

iVFC: A proactive methodology for task offloading in VFC

by Aisha Muhammad Ahmad Hamdi

Thesis submitted in fulfilment of the requirements for
the degree of

Doctor of Philosophy

under the supervision of Farookh Hussain

University of Technology Sydney
Faculty of Engineering and Information Technology (FEIT)

July 2023

Certificate of Original Authorship

I, Aisha Hamdi declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology in the school of Computer Science at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature:

Date: 31/07/2023

ABSTRACT

The Internet of Things plays an important role in the development of the Internet of Vehicles (IoV), where vehicles have become more connected and intelligent through equipment of various functional and more advanced technologies. This has led to the emergence of many vehicular applications such as augmented reality (AR) and self-driving applications. However, these applications require a large number of computational resources and generate a huge number of tasks and complex data that requires a large amount of computation and storage capacity. As a result, the demand for more computational and communication resources from these vehicles has increased, however this increasing demand for computational resources from such applications has not been met. To meet the increasing demand for enhanced computation and communication capacities in IoV, the common solution is to process these tasks using the high-capacity servers remotely located in the cloud to reduce energy consumption and provide high storage capacity for vehicles. However, the cloud is not an ideal solution due to the long transmission distance and latency between the source vehicles and the cloud servers, which leads to an increase in latency and network congestion. Therefore, vehicular fog computing (VFC) has been proposed as a promising solution to address the limitations of traditional cloud computing.

In VFC, the idle resources of moving and parked vehicles can be used for computation purposes to minimize the processing delay of compute-intensive vehicular applications by offloading tasks from the edge servers or vehicles to nearby fog node vehicles for execution. However, the offloading decision is a complicated process and the selection of an appropriate target node is a crucial decision that the source node has to make.

After studying the recent literature on task offloading in VFC, we found that many solutions have been proposed in the literature to handle the task offloading process, however, most solutions are reactive-based. This means that each fog node

vehicle will offload its computation tasks when it consumes all of its resources and becomes overloaded, which slows down the task execution process and affects the performance of the VFC network.

This thesis presents a novel and intelligent methodology for task offloading in VFC. The novelty of the proposed methodology lies in its proactive nature. By leveraging prior utilization-based prediction techniques, our methodology can proactively determine the need to offload a task to a target fog node vehicle based on the prior utilization-based prediction technique. Particularly, under the proposed methodology, the vehicle's need for computational resources in the next time slot is intelligently predicted and this prediction is used as the criterion for target node selection and task offloading. Furthermore, the proposed methodology includes an incentive mechanism to motivate fog node vehicles (FNvs) to accept an incoming task.

By emphasizing predictive decision-making and incentivized collaboration, our approach ensures that task offloading process is efficient, real-time, and responsive to satisfy the requirements of IoV applications.

Acknowledgements

At the end of my scientific journey, during which I faced numerous challenges, I begin by expressing with my heartfelt thanks to Allah, who guided me along the path, illuminating my way with hope to achieve my goals and granting me the patience to relish the sweetness of success upon arrival.

I am grateful to myself for overcoming obstacles and maintaining focus on the research objectives, even during moments of doubt, for the countless hours of research and analysis to explore new ideas and pushing the boundaries of my knowledge in the field, for my insistence on continuing and not giving up despite the difficulties I faced.

I extend my deepest appreciation to Professor Farookh Hussain who was not only a supervisor, but also as a steadfast supporter and an inspiring mentor. His unwavering guidance, patience, and willingness to lend a helping hand during challenging moments were instrumental in my journey. I am indebted to him for the abundance of success I now celebrate.

Furthermore, my heartfelt appreciation goes to my parents, my husband, and my two beloved daughters, Tala and Diala, who have been my constant inspiration and motivation throughout this thesis journey. To my friends, whose unwavering support and encouragement have been the cornerstone of this endeavour. Without their collective backing, this accomplishment would not have been possible.

I would like to express my sincere gratitude to Bitbrains IT Services Inc. for graciously providing the GWA-T-12 Bitbrains dataset. This dataset has been invaluable to my research and I appreciate the opportunity to utilize it in my work. In accordance with the copyright note provided by Bitbrains IT Services Inc., I acknowledge the source of the data and comply with their request. I also extend my thanks to

the authors of the "CCGrid 2015 paper" and the Grid Workloads Archive for their contributions to this dataset.

Lastly, I am immensely grateful to Jazan University for granting me the opportunity to finish my Ph.D., Saudi Arabian Cultural Mission (SACM) in Australia for supporting me during my Ph.D. journey, and the University of Technology Sydney (UTS) for granting me the opportunity to pursue my studies within its nurturing environment, which became a second home to me.

List of Publications

The following is a list of my research papers during my PhD study.

Journal Papers

1. **Hamdi, Aisha Muhammad A., Farookh Khadeer Hussain, and Omar K. Hussain** "Task offloading in vehicular fog computing: State-of-the-art and open issues." *Future Generation Computer Systems, Elsevier*, (**JCR Q1 Journal**).
2. **Hamdi, Aisha Muhammad A., Farookh Khadeer Hussain, and Omar K. Hussain** "iVFC: An intelligent framework for task offloading in vehicular fog computing" (submitted)
3. **Hamdi, Aisha Muhammad A. and Farookh Khadeer Hussain**, "An incentive mechanism for task offloading in VFC: A Stackelberg game-based approach", (under preparation)

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

وَقُلْ رَبِّ زِدْنِي عِلْمًا

Table of contents

Certificate	i
Abstract	ii
Acknowledgments	iv
List of Publications	vi
List of Figures	xiv
List of Tables	xvii
Abbreviation	xix
1 Introduction	1
1.1 Introduction	1
1.2 The potential of IoT in the internet of vehicles (IoV)	2
1.3 IoV and the emergence of vehicular fog computing (VFC)	4
1.4 The concept of vehicular fog computing (VFC)	5
1.4.1 Vehicular fog computing architecture	7
1.4.2 Resource management in VFC	7
1.5 Task offloading in VFC and its related issues	10
1.6 Aim of this Thesis	13
1.7 Scope of this thesis	14
1.8 Significance of the thesis	14
1.8.1 Scientific contributions	15
1.8.2 Social contributions	15

1.9	Plan of the thesis	16
1.10	Conclusion	18
2	A Systematic Literature Review	20
2.1	Introduction	20
2.2	SLR contributions	20
2.3	Key requirements needed to form an SLA during the task offloading process	22
2.4	Systematic review protocol	24
2.4.1	Searching the literature	24
2.4.2	Defining the inclusion and exclusion criteria	26
2.4.3	Quality assessment of the shortlisted articles	27
2.4.4	Shortlisted papers for the SLR and their classification into the broad areas	31
2.5	Analysis of the shortlisted papers against the requirements of task offloading in VFC	31
2.5.1	Analysis of papers in the category of task offloading in VFC	31
2.5.2	Analysis of papers in the category of fog node selection	34
2.5.3	Analysis of papers in the category of QoS assessment	49
2.6	Open research issues	60
2.7	Limitations of this SLR	62
2.8	Conclusion	62
3	Problem Definition	63
3.1	Introduction	63
3.2	Key definitions	63

3.3	Problem definition	66
3.4	Research Questions	67
3.4.1	Research Question 1 (RQ1)	68
3.4.2	Research Question 2 (RQ2)	68
3.4.3	Research Question 3 (RQ3)	68
3.4.4	Research Question 4 (RQ4)	68
3.5	Research objectives	68
3.5.1	Research Objective 1	68
3.5.2	Research Objective 2	69
3.5.3	Research Objective 3	69
3.5.4	Research Objective 4	70
3.6	Conclusion	70
4	Research Methodology and Solution Overview	71
4.1	Introduction	71
4.2	Key definitions	71
4.3	Selected Research Methodology	73
4.4	Overview of the proposed iVFC solution	76
4.4.1	Architecture of the iVFC	76
4.4.2	Overview of the proposed iVFC framework	78
4.5	Overview of the solution for research objective 1 (RO1)	82
4.6	Overview of the solution for research objective 2 (RO2)	85
4.7	Overview of the solution for the research objective 3 (RO3)	87
4.8	Evaluation and validation of the proposed iVFC solution (RO 4)	87

4.8.1	The validation steps for the solution to research objective 1 (RO1)	88
4.8.2	The validation steps for the solution to research objective 2 (RO2)	90
4.8.3	The validation steps for the solution to research objective 3 (RO3)	91
4.9	Conclusion	93
5	A proactive-based task offloading in VFC using machine learning prediction techniques	94
5.1	Introduction	94
5.2	The proposed framework of the iVFC-predictive analytic module . . .	95
5.2.1	Workload observation	96
5.2.2	Workload prediction	96
5.2.3	The overloading decision	97
5.3	Evaluation of the proposed iVFC-predictive analytic module	97
5.3.1	Dataset	97
5.3.2	The experimental setup and implementation	99
5.3.3	Evaluation metrics	114
5.4	Results and discussion	116
5.5	Conclusion	121
6	An incentive-based framework for task offloading in VFC	122
6.1	Introduction	122
6.2	The proposed framework of the iVFC-incentive module	122

