Chen, and Chunyang Chen. Energy-efficient computation offloading in vehicular edge cloud computing. 8:37632–37644, 2020.

[159] Rahul Yadav, Weizhe Zhang, Omprakash Kaiwartya, Houbing Song, and Shui Yu. Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing. 69(12):14198–14211, December 2020.

[160] Shuai Yu, Xiaowen Gong, Qian Shi, Xiaofei Wang, and Xu Chen. EC-SAGINs: Edge computing-enhanced space-air-ground integrated networks for internet of vehicles. 2021. doi: 10.1109/JIOT.2021.3052542. Early access.

[161] Chao Sun, Wei Ni, and Xin Wang. Joint computation offloading and trajectory planning for UAV-assisted edge computing. 20(8):5343–5358, August 2021. doi: 10.1109/TWC.2021.3067163.

[162] Martin Isaksson and Karl Norrman. Secure federated learning in 5G mobile networks. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, December 2020. doi: 10.1109/GLOBECOM42002.2020.9322479.

[163] Weifeng Gao, Zhiwei Zhao, Geyong Min, Qiang Ni, and Yuhong Jiang. Resource allocation for latency-aware federated learning in industrial internet of things. 17(12):8505–8513, December 2021. doi: 10.1109/TII.2021.3073642.

[164] Zhenjiang Zhang, Wenyu Zhang, and Fan-Hsun Tseng. Satellite mobile edge computing: Improving qos of high-speed satellite-terrestrial networks using edge computing techniques. *IEEE network*, 33(1):70–76, 2019.

[165] Swapnil Sadashiv Shinde, Dania Marabissi, and Daniele Tarchi. A network operator-biased approach for multi-service network function placement

in a 5g network slicing architecture. *Computer Networks*, 201(108598): 108598:1–108598:14, 2021. ISSN 1389-1286. doi: 10.1016/j.comnet.2021. 108598. URL `https://www.sciencedirect.com/science/article/pii/S1389128621004989`.

[166] Technical Specification Group Services and System Aspects. Study on enhancement of support for edge computing in 5g core network (5gc). Technical Report 23.748 v17.0.0, 3GPP, 2020.

[167] Zaib Ullah, Fadi Al-Turjman, Leonardo Mostarda, and Roberto Gagliardi. Applications of artificial intelligence and machine learning in smart cities. *Computer Communications*, 154:313–323, 2020. ISSN 0140-3664. doi: 10.1016/j.comcom.2020.02.069. URL `https://www.sciencedirect.com/science/article/pii/S0140366419320821`.

[168] Latif U. Khan, Ibrar Yaqoob, Nguyen H. Tran, S. M. Ahsan Kazmi, Tri Nguyen Dang, and Choong Seon Hong. Edge-computing-enabled smart cities: A comprehensive survey. *IEEE Internet of Things Journal*, 7(10):10200–10232, 2020. doi: 10.1109/JIOT.2020.2987070.

[169] Leonardo Guevara and Fernando Auat Cheein. The role of 5g technologies: Challenges in smart cities and intelligent transportation systems. *Sustainability*, 12(16):6469, 2020.

[170] Hamid Menouar, Ismail Guvenc, Kemal Akkaya, A Selcuk Uluagac, Abdullah Kadri, and Adem Tuncer. Uav-enabled intelligent transportation systems for the smart city: Applications and challenges. *IEEE Communications Magazine*, 55(3):22–28, 2017.

[171] Jingjing Wang, Chunxiao Jiang, Kai Zhang, Tony QS Quek, Yong Ren, and Lajos Hanzo. Vehicular sensing networks in a smart city: Principles, technologies and applications. *IEEE Wireless Communications*, 25(1):122–132, 2017.

[172] Kai Wang, Hao Yin, Wei Quan, and Geyong Min. Enabling collaborative edge computing for software defined vehicular networks. *IEEE Network*, 32 (5):112–117, 2018. doi: 10.1109/MNET.2018.1700364.

[173] Ousmane Sadio, Ibrahima Ngom, and Claude Lishou. Design and prototyping of a software defined vehicular networking. *IEEE Transactions on Vehicular Technology*, 69(1):842–850, 2020. doi: 10.1109/TVT.2019.2950426.