6.3	Mathematical formulation of the solution for the proposed iVFC-incentive module using Stackelberg game theory	126
6.4	Evaluation of the proposed iVFC-incentive module	129
6.4.1	Dataset	129
6.4.2	The selected implementation platform	129
6.4.3	The experimental setup and implementation	130
6.4.4	Evaluation metrics	136
6.5	Results and discussion	136
6.6	Conclusion	146
7	An intelligent framework for target node vehicle selection in the iVFC system	148
7.1	Introduction	148
7.2	The proposed framework of the iVFC-TNv selection module	149
7.3	Evaluation of the proposed iVFC-TNv selection module	150
7.3.1	Dataset	150
7.3.2	The selected implementation platform	151
7.3.3	The experimental set up and implementation	152
7.3.4	Evaluation metrics	174
7.4	Results and discussion	175
7.4.1	Evaluation of the TOPSIS method	175
7.4.2	Evaluation of the XGBoost method	177
7.4.3	Evaluation of the DNNs method	180
7.4.4	Comparison of the three methods used to develop the iVFC-TNv selection module	181

7.5 Conclusion	184
8 Conclusion and future work	187
8.1 Introduction	187
8.2 Problems addressed in this thesis	188
8.3 Contributions to the existing literature	188
8.3.1 Systematic literature review (SLR)	189
8.3.2 Development of a novel framework called iVFC for task offloading in VFC	189
8.3.3 Evaluation and validation of the proposed framework	191
8.4 Conclusion and future work	191

List of Figures

1.1	IoT in relation to IoV and VFC	6
1.2	Visual description of VFC environment	6
1.3	Thesis structure	19
4.1	The proposed Design Science Research Methodology	76
4.2	The architecture of the proposed iVFC system	78
4.3	Overview of the iVFC framework	81
4.4	Working steps of the proposed iVFC-predictive analytic module	83
4.5	Framework of the proposed iVFC-predictive analytic module	84
4.6	Working steps of the proposed iVFC-incentive module	86
4.7	Working steps of the proposed iVFC-TNv selection module	88
4.8	The proposed framework of the iVFC-TNv selection module	89
5.1	The proposed framework of the iVFC-predictive analytic module	98
5.2	Snapshot of the dataset used to train the prediction models	100
5.3	CPU utilization prediction experiment on Azure portal	103
5.4	Memory usage prediction experiment on Azure portal	104
5.5	Fuzzy logic model for predicting the workload of FNvs	108
5.6	The fuzzy sets for the CPU utilization input variables	110
5.7	The fuzzy sets for the memory usage input variables	111

5.8	The fuzzy sets for the overloading decision input variables	112
5.9	The fuzzy logic model rules to obtain the offloading decision outputs .	113
5.10	Predicted vs. true for the exponential smoothing algorithm for CPU utilization prediction	118
5.11	Residuals histogram for the exponential smoothing algorithm for CPU utilization prediction	119
5.12	Predicted vs. true for the exponential smoothing algorithm for memory usage prediction	119
5.13	Residuals histogram for the exponential smoothing model for memory usage prediction	120
5.14	Snapshot of the calculations of the accuracy of the fuzzy logic model using Excel	120
6.1	Working steps of the proposed iVFC-incentive module	125
6.2	Effect of using the proposed Stackelberg game-based incentive mechanism on the participation level of FNvs (attempt 1, FNvs =10, TNvs =5)	138
6.3	Snapshot of the implementation of proposed the Stackelberg game-based incentive mechanism in Google Collab., attempt 1	139
6.4	Effect of using the proposed Stackelberg game-based incentive mechanism on the participation level of FNvs (attempt 2, FNvs =15, TNvs =10)	140
6.5	Snapshot of the implementation of proposed the Stackelberg game-based incentive mechanism in Google Collab., attempt 2	141
6.6	Effect of using the proposed Stackelberg game-based incentive mechanism on the participation level of FNvs (attempt 3, FNvs =20, TNvs =13)	142

6.7	Snapshot of the implementation of proposed the Stackelberg game-based incentive mechanism in Google Collab., attempt 3	143
6.8	Non-incentive scenario (attempt 1, FNvs =10, TNvs =8)	143
6.9	Snapshot of the implementation of the non-incentive scenario in Google Collab., attempt 1	144
6.10	Non-incentive scenario (attempt 2, FNvs =15, TNvs =13)	144
6.11	Snapshot of the implementation of the non-incentive mechanism in Google Collab., attempt 2	145
6.12	Non-incentive scenario (attempt 3, FNvs =20, TNvs =17)	145
6.13	Snapshot of the implementation of the non-incentive mechanism in Google Collab., attempt 3	146
7.1	Framework of the iVFC-TNv selection module	150
7.2	Hierarchical structure for AHP method	155
7.3	Snapshot of the accuracy calculations of TOPSIS method using Excel	177
7.4	XGBoost method ranking compared to the actual ranking of the top-30 nodes	179
7.5	XGBoost method ranking compared to the actual ranking of the whole dataset	179
7.6	Prediction results for the training dataset using DNNs with four layers	182
7.7	Prediction results for the testing dataset using DNNs with four layers	182
7.8	Prediction results for the top-30 testing dataset using DNNs with four layers	183
7.9	Comparing the different ranking methods based on MAE	185
7.10	Comparing the different ranking methods based on run time in seconds	185

List of Tables

2.1	Search terms	25
2.2	Paper selection stages.	27
2.3	Quality assessment criteria used.	28
2.4	Assessment of the studies against the four criteria.	28
2.5	Papers that meet the quality evaluation criteria from Step 3 of the selection process.	36
2.6	Comparison of the selected papers against requirements R1-R3 for efficient task offloading.	51
5.1	ML time series prediction experiments to build and train the prediction models	102
5.2	The parameters used in the time series prediction experiments	102
5.3	Prediction values for an interval of 15 minutes for CPU utilization and memory usage of the 100 FNvs	105
5.4	Comparison of the top-10 ML algorithms applied on experiment 2 . .	117
5.5	Calculations of the evaluation metrics of the fuzzy logic model	118
6.1	Comparison of Experiments: Stackelberg Game theory-based incentive vs. Non-incentive Scenario	137
7.1	AHP decision matrix	154

7.2	Weighted percentage of the fundamental scale used for AHP method .	156
7.3	The pair-wise matrix for AHP method	157
7.4	Sum of the pair-wise matrix for AHP method	158
7.5	The normalized pair-wise matrix for AHP method	158
7.6	Criteria weights using AHP method	158
7.7	Consistency calculations for AHP method, step 1	159
7.8	Consistency calculations for AHP method, step 2	159
7.9	Consistency calculations for AHP method, step 3	160
7.10	The random index of a randomly generated pair-wise matrix	160
7.11	Experiments of implementing artificial neural network to rank the list of TNvs	170
7.12	Experiments of implementing deep neural networks with three hidden layers to rank the list of TNvs	172
7.13	Experiments of implementing deep neural networks with four hidden layers to rank the list of TNvs	173
7.14	Experiments of implementing deep neural networks with five hidden layers to rank the list of TNvs	174
7.15	The top-30 ranked TNvs according to the TOPSIS method compared to the actual ranking	176
7.16	Experiments using XGBoost for ranking the list of the TNvs	178
7.17	Summary of the experiments using DNNs with varying numbers of hidden layers	180
7.18	MAE and run time for the three methods for TNv selection	184

Abbreviation

AI	Artificial Intelligence
AHP	Analytic Hierarchy Process
ANN	Artificial Neural Network
AR	Augmented Reality
C.I	Consistency Index
CPU	Central Processing Unit
DL	Deep Learning
DNNs	Deep Neural Networks
DSRM	Design Science Research Methodology
FN _v	Fog Node Vehicle
FP	False Positive
FSC	Fog Service Consortium
FSP	Fog Service Provider
FSN	Fog Server Node
FR	Fog Repository
GPS	Global Positioning Systems
IA	Ideal Alternative
IoT	Internet of Things
IoV	Internet of Vehicles
MCDM	Multi-Criteria Decision-Making
MAE	Mean Absolute Error
ML	Machine Learning

Abbreviations

NE	Nash Equilibrium Game Theory
NRMSE	Normalized Root Mean Square Error
QoS	Quality of Service
RAM	Random Access Memory
RI	Random Index
RSU	Roadside Units
SLA	Service Level Agreement
SLR	Systematic Literature Review
SN _v	Source Node Vehicle
TN	True Negative
TN _v	Target Node Vehicle
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
TP	True Positive
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VEC	Vehicular Edge Computing
VFC	Vehicular Fog Computing
VMs	Virtual Machines
XGBoost	Extreme Gradient Boosting

Chapter 1

Introduction

1.1 Introduction

With the development and increased adoption of fog technologies in the future, vehicles might seamlessly communicate with each other, making intelligent decisions and enhancing various aspects of daily life. This interconnected reality is made possible through the Internet of Vehicles (IoV), an extension of the broader concept of the Internet of Things (IoT) specifically designed for the automotive industry. As vehicles become more connected and intelligent, they generate an enormous amount of data and computational tasks that demand high computation and storage capacities [1]. This growing demand has led to the emergence of vehicular fog computing (VFC) as a promising solution to address the limitations of traditional cloud computing where vehicles can serve as fog nodes to process tasks for other resource-limited vehicles or fog servers. But how does this innovative computing paradigm change the way tasks are processed and offloaded in vehicular environments?

By exploring the complexities of resource management in the VFC environment and how to benefit from the computation resources available in today's advanced vehicles, this thesis explores the optimal utilization of fog node vehicles (FNvs) and the challenges associated with making intelligent offloading decisions among vehicles.

This chapter is organized as follows. Section 1.2 introduces IoV as one significant application of IoT. Section 1.3 introduces the emergence of VFC. Section 1.4 introduces the concept of VFC and outlines the related issues. Section 1.5 addresses task offloading issues in VFC. Section 1.6 identifies the research objective. Section 1.7

defines the scope of this study by specifying its boundaries. Section 1.8 highlights the significance of this thesis and its contributions. Section 1.9 provides an overview of the structure of this thesis and section 1.10 concludes this chapter.

1.2 The potential of IoT in the internet of vehicles (IoV)

In today's connected world, information systems play an essential role in collecting, processing, and disseminating data for diverse uses. An information system is a structured and interconnected group of components that work together within an organisation to gather, process, store, and distribute data and information [2]. It includes both the technological infrastructure and the people, processes, and procedures involved in data management and utilization for decision-making and accomplishing organisational goals. With the presence of the IoT, the environment of information systems has expanded, allowing for integrating physical objects with digital networks [2].

The IoT concept refers to a network of physically connected objects equipped with sensors, software, and connection capabilities [1]. With the IoT, a vast number of devices are able to communicate with each other via the Internet to gather and exchange data with each other, communicate, make intelligent decisions, and perform actions to enhance various aspects of daily life [1].

IoT plays an important role in the development of information and communication technologies [3]. One aspect where IoT helps in expanding communication mechanisms is the IoV. The IoV extends the concept of IoT by including to the domain of vehicles. It involves equipping vehicles with sensors, connectivity, and computing capabilities to enable communication and data exchange between vehicles, roadside infrastructure, and other entities. The IoV depends on a number of technologies such as wireless communication, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, sensors and global positioning sys-

tems (GPS). These technologies allow for the smooth integration of vehicles into the larger IoT ecosystem, which creates a connected and intelligent vehicular network [3].

We list some of the common applications of IoT in the Internet of Vehicles as follows [3]:

- a) Connected Car Services: IoT makes it possible for vehicles to be connected, opening the door to a range of features like infotainment, navigation, and remote diagnostics. Vehicles can be connected to the internet, giving users access to real-time traffic updates, weather reports, and entertainment alternatives.
- b) Intelligent Traffic Management: To collect real-time traffic information, IoT sensors and communication networks can be installed at intersections and on roads to help in managing congestion, streamlining traffic, and improving overall transportation efficiency.
- c) Vehicle-to-Vehicle (V2V) Communication: IoT makes it possible for vehicles to interact with one another and share details like speed, location, and intent.
- d) Vehicle-to-Infrastructure (V2I) Communication: IoT enables vehicles to communicate with various infrastructure components, including traffic lights, parking metres, and toll booths. This communication enables efficient toll collection, intelligent parking guidance, and optimised traffic light timing.
- e) Autonomous Vehicles: IoT technology is essential in enabling autonomous driving. IoT sensors, including LiDAR, radar, and cameras, gather information from the environment around the vehicle, enabling perception and decision-making algorithms to move through and interact with it.

1.3 IoV and the emergence of vehicular fog computing (VFC)

With the development brought by IoT into the IoV, vehicles have become more connected and intelligent through different functional equipment and more advanced technologies. This has led to the emergence of many vehicular applications, ranging from entertainment applications such as gaming to augmented reality (AR) and self-driving applications [4]. However, these applications require a large number of computational resources (such as computational power or memory) and generate a huge number of tasks and complex data that require high computation and storage capacity [5]. Moreover, some applications are delay-sensitive and must be processed within specified time constraints, particularly those applications dealing with emergency and natural disaster applications [6]. As a result, the demand for more computational and communication resources from these vehicles has increased, while vehicles have been unable to satisfy this increasing demand for computational resources from such applications [7]. To meet the increasing demand for high computation and communication capacities in IoV, a common solution was to process these tasks using the high-capacity servers remotely located in the cloud to reduce energy consumption and provide high storage capacity for vehicles [7].

However, even though the cloud is rich in computation and communication resources, it is not an ideal solution due to the long transmission distance and latency between the source vehicles and the cloud servers. Moreover, offloading to the cloud leads to an increase in latency and network congestions, particularly when the cloud servers are in different geographical regions [8]. Therefore, to reduce the transmission distance and alleviate the heavy burden on the cloud, the notion of vehicular edge computing (VEC) has been proposed as a possible solution to enhance task offloading-related issues [9].

The notion of VEC has brought computational resources available in the cloud

closer to end users (vehicles) and has enabled vehicles to process their computation tasks using the available resources of edge servers [8]. VEC plays an important role in providing edge services (in the form of computational resources) with lower delay and higher bandwidth to the vehicles [9]. In VEC, vehicles act as communication, computation, and storage resource providers while the roadside units (RSU) are the edge servers that are deployed close to vehicles to gather, process and store data in a timely manner. In VEC, when there is not enough capacity in the form of computational resources in the source vehicles, it can offload their computation tasks to RSUs for processing and obtain the execution results [9]. However, VEC suffers from the high load on the edge servers during peak hours and the high cost of the deployment of extra servers (RSU) to cover a wider range of geographical areas [8]. To overcome the limitations of edge computing, VFC has been proposed as a promising solution [5], [10].

VFC expands the concept of fog computing by focusing primarily on the automotive industry. By integrating fog computing to the automotive sector, VFC expands the capabilities of IoT and IoV. It entails embedding fog computing resources within vehicles and their surrounding infrastructure, enabling efficient data processing and communication [10].

Figure 1.1 shows a visual description of the relation between IoT, IoV and VFC.

1.4 The concept of vehicular fog computing (VFC)

VFC can be defined as communication between a group of smart vehicles located near each other and connected through a peer-to-peer model of communication [11]. In VFC, the idle resources of moving and parked vehicles can be used for computation purposes to minimize the processing delay of the compute-intensive vehicular applications [12] [5]. The basic premise of VFC is to offload the tasks from

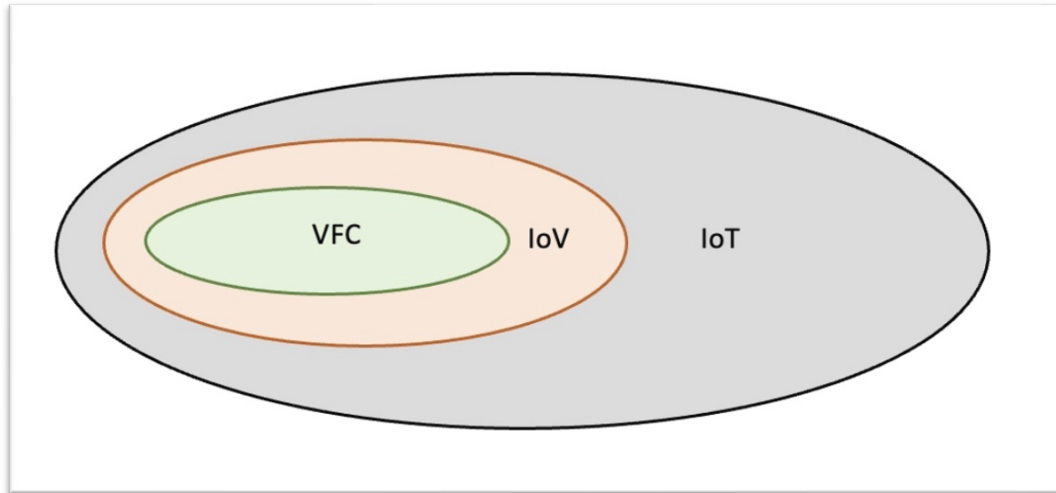


Figure 1.1 : IoT in relation to IoV and VFC

the edge servers or vehicles to nearby fog node vehicles for execution [8]. Figure 1.2 shows a visual description of the VFC environment.