[174] Bo Yang, Xuelin Cao, Joshua Bassey, Xiangfang Li, and Lijun Qian. Computation offloading in multi-access edge computing: A multi-task learning approach. 20(9):2745–2762, September 2021. doi: 10.1109/TMC.2020.2990630.

[175] Bo Yang, Xuelin Cao, Kai Xiong, Chau Yuen, Yong Liang Guan, Supeng Leng, Lijun Qian, and Zhu Han. Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions. 28(2):40–47, April 2021. doi: 10.1109/MWC.001.2000292.

[176] Jianan Sun, Qing Gu, Tao Zheng, Ping Dong, Alvin Valera, and Yajuan Qin. Joint optimization of computation offloading and task scheduling in vehicular edge computing networks. *IEEE Access*, 8:10466–10477, 2020. doi: 10.1109/ACCESS.2020.2965620.

[177] Peng Qin, Yang Fu, Guoming Tang, Xiongwen Zhao, and Suiyan Geng. Learning based energy efficient task offloading for vehicular collaborative edge computing. 71(8):8398–8413, August 2022. doi: 10.1109/TVT.2022.3171344.

[178] Wenhao Fan, Jie Liu, Mingyu Hua, Fan Wu, and Yuan'an Liu. Joint task offloading and resource allocation for multi-access edge computing assisted by parked and moving vehicles. 71(5):5314–5330, May 2022. doi: 10.1109/TVT.2022.3149937.

[179] M. Nadeem Ahangar, Qasim Z. Ahmed, Fahd A. Khan, and Maryam Hafeez. A survey of autonomous vehicles: Enabling communication technologies and challenges. *Sensors*, 21(3), 2021. ISSN 1424-8220. doi: 10.3390/s21030706.

[180] Adel A Ahmed and Ahmad A Alzahrani. A comprehensive survey on handover management for vehicular ad hoc network based on 5g mobile networks technology. *Transactions on Emerging Telecommunications Technologies*, 30 (3):e3546, 2019.

[181] Zhaolong Ning, Peiran Dong, Xiaojie Wang, Xiping Hu, Jiangchuan Liu, Lei Guo, Bin Hu, Ricky Kwok, and Victor C. M. Leung. Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks. 21 (4):1319–1333, April 2022. doi: 10.1109/TMC.2020.3025116.

[182] Dinh C. Nguyen, Ming Ding, Pubudu N. Pathirana, Aruna Seneviratne, Jun Li, Dusit Niyato, Octavia Dobre, and H. Vincent Poor. 6G Internet of Things: A comprehensive survey. 9(1):359–383, January 2022. doi: 10.1109/JIOT.2021.3103320.

[183] Yueyue Dai, Du Xu, Sabita Maharjan, and Yan Zhang. Joint load balancing and offloading in vehicular edge computing and networks. 6(3):4377–4387, June 2019. doi: 10.1109/JIOT.2018.2876298.

[184] Yuxuan Sun, Xueying Guo, Jinhui Song, Sheng Zhou, Zhiyuan Jiang, Xin Liu, and Zhisheng Niu. Adaptive learning-based task offloading for vehicular edge

computing systems. 68(4):3061–3074, April 2019. doi: 10.1109/TVT.2019.2895593.

[185] Zhenyu Zhou, Junhao Feng, Zheng Chang, and Xuemin Shen. Energy-efficient edge computing service provisioning for vehicular networks: A consensus admm approach. 68(5):5087–5099, May 2019. doi: 10.1109/TVT.2019.2905432.

[186] Xiaojie Wang, Zhaolong Ning, Song Guo, and Lei Wang. Imitation learning enabled task scheduling for online vehicular edge computing. 21(2):598–611, February 2022. doi: 10.1109/TMC.2020.3012509.

[187] Yang Liu, Jianshan Zhou, Daxin Tian, Zhengguo Sheng, Xuting Duan, Guixian Qu, and Victor C. M. Leung. Joint communication and computation resource scheduling of a uav-assisted mobile edge computing system for platooning vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):8435–8450, 2022. doi: 10.1109/TITS.2021.3082539.

[188] Junhui Zhao, Qiuping Li, Yi Gong, and Ke Zhang. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(8):7944–7956, 2019. doi: 10.1109/TVT.2019.2917890.

[189] Quyuan Luo, Changle Li, Tom Luan, and Weisong Shi. Minimizing the delay and cost of computation offloading for vehicular edge computing. *IEEE Transactions on Services Computing*, pages 1–1, 2021. doi: 10.1109/TSC.2021.3064579.