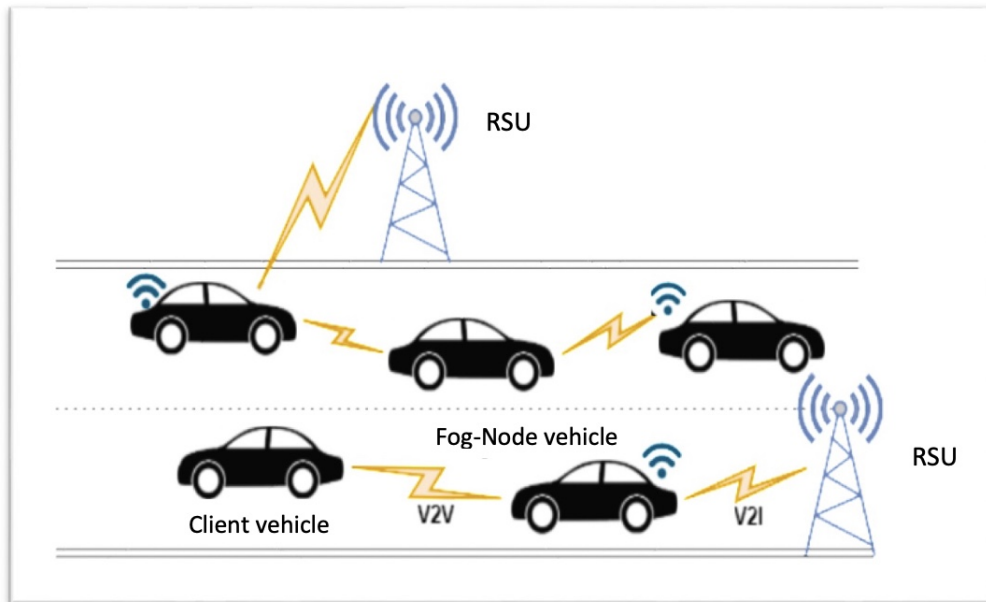


Figure 1.2 : Visual description of VFC environment [13]

1.4.1 Vehicular fog computing architecture

Generally speaking, VFC consists of three layers: the cloud computing layer, the fog computing layer and the vehicular computing layer [14].

The fog computing layer is located at the edge of the network and provides local services for end-users. These services are offered with shorter latency and broader bandwidth. The cloud computing layer enhances the ability of fog computing through pre-scheduling the required resources and providing further data processing and permanent data storage. The vehicular computing layer communicates directly with fog computing layer in a wireless manner to provide more computing and communication capabilities.

1.4.2 Resource management in VFC

VFC is a platform that provides local data processing and storage capabilities in a distributed manner instead of sending them to the remote cloud. VFC supports additional geo-distributed applications that cater to user engagement demands, entertainment and social needs, and improved communication services for people [14].

Due to the geographical distribution of VFC, it is necessary to organize and manage all vehicular fog nodes efficiently in the practice of VFC deployment. Therefore, resources in VFC environment have to be managed effectively to ensure that resources are utilized optimally to achieve the quality of service (QoS).

Resource management is the process of planning, allocating, and optimizing different types of resources within a system in order to successfully accomplish the desired objectives [14]. In VFC, resource management includes the efficient allocation and utilization of computing, storage, and network resources in a VFC environment. Therefore, resource management issue arises as a challenge in the process of organizing and managing vehicular fog nodes [14].

Recently, several researchers have focused on proposing innovative solutions for resource management in VFC. For example, Zhu et al. [15] addressed two problems that resource management approaches need to tackle. First, one vehicle may be within the contact range of several vehicular fog nodes. In this scenario, it is a matter of how to select the most appropriate fog node. Second, the intensive distribution of vehicular fog nodes may result in difficulties in providing direct communication between vehicles due to the long distance. Therefore, installing resource management systems on cellular fog nodes is more feasible. Ghobaei-Arani et al. [16] classified resource management issues into six categories: application placement, resource scheduling, task offloading, load balancing, resource allocation, and resource provisioning.

Managing resources in the VFC environment involves different aspects where each aspect has its related problems and issues.

In the following, we outline and discuss some of the key issues related to resource management in VFC [4], [14], [16], [17]:

- a) Task Offloading: Resource management is primarily concerned with determining which computational tasks should be offloaded from one vehicle to another vehicle or fog server. The decision of task offloading is typically based on factors such as task characteristics (e.g., computation intensity, data size), vehicle conditions (e.g., battery level, available resources), and network conditions (e.g., latency, bandwidth). Offloading decisions can be made by the vehicles themselves or by a central controller.
- b) Reactive vs. proactive offloading: Making an offloading decision when the source node vehicle's capacity is full or has hit capacity thresholds is known as reactive offloading. However, an offloading decision can be more efficient if the need for offloading can be determined a priori by the source node. The pro-

cess of determining the need to offload tasks in advance is known as proactive offloading

- c) Resource Allocation: When a vehicle decides to offload a task, resource management involves allocating the different resources needed to process the task such as computing, storage, and network at the chosen target node vehicle (TNv) to handle this task. This includes determining the optimal TNv to execute the tasks based on factors such as proximity, resource availability, and load balancing. The allocation process should consider the rapid changes in the availability of fog node vehicles that occur due to the dynamic nature in the VFC environment.
- d) Quality of Service (QoS) Management: It is important to consider QoS requirements for the offloaded tasks in resource management in VFC. Depending on the task, QoS requirements may differ in terms of response time, reliability, and throughput. To meet the desired QoS targets, resource management techniques must prioritize the tasks accordingly.
- e) Mobility Management: In VFC, vehicles are mobile, adding an additional challenge to resource management. The connectivity of vehicles to fog nodes may change as they move, making it necessary to have resource management techniques that can handle this seamlessly as vehicles move.
- f) Energy efficiency: In resource management, energy efficiency is a crucial consideration due to the limited energy resources of vehicles. Different techniques such as dynamic power management, task scheduling, and resource allocation can be used to optimize energy consumption to meet the computational demands of the offloaded tasks.
- g) Security and Privacy: Resource management in VFC should also include security and privacy aspects. When offloading tasks between vehicles in the

VFC environment, it is important to have appropriate security mechanisms in place to protect against unauthorized access, data breaches, and malicious attacks. Privacy-preserving techniques can be employed to protect sensitive information during task offloading and resource allocation.

This research focuses on the task offloading aspect of resource management in VFC and explores the different issues associated with offloading tasks between vehicles and fog nodes within the VFC environment.

1.5 Task offloading in VFC and its related issues

Task offloading is a technique that solves the problem of resource constraints in distributed computing by enabling a resource-limited edge server/vehicle to execute its computation tasks. This is achieved by offloading them to nearby resource-rich fog-node vehicles to improve the system performance and reduce energy consumption and execution delay. [16]. In VFC, the source nodes (vehicles) are mobile nodes with intensive computation tasks that need to be executed within time constraints. The execution of these tasks locally is difficult due to the resource limitation of the source vehicle and the time constraints of the job. Therefore, offloading tasks to another fog-node vehicle can solve such an issue [18]. The offloading decision is a complicated process that has three main stakeholders: the mobile vehicle (i.e., source node vehicle (SNv)) with the intensive computation tasks, the communication link used to transfer the signal, and the fog node (i.e., target node vehicle (TNv)) that executes the offloaded tasks and returns the execution results. The SNv is responsible for the following activities:

- a) identifying the need to offload a given task or tasks;
- b) identifying an appropriate TNv;
- c) communicating and negotiating with the TNv in relation to the task being

- offloaded and the reward offered for resource sharing ;
- d) offloading the task to the TNv; and
- e) receiving the completed task from the TNv.

The offloading decision and the selection of an appropriate TNv considering constraints (such as but not limited to latency and QoS) is a crucial decision that the source node has to make.

Latency refers to the delay which occurs in transmitting data from the SNv to the TNv and receiving a response. This can be due to the distance between the requester SNv and the fog server that receives the offloading requests. Latency plays an important role in determining the overall performance of the system. To ensure optimal latency, the SNv needs to identify a TNv that can minimize the transmission delay.

It is also important for the SNv to evaluate the QoS requirements for the specific task or application being offloaded. QoS includes several metrics, including reliability, availability, throughput, and response time, and the SNv needs to ensure that the selected TNv can meet these requirements. For example, for applications that require real-time data processing or time-sensitive tasks such as video streaming or autonomous vehicle control, low latency and high QoS are required. The SNv must identify a TNv that not only offers low latency but also has sufficient computational resources to handle the workload while guaranteeing QoS.

Concomitant with the issue of offloading in VFC, three pertinent and pressing questions arise that require further investigation. The first question is when does the node have to offload the task. The second question is how much workload can the node offload, taking into account the capacity of the selected TNv to process the task. The third question is where to offload the task i.e., determining to which

TNv to offload the task [19]. The right decision of when, how much and where to offload the task will determine the effectiveness of the task offloading decision.

Furthermore, while a timely decision regarding these three questions is critical, the complexity of the decision-making process is compounded due to the temporal variations of VFC system mobility, uncertainty in the resulting offloading latency, and unknown traffic statistics [19].

The task offloading process is confronted by several critical challenges, which could negatively affect the efficiency of task completion. We list the most pertinent ones and discuss them as follows:

The first issue is that all the recent proposed task offloading methods are reactive, which means that offloading the task is carried out when the SNv's capacity is full or has hit the capacity thresholds. However, the offloading decision can be more efficient if the need for offloading can be determined a priori by the SNv. We term the process of determining the need to offload tasks in advance as proactive offloading. The prior utilization profile of the source node can be used for this purpose to intelligently predict a prior need to offload rather than offloading after the need has become apparent.

The second issue is the lack of information of the available FNvs to host the offloaded tasks and their capacity (i.e., uncertainty) is a crucial determinant of the task offloading process. The lack of relevant information about available host nodes and their suitability to host tasks leads to uncertainty in which the SNv is uncertain about where to offload its task and what is the capacity of the available TNvs, and this needs to be modelled. The prior knowledge of these information can help the overloaded SNv to select the optimal TNv to which to offload the task [5].

The third issue pertains to minimising the total time in the selection of the best TNvs for offloading the task. Therefore, task offloading can be implemented

efficiently if and only if an effective TNv selection technique is used. The employment of predictive offloading, which can be used to further reduce latency and improve QoS, will contribute to efficient decision making.

The fourth issue is how to motivate FNvs to share their idle resources and participate in the task offloading process to guarantee an availability of sufficient resources at the time of task offloading to process the task.

To address the aforementioned four research challenges, in this thesis, we propose a proactive task offloading methodology that will consider a TNv selection mechanism to minimize the total delay and reduce uncertainty and an incentive mechanism to ensure task offloading acceptance and efficient task completion.

1.6 Aim of this Thesis

In this thesis, we present an intelligent methodology for task offloading in VFC. The methodology can proactively determine the need to offload a task to a target fog node vehicle (TNv) based on the prior utilization-based prediction technique. Particularly, under the proposed methodology, we intelligently estimate the vehicle's need for computational resources in the next time slot and use this prediction as the criterion for TNv selection and tasks offloading. Furthermore, our methodology considers the selection of TNv based on its available workload information to minimize uncertainty and to enhance the delay performance of the VFC system. Finally, an incentive mechanism is used in our methodology to motivate FNvs to take an incoming task.

The task offloading solution proposed by our methodology contributes to enhance the process of task offloading in VFC to satisfy the requirements of IoV applications. These requirements include delay minimization and efficient task completion. The benefits of our methodology include the optimal selection of TNv for offloading

the task and the efficient motivation for FNvs for resource sharing to guarantee offloading acceptance and efficient task completion.

1.7 Scope of this thesis

This thesis develops an intelligent methodology for task offloading in VFC environments. The scope of this thesis is as follows:

1. Workload prediction: The methodology utilizes a prior utilization-based prediction technique to proactively determine the need for task offloading to a TNv. It involves the conceptualisation and development of a predictive algorithm to predict the vehicle's future workload based on its current workload (i.e., their CPU utilization and memory usage) to determine when it is going to be overloaded.
2. TNv Selection: The proposed methodology incorporates the predicted workload as the criterion for intelligently selecting the optimal TNv that can efficiently handle the offloaded task.
3. Incentive Mechanism: An incentive mechanism is included in the methodology to motivate FNvs to accept incoming tasks and participate in resource sharing.

The scope of this thesis also includes the development of a prototype system to validate and evaluate the proposed algorithms.

1.8 Significance of the thesis

This thesis develops an intelligent framework to handle the task offloading process in VFC in a proactive manner that utilizes the idle computation resources available in the advanced smart vehicles in a more efficient way. This is achieved by applying intelligent methods and schemes that will help resource-limited fog nodes to execute

their tasks within an appropriate time frame while guaranteeing the QoS provided by fog service provider nodes.

Therefore, in this thesis, we propose, develop and evaluate an iVFC: an intelligent framework for task offloading in VFC to address the gaps identified in the literature. The contributions of this thesis are divided into scientific contributions and social contributions, which are outlined in the following subsections.

1.8.1 Scientific contributions

The scientific significance of this thesis is as follows:

- a) This research is the first of its type to propose a predictive mechanism for task offloading. Previous research (refer to chapter 2) focuses on reactive mechanisms only.
- b) This research proposes a workload-based TNv selection approach for making a reliable judgement and selection of the best possible TNv to which to offload the task.
- c) This research proposes the use of an incentive mechanism to encourage FNvs to participate in the VFC system.
- d) This research proposes and develops an innovative proactive offloading-based approach for use in VFCs. Such an approach will minimise the time taken for task offloading and helps in selecting a reliable target node for executing tasks.

1.8.2 Social contributions

The Social significance of this thesis is as follows:

- a) The outcomes from this research will contribute to the adoption and roll out of the VFC and its use in smart vehicles.

- b) As Saudi Arabia is now in the phase of building advanced smart cities as part of Vision 2030, this research will contribute to the roll out of smart vehicles as a rich resource to meet the demand of computation and communication capacities within these smart cities.
- c) The proposed framework can help to develop intelligent mechanisms to generate carbon credits from smart vehicles. This can be done by focusing on reducing the emissions associated with vehicle usage by using energy-efficient vehicles that have lower emissions or by enabling vehicle-to-grid (V2G) technology that allows the vehicles to provide energy back to the grid during peak demand periods.

1.9 Plan of the thesis

This thesis provides an overall methodology to develop an intelligent framework for task offloading in a VFC system. To develop the proposed framework, intelligent mechanisms using machine learning methods are applied to develop three modules: a predictive analytic module to help the overloaded source node vehicle (SNv) make the offloading decision, a TNv selection module to assist the overloaded SNv select the most optimal TNv to handle the offloaded task, and an incentive mechanism to encourage FNvs to participate in the task offloading process by sharing their idle resources. To achieve these objectives, this thesis is divided into eight chapters as detailed in Figure 1.3, and is organized as follows:

Chapter 1: This chapter introduces this thesis by presenting the concept of VFC and its emergence. It also highlights the issue of managing resources within a VFC environment. Then, it focuses on the task offloading issue and presents the different problems associated with offloading a task from one node to another, which formulates the objective of this thesis. Then it defines the thesis objective followed by the thesis scope. In addition, it outlines the various contributions of this research.

Chapter 2: This chapter gives a detailed explanation of the systematic literature review that was conducted to cover the recent literature on task offloading in VFC. The purpose of conducting an SLR is to identify the research gaps that this thesis addresses and clarifies the research gaps which have not been previously addressed by any other research.

Chapter 3: Based on the research gaps identified in Chapter 2, Chapter 3 identifies the research problem and the main research question and then it outlines the formulated research sub-questions. Then, it identifies the research objectives.

Chapter 4: In Chapter 4, the research methodology is explained in detail. This methodology addresses the research gaps that have been identified in the literature review. Chapter 5 also includes an overview of the proposed solution and how each research question is addressed.

Chapter 5: This chapter includes the different steps involved in developing a proactive-based framework for task offloading in VFC, corresponding to research objective 1. The purpose of this framework is to facilitate the proactive-based handling of task offloading for the VFC system in which the workload of each FNV is predicted to monitor and control its overloading condition. This framework comprises three phases, which are discussed in detail in Chapter 4. Chapter 4 also includes the steps to validate and evaluate the proposed predictive framework to answer the first part of research objective 4, which is validation of research objective 1.

Chapter 6: Chapter 6 overviews the steps involved in building an incentive-based framework for task offloading in VFC using a game theory approach, corresponding to research objective 2. The purpose of this framework is to encourage FNVs to participate in the task offloading process by sharing their idle computation resources. The aim of this framework is to increase the level of participation of FNVs in the VFC environment. Chapter 6 also includes the steps to validate and evaluate

the proposed selection framework to answer the second part of research objective 4, which is the validation of the research objective 2.

Chapter 7: Chapter 7 identifies the different steps involved in building a fog service provider selection framework for task offloading in VFC using three different methods, corresponding to research objective 3. The purpose of this framework is to assist the overloaded SNv to find the most optimal TNv that can efficiently handle its task. Chapter 7 also includes the steps to validate and evaluate the proposed selection framework to answer the last part of the research objective 4, which is to validate research objective 3.

Chapter 8: Chapter 8 concludes this thesis by summarising what has been achieved in this thesis and the work that will be conducted in the future to extend this research.

1.10 Conclusion

This chapter introduced the emergence of the VFC paradigm and discussed VFC from different aspects related to resource management. Then, it defined the research objective and gave an outline of the research scope. In addition, it identified the different contributions of this research.

In the next chapter, a detailed explanation of the systematic literature review that was conducted to cover the recent literature on task offloading in VFC is provided. The purpose of this SLR is to identify the research gaps that this thesis addresses and to clarify that these research gaps have not been previously addressed by any other research.