[190] Qiang Ye, Weisen Shi, Kaige Qu, Hongli He, Weihua Zhuang, and Xuemin Shen. Joint RAN slicing and computation offloading for autonomous vehicular

networks: A learning-assisted hierarchical approach. *IEEE Open Journal of Vehicular Technology*, 2:272–288, 2021. doi: 10.1109/OJVT.2021.3089083.

[191] Zhaolong Ning, Kaiyuan Zhang, Xiaojie Wang, Lei Guo, Xiping Hu, Jun Huang, Bin Hu, and Ricky Y. K. Kwok. Intelligent edge computing in internet of vehicles: A joint computation offloading and caching solution. 22(4): 2212–2225, April 2021. doi: 10.1109/TITS.2020.2997832.

[192] Shichao Li, Siyu Lin, Lin Cai, Wenjie Li, and Gang Zhu. Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing. 69(3):3384–3398, March 2020. doi: 10.1109/TVT.2020.2967882.

[193] Yueyue Dai, Du Xu, Sabita Maharjan, and Yan Zhang. Joint load balancing and offloading in vehicular edge computing and networks. 6(3):4377–4387, June 2019. doi: 10.1109/JIOT.2018.2876298.

[194] Yueyue Dai, Du Xu, Sabita Maharjan, and Yan Zhang. Joint computation offloading and user association in multi-task mobile edge computing. 67(12): 12313–12325, December 2018. doi: 10.1109/TVT.2018.2876804.

[195] Weiyang Feng, Siyu Lin, Ning Zhang, Gongpu Wang, Bo Ai, and Lin Cai. Joint C-V2X based offloading and resource allocation in multi-tier vehicular edge computing system. 41(2):432–445, 2023. doi: 10.1109/JSAC.2022.3227081.

[196] Ying He, Yuhang Wang, Qiuzhen Lin, and Jianqiang Li. Meta-hierarchical reinforcement learning (MHRL)-based dynamic resource allocation for dynamic vehicular networks. 71(4):3495–3506, April 2022. doi: 10.1109/TVT.2022.3146439.

[197] Jinming Shi, Jun Du, Jian Wang, and Jian Yuan. Deep reinforcement learning-based V2V partial computation offloading in vehicular fog computing. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, Nanjing, China, March-April 2021. doi: 10.1109/WCNC49053.2021.9417450.

[198] Xuan-Qui Pham, Thien Huynh-The, Eui-Nam Huh, and Dong-Seong Kim. Partial computation offloading in parked vehicle-assisted multi-access edge computing: A game-theoretic approach. 71(9):10220–10225, September 2022. doi: 10.1109/TVT.2022.3182378.

[199] Xiaoyu Zhu, Yueyi Luo, Anfeng Liu, Md Zakirul Alam Bhuiyan, and Shaobo Zhang. Multiagent deep reinforcement learning for vehicular computation offloading in IoT. 8(12):9763–9773, June 2021. doi: 10.1109/JIOT.2020.3040768.

[200] Lei Liu, Ming Zhao, Miao Yu, Mian Ahmad Jan, Dapeng Lan, and Amirhosein Taherkordi. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. 24(2):2169–2182, February 2023. doi: 10.1109/TITS.2022.3142566.

[201] Fengxiao Tang, Bomin Mao, Nei Kato, and Guan Gui. Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges. 23(3):2027–2057, Third Quarter 2021. doi: 10.1109/COMST.2021.3089688.

[202] Dairu Han, Qiang Ye, Haixia Peng, Wen Wu, Huaqing Wu, Wenhe Liao, and Xuemin Shen. Two-timescale learning-based task offloading for remote IoT in integrated satellite-terrestrial networks. 10(12):10131–10145, June 2023. doi: 10.1109/JIOT.2023.3237209.

[203] Fengxiao Tang, Bomin Mao, Nei Kato, and Guan Gui. Comprehensive survey on machine learning in vehicular network: Technology, applications and

challenges. 23(3):2027–2057, Third Quarter 2021. doi: 10.1109/COMST.2021.3089688.

[204] Fuhui Zhou, Rose Qingyang Hu, Zan Li, and Yuhao Wang. Mobile edge computing in unmanned aerial vehicle networks. 27(1):140–146, February 2020. doi: 10.1109/MWC.001.1800594.