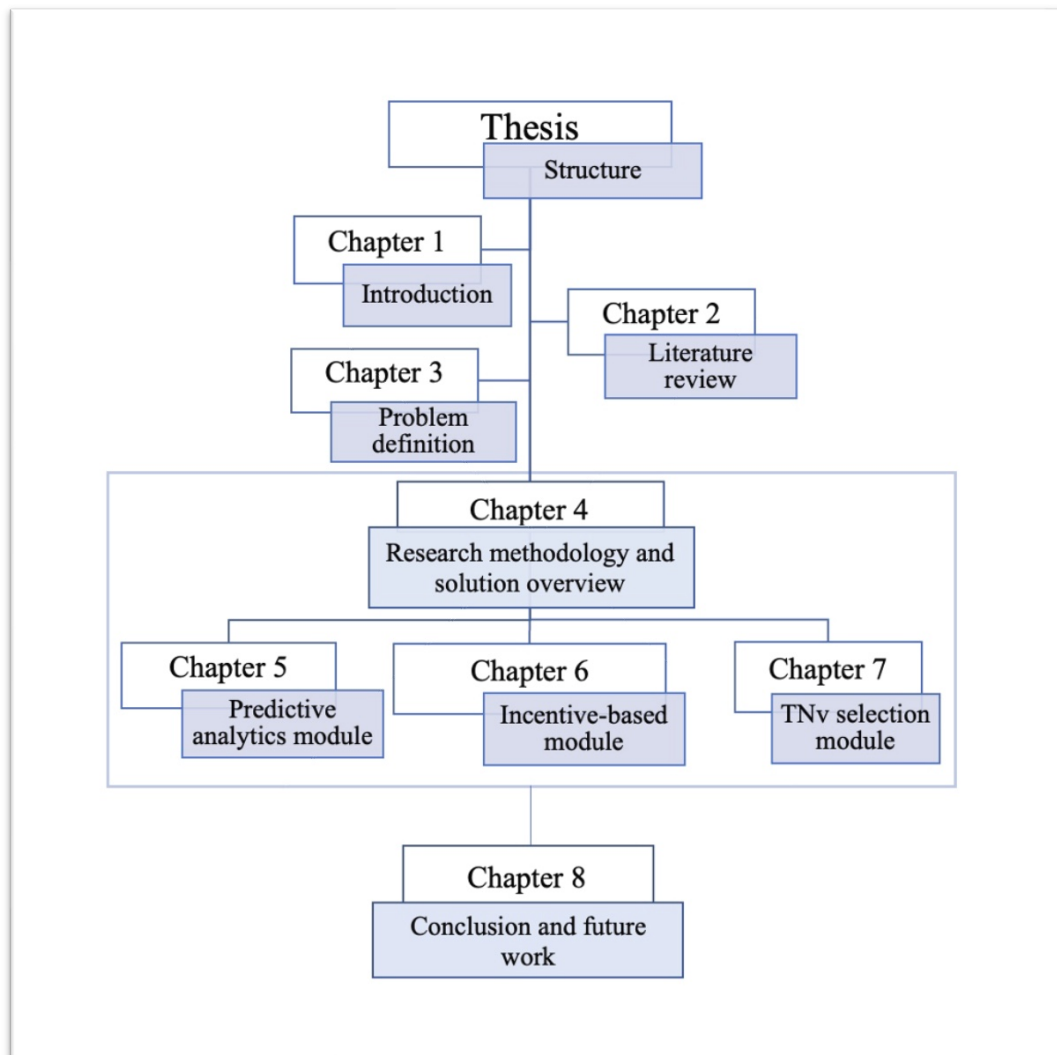


Figure 1.3 : Thesis structure

Chapter 2

A Systematic Literature Review

2.1 Introduction

The previous chapter overviewed the problem addressed in this thesis. Building on the previous chapter, this chapter provides a detailed explanation of the systematic literature review (SLR) that was conducted to cover the literature on task offloading in VFC.

This chapter is organized as follows. Section 2.2 outlines the contributions made by this SLR. Section 2.3 highlights the key requirements used to formulate this SLR. Section 2.4 discusses the protocol used to shortlist the papers chosen for this SLR, including the inclusion and exclusion criteria used for searching the literature. Section 2.5 includes an analysis of the shortlisted papers against the requirements of task offloading in VFC. Section 2.6 highlights the open gaps found in the selected literature. Section 2.7 outlines the limitations associated with this SLR and section 2.8 concludes this chapter.

The contents of this chapter have been published in the Future Generation Computer System journal, which is currently ranked in the top quartile of journals (JCR Q1). The contents of this SLR are available at the following link: <https://doi.org/10.1016/j.future.2022.03.019>.

2.2 SLR contributions

In VFC, the idle resources of moving and parked vehicles can be used for computation purposes to minimize the processing delay of compute-intensive or time-

critical vehicular applications [20][21]. The basic premise of VFC is to offload tasks from the edge servers or vehicles to nearby fog nodes (vehicles) for execution [22]. However, the offloading decision is a complicated process in which there is a requester or source node vehicle (SNv) with intensive computation tasks, and a receiver or target node vehicle (TNv) that executes the offloaded tasks and returns the execution results.

Furthermore, due to the dynamicity in VFC, the location of the SNv and TNv is not constantly fixed [23]. Hence, the SNv may need to offload to a TNv with no past offloading history. This chapter argues that the selection process should broadly follow the steps needed to form a service level agreement (SLA) to ensure that the right TNv is selected by managing the different constraints.

To address this gap, an SLR on task offloading is conducted to identify whether the examined literature considers the requirements identified in R1–R3. We then discuss the open gaps and areas of future research directions based on our analysis. The key contributions of this SLR to the literature are as follows:

- It argues that the process of selecting target fog nodes in VFC should broadly follow the steps needed to form an SLA.
- Specific to task offloading in VFC, it defines the three requirements (R1–R3) that need to be considered to ensure that the right target fog node is selected by managing the different constraints
- It identifies the existing approaches from the literature in the context of task offloading in VFC and determines their shortcomings from the perspective of forming an informed SLA.
- It identifies the gaps in the existing approaches and introduces these as open issues and future research directions.

2.3 Key requirements needed to form an SLA during the task offloading process

Specific to VFC, the stages in forming an informed SLA to determine to which fog node to offload the task are as follows:

1. Proactively determining when a source node vehicle needs to offload its task (hereafter considered as Requirement 1-R1)

As previously mentioned, VFC deals with compute-intensive and time-critical applications. To ensure that such applications are processed promptly, the SNv has to proactively determine when it will overload, which task/s will need to be processed when, and thus which task/s need to be offloaded [24]. These are important considerations that the SNv needs to determine before deciding to which TNv it should offload. The ability of a SNv to meet this requirement will depend on its ability to implement intelligent approaches and algorithms that use prior knowledge and predict its own workload, which in turn assists in reliable offloading decision making in terms of how much and when to offload [23], [25], [26].

2. Incentivizing the fog node vehicles (FNvs) to share information in various criteria that will enable a source node vehicle (SNv) to make an informed selection decision (hereafter considered as Requirement 2 - R2)

Before a task can be offloaded, a reliable TNv must be selected. In VFC, many factors such as the availability of resources at the TNv at the time of offloading, TNv mobility, and their trustworthiness inform this selection process. The availability of computing resources in the TNvs will ensure that the task is executed per the required QoS [27]. A vehicle's mobility too will determine if the target node is present within the vicinity of the requester node to process

its application [20]. Hence, there is a need for a mechanism that can provide either real-time or semi-real time information about the potential TNv in the above criteria, which will assist the SNv in the selection process. In other words, there is a need to incentivize TNvs to provide information that will assist them in participating in VFC. This leads to the following requirement of an intelligent mechanism aggregating a TNv's values in the required metrics and making recommendations to the SNv concerning the selection of the best TNv from the available ones.

3. The source node vehicle (SNv) uses the reputation of the target node vehicle (TNv) as a feedback mechanism to decide whether to form a future offloading request with it or not (hereafter considered as Requirement 3 - R3)

As the SNv depends on the TNv to complete its offloaded tasks, it needs to ensure that it forms an SLA only with nodes that have a record of committing to the offloaded tasks. This distinguishes a malicious TNv from a reliable or trusted node to reduce offloading failures. It is also essential to consider the high mobility of vehicles in the VFC network, as fog nodes may leave the system before the task is completed, interrupting the execution process [28]. One way of doing this is to determine the social reputation of the TNv after the task offloading process, which represents the extent of the commitment to the formed SLA within the prescribed deadline [29]. This assessment will help other SNvs select the appropriate TNv. The objective of this requirement is to compute this reputation value.

Thus, the task offloading process in VFC needs to address different requirements or else it could negatively affect the efficiency of task completion.

2.4 Systematic review protocol

This section explains the various steps used to identify the relevant papers which will be chosen for this SLR. By following the guidelines presented by Keele et al. [30], the following four-step process was conducted:

Step 1: Searching the literature: This step includes identifying the main data sources, defining the different search terms and the search procedures used to collect the relevant studies in the existing literature that address the aforementioned requirements R1–R3.

Step 2: Defining the inclusion and exclusion criteria: This step defines and uses the inclusion and exclusion criteria to guide the selection of the most relevant papers from the shortlisted ones.

Step 3: Undertaking a quality assessment of the shortlisted articles: In this step, four criteria (QA.1 – QA.4) were developed and applied to review each of the shortlisted articles.

Step 4: Shortlisting papers for the SLR and classifying them into three broad areas.

2.4.1 Searching the literature

The details of this step are as follows:

- Database used in the search process: The search process in this SLR was conducted based on the following literature sources:
 1. IEEE Xplore Digital Library (<http://ieeexplore.ieee.org>).
 2. Scopus database (<https://www.scopus.com>).
 3. ProQuest Science and Technology (www.proquest.com/).
 4. Web of Science database (www.webofknowledge.com).

These databases were selected primarily because they provide enough coverage of the literature that is relevant for this SLR.

- Search terms used: Key terms were used to search for the literature from the databases. These terms relate to capturing the relevant information from the research questions. Table 2.1 shows the main categories and keywords used in this SLR.

Table 2.1 : Search terms

Search category	Keywords
Vehicular Fog Computing	VFC, vehicular networks, vehicular fog networks
Task offloading	computation offloading, task offloading, task assignment

The final search term is formulated using the Boolean operators (AND, OR) to connect the keywords. To capture the relevant texts from the literature, quotation marks were used in the search query as follows:

(“Vehicular fog computing” OR “VFC” OR “vehicular networks” OR “vehicular fog networks”) AND (“task offloading” OR “computation offloading” OR “task assignment”).

- Search period and results:

Search period and results: The search was carried out on 13th June 2023. The search terms were used on each database to select journals and conference papers published between 2015–2023 as there was very little consideration given to the task offloading process in VFC before this period. The initial search process on the databases resulted in 791 papers being shortlisted. These papers were further filtered based on the inclusion and exclusion criteria defined in the next step.

2.4.2 Defining the inclusion and exclusion criteria

The details of this step are as follows:

- Paper selection criteria: To determine whether the paper must be included in, or excluded from the SLR, the following inclusion and exclusion criteria were applied to the initial search results.

Inclusion criteria: For a paper to be included in the SLR, it needs to meet the following inclusion criteria:

Criterion 1: It must have been published between 2015–2023.

Criterion 2: The paper's primarily focuses must be on VFC.

Exclusion criteria: If the paper meets the following criteria, it will be excluded from the SLR process:

Criterion 1: The paper is written in a language other than English.

Criterion 2: It is a thesis, book, or a book chapter.

Criterion 3: It does not focus on the algorithmic design of the task to assist in the offloading problem in VFC.

Criterion 4: It primarily focuses on cloud computing, fog or edge computing and does not differentiate it from VFC.

Criterion 5: It is a duplicate of similar studies.

- Paper selection procedure: Paper selection procedure: After applying the inclusion criterion related to the publication date, the total number of papers was reduced from 791 to 781. Of the shortlisted papers, 656 were from IEEE, 107 were from ProQuest, 15 were from Scopus and three were from Web of Science. Endnote was used to export the details of the shortlisted articles to

filter them further and remove duplicates. Then, three filtration stages were conducted:

In the first stage, the title and keywords of all the 781 studies were read, and the unrelated articles were excluded. In the case of an unclear title, the articles were moved to the second stage. This resulted in 131 articles being transferred to the second stage.

In the second stage, the abstracts of the papers were read, and those which matched the study's aim were moved to the third stage. This resulted in 85 articles being moved to the third stage.

In the final stage, the full texts of the shortlisted 85 papers were read and only 54 articles were chosen as relevant papers to our SLR. The paper selection stages are presented in Table 2.2.

Table 2.2 : Paper selection stages.

Database	Number of papers	Papers after title and keyword exclusion	Papers after reading abstracts	Final number of papers
IEEE	656	115	71	47
ProQuest	107	10	8	4
Scopus	15	5	5	3
Web of science	3	1	1	0
Total	781	131	85	54

2.4.3 Quality assessment of the shortlisted articles

In this step, four criteria (QA.1 – QA.4) were developed and applied to review each of the shortlisted articles. Table 2.3 presents the quality assessment criteria

used to assess each shortlisted article. If a paper met a quality assessment criterion, it was given a score of 1. Otherwise, it was given 0. Only those articles that met at least three criteria (i.e., received at least a score of 3 out of 4) were selected for further analysis. Table 2.4 shows the score of the shortlisted 54 articles in each of the quality assessment criteria. As seen from the table, 47 articles met the required score in the quality assessment criteria and were chosen for further analysis in the SLR.

Table 2.3 : Quality assessment criteria used.

QA.1	Does the paper cover relevant work and explore the research topics comprehensively?
QA.2	Is the algorithm\methodology used in the selected paper explained clearly?
QA.3	Does the paper describe and evaluate the results and applications clearly?
QA.4	Does the paper provide clear findings with justifiable results and conclusions?

Table 2.4 : Assessment of the studies against the four criteria.

Paper	QA.1	QA.2	QA.3	QA.4	Points
[31]	✓	✓	✓	✓	4
[27]	✓	✓	✓	✓	4
[20]	✓	✓	✓	✓	4
[32]	✓	✓	✓	✓	4
[22]	✓	✗	✗	✓	2
[33]	✓	✓	✓	✗	3
[34]	✓	✓	✓	✓	4
[28]	✓	✓	✓	✓	4
[21]	✓	✓	✓	✓	4

Continued on the next page

Table 2.4 (Continued)

Paper	QA.1	QA.2	QA.3	QA.4	Points
[35]	✓	✓	✓	✓	4
[36]	✓	✓	✓	✓	4
[25]	✓	✓	✓	✓	4
[26]	✓	✓	✓	✓	4
[37]	✓	✓	✓	✓	4
[38]	✓	✓	✓	✓	4
[39]	✓	✓	✓	✓	4
[40]	✗	✓	✗	✓	2
[41]	✗	✓	✓	✗	2
[42]	✓	✓	✓	✓	4
[43]	✓	✓	✓	✓	4
[44]	✓	✓	✓	✓	4
[45]	✓	✓	✓	✓	4
[23]	✓	✓	✓	✓	4
[46]	✗	✗	✓	✓	2
[47]	✓	✓	✓	✓	4
[48]	✓	✓	✓	✓	4
[49]	✓	✓	✓	✓	4
[50]	✗	✓	✓	✗	2
[29]	✓	✓	✓	✓	4
[51]	✓	✓	✓	✓	4
[52]	✓	✓	✓	✓	4
[53]	✓	✓	✓	✓	4

Continued on the next page

Table 2.4 (Continued)

Paper	QA.1	QA.2	QA.3	QA.4	Points
[54]	✗	✗	✓	✓	2
[11]	✗	✗	✓	✓	2
[55]	✓	✓	✓	✓	4
[56]	✓	✓	✓	✓	4
[57]	✓	✓	✓	✓	4
[58]	✓	✓	✓	✓	4
[59]	✓	✓	✓	✓	4
[60]	✓	✓	✓	✓	4
[61]	✓	✓	✓	✓	4
[62]	✓	✓	✓	✓	4
[63]	✓	✓	✓	✓	4
[64]	✓	✓	✓	✓	4
[65]	✓	✓	✓	✓	4
[66]	✓	✓	✓	✓	4
[67]	✓	✓	✓	✓	4
[68]	✓	✓	✓	✓	4
[69]	✓	✓	✓	✓	4
[70]	✓	✓	✓	✓	4
[71]	✓	✓	✓	✓	4
[72]	✓	✓	✓	✓	4
[73]	✓	✓	✓	✓	4
[74]	✓	✓	✓	✓	4
End of Table 2.4					

2.4.4 Shortlisted papers for the SLR and their classification into the broad areas

Table 6 shows the 47 shortlisted articles that are analyzed further against requirements R1–R3. These papers are grouped under one of three categories, namely task offloading in VFC, fog node selection and QoS assessment after execution. Sections [2.5.1](#) - [2.5.3](#) discuss the papers that fall under each category.

2.5 Analysis of the shortlisted papers against the requirements of task offloading in VFC

2.5.1 Analysis of papers in the category of task offloading in VFC

For efficient task offloading in VFC, the fog node needs to predetermine when it is going to be overloaded and needs to offload its task. Furthermore, it also needs to know how much workload and to whom it should offload its tasks. It is challenging for a fog node to answer these questions as it needs to predict its own workload, however these are key considerations in making timely offloading decisions [\[24\]](#). Different methods have been proposed in the literature to handle offloading decisions. One method is to use learning-based offloading, where a fog node learns the behaviour of future interaction. Ning et al. [\[45\]](#) proposed a deep reinforcement learning model based on the queuing theory to handle offloading decisions. Factors such as the arrival rate of moving vehicles, the arrival rate of offloading task flows and the number of parked vehicles are considered. Based on this, they proposed a redirection model to balance offloading task flows between the different nodes. Lee and Lee [\[23\]](#) proposed an offloading scheme using a deep learning neural network to study the mobility patterns of different vehicles to predict the availability of resources for future offloading decisions. In this proposed approach, the proximal policy optimization algorithm, which is one of the most recent deep reinforcement

learning methods, is used by integrating the recurrent neural network into the deep neural network. This is used to study previous trends in allocating the available resources in the VFC environment. Liao et al. [58] proposed an online learning-based framework for intelligent task offloading in which vehicles learn to find the optimal task offloading strategy with the least delay. The learning process is based on queuing delay, handover cost, and the trustworthiness of the available vehicular fog nodes. Kazmi et al. [69] proposed a framework that utilizes a deep reinforcement learning mechanism. In the proposed deep reinforcement learning-based framework, an agent interacts with its environment and learns to take actions based on the observed states within this environment. The agent is trained to make decisions on how to offload, schedule tasks, and allocate resources based on factors such as available computational resources, energy constraints, and network conditions. Sarkar and Kumar [70] proposed a delay-aware intelligent task offloading strategy for vehicular fog computing networks. The proposed strategy uses deep reinforcement learning and Markov decision processes to make intelligent task offloading decisions based on the current state of the network. Vehicle mobility and communication bandwidth constraints are considered in the proposed strategy to minimize the overall network latency. Wei et al. [71] proposed a multi-agent deep reinforcement learning approach to improve resource utilization in the VFC environment. The proposed approach addresses the challenges of many-to-many task offloading in dynamic vehicular environments using a partially observable Markov decision process (POMDP) to formulate the offloading process. Gao et al. [72] used a multi-agent reinforcement learning approach to address the problem of task offloading and resource allocation in heterogeneous vehicular fog computing environments where multiple intelligent agents, representing vehicles, learn to make collaboratively offloading decisions based on their local observations and interactions with neighbouring agents. In the proposed approach, a Transformer-based long sequence forecasting network (TLSFN) is used

to predict the current and future task queuing delay of the edge servers to manage the future task processing competition information. In [73], the authors used deep reinforcement learning (DRL) algorithms to propose a federated learning framework to provide a decentralized task allocation. Using the proposed approach, there is no need for a central entity and each vehicle can learn and make independent decisions based on its local observations and interactions with the environment. The proposed framework consists of two main components: the local learner that embedded within each vehicle to employ DRL techniques to help the vehicle learn an optimal task allocation policy based on its observations, and the coordinator that acts as a communication medium among vehicles to facilitate knowledge sharing and coordination without the need to access the data of each vehicle directly.

Other methods are proposed to enable a fog node to offload its computation tasks when it reaches its capacity limit. For example, Lin et al. [39] proposed an offloading scheme based on the workload and capacity of the available fog nodes using a greedy algorithm. In the proposed approach, the edge node will offload its task to the available vehicular nodes when there is no more capacity to execute the task locally. Wang et al. [37] and Ye et al. [49] proposed resource allocation algorithms to allocate the available resources of fog nodes during task offloading. Wu et al. [28] and Wu et al. [21] handled the offloading issue using a discounted semi-Markov decision process (SMDP) and an iterative algorithm. While such algorithms assist in offloading tasks, a key drawback of learning-based studies is that the offloading decision focuses on predicting the fog nodes' mobility to help the overloaded fog node decide where to offload its tasks. While this is beneficial, it does not assist the overloaded fog node in determining beforehand when it will be overloaded and thus enable it to make proactive decisions. Liu et al. [66] proposed a distributed algorithm using a combination of fog computing and cloud computing to efficiently offload tasks in vehicular networks. Based on network conditions and resource availability, the

algorithm optimizes task execution by offloading tasks to fog nodes or cloud servers.

The existing approaches adopt a reactive-based offloading process. The requester node will not start to offload its computation task until no more resources are available to execute the task locally. This will increase the latency and offloading cost in the case of an overload. To avoid this, prior knowledge of when a node will offload is needed which will also assist in reducing other metrics such as cost and energy consumption. Such an approach has not been proposed in the literature.

2.5.2 Analysis of papers in the category of fog node selection

Selecting the most appropriate target fog node will determine how efficient the execution result will be. Different mechanisms are proposed in the selected literature to choose the most appropriate target fog node for task offloading.

Using different selection criteria to select a fog node

Approaches in this category select a fog node based on criteria such as their distance from the requester node, availability in the area or the time needed to execute the tasks. For example, Zhu et al. [31] proposed a dynamic task allocation solution called Folo for task allocation across stationary and mobile fog nodes. In the proposed approach, selecting the target fog node depends on estimating the service time by the infrastructure fog node and choosing the fog node with the shortest service time. In [27], the authors proposed a distributed task offloading scheme called Chameleon. The proposed approach based on fog node workload observations takes high-resolution images within specific latency constraints. Selecting which target fog node to offload depends on choosing the node with the shortest path and the observed node workload at the end of each offloading process. Rahman et al. [38] proposed a context-aware opportunistic offloading scheme in which the most suitable vehicle to execute the task is selected based on the direction, speed and delay of

all vehicles available in the requester range. These proposed selection mechanisms assume that fog node vehicles are available at any time to execute the offloaded tasks. This assumption may be impractical in a real-world scenario as some fog nodes may be available but unwilling to accept and complete offloaded tasks. In Liu et al. [66], the neighbour selection phase was one of the three phases of the proposed algorithm for efficient task offloading. In this phase, vehicles identify nearby fog nodes and exchange information to build a neighbour table, which contains the details of neighbouring fog nodes and their associated attributes. In [70], the authors proposed a delay-aware task offloading strategy that considers the available resources of individual fog nodes and allocates them to requesting client vehicles in proximity.

Learning-based mechanisms are also used to propose a selection mechanism for task offloading. For example, Rejiba et al. [44] proposed an advice-based learning method in which the best performing vehicles are selected based on learning their performance from a neighbour RSU who already knows those vehicles. Zhao et al. [33] used the deep reinforcement learning (DRL) method to allocate vehicles' resources based on their incentive mechanisms to reduce task offloading conflicts that occur due to the simultaneous offloading decisions. In their proposed approach, a queuing model is used to sort offloading choices according to the accumulated rewards won by those vehicles as incentives for executing other vehicles' tasks. Lee and Lee [23] combined a heuristic algorithm with reinforcement learning (RL) to allocate fog resources to vehicles' applications. They use a deep recurrent neural network to get the patterns of the availability of vehicles and RSU resources and then use these patterns to select fog nodes that can handle the offloaded task.

Table 2.5 : Papers that meet the quality evaluation criteria from Step 3 of the selection process.

Paper	Paper title	Category
[31]	Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing	Fog node selection
[27]	Chameleon: Latency and Resolution Aware Task Offloading for Visual-Based Assisted Driving	Fog node selection
[20]	Task Offloading for Vehicular Fog Computing under Information Uncertainty: A Matching Learning approach	Fog node selection
[32]	Reliable Task Offloading for Vehicular Fog Computing Under Information Asymmetry and Information Uncertainty	Fog node selection
[33]	Contract-Based Computing Resource Management via Deep Reinforcement Learning in Vehicular Fog Computing	Fog node selection
[34]	Computation Resource Allocation and Task Assignment Optimization in Vehicular Fog Computing: A Contract Matching Approach	Fog node selection
[28]	A Task Offloading Scheme in Vehicular Fog and Cloud Computing System	Task offloading decision
[21]	Delay-Sensitive Task Offloading in the 802.11p-Based Vehicular Fog Computing Systems	Task offloading decision
[35]	Efficient Task Completion for Parallel Offloading in Vehicular Fog Computing	Fog node selection
Continued on the next page		

Table 2.5 (Continued)

Paper	Paper title	Category
[36]	Exploiting Moving Intelligence: Delay-Optimized Computation Offloading in Vehicular Fog Networks	Fog node selection
[25]	Optimal Task Allocation in Vehicular Fog Networks Requiring URLLC: An Energy-Aware Perspective	Fog node selection
[26]	Toward Dynamic Computation Offloading for Data Processing in Vehicular Fog based F-RAN	Fog node selection
[37]	Application-Aware Offloading Policy Using SMDP in Vehicular Fog Computing Systems	Task offloading decision
[38]	Context-aware opportunistic computing in vehicle-to-vehicle networks	Fog node selection
[39]	Cost Minimization with Offloading to Vehicles in Two-tier Federated Edge and Vehicular-Fog Systems	Task offloading decision
[42]	A Low-Latency and Massive-Connectivity Vehicular Fog Computing Framework for 5G	Fog node selection
[43]	Securing Parked Vehicle Assisted Fog Computing With Blockchain and Optimal Smart Contract Design	Fog node selection QoS assessment
Continued on the next page		

Table 2.5 (Continued)

Paper	Paper title	Category
[44]	Computation Task Assignment in Vehicular Fog Computing: A Learning Approach via Neighbour Advice	Fog node selection
[45]	Deep Reinforcement Learning for Intelligent Internet of Vehicles: An Energy-Efficient Computational Offloading Scheme	Task offloading decision
[23]	Resource Allocation for Vehicular Fog Computing using Reinforcement Learning Combined with Heuristic Information	Task offloading decision
[47]	Mobility Prediction-Based Joint Task Assignment and Resource Allocation in Vehicular Fog Computing	Fog node selection
[48]	Multi-Destination Computation Offloading in Vehicular Networks	Fog node selection
[49]	Processing capability and QoE driven optimized computation offloading scheme in vehicular fog based F-RAN	Fog node selection Task offloading decision
[29]	Blockchain-Based Reputation Management for Task Offloading in Micro-Level Vehicular Fog Network	Fog node selection QoS assessment
Continued on the next page		

Table 2.5 (Continued)

Paper	Paper title	Category
[51]	VFC-Based Cooperative UAV Computation Task Offloading for Post-disaster Rescue	Fog node selection
[52]	Fog Computing Model and Efficient Algorithms for Directional Vehicle Mobility in Vehicular Network	Fog node selection
[53]	Mobile Vehicles As Fog Nodes For Latency Optimization In Smart Cities	Fog node selection
[55]	A Deadline-Aware Offloading Scheme for Vehicular Fog Computing at Signalized Intersection	Fog node selection
[56]	Adaptive Offloading for Time-critical Tasks in Heterogeneous Internet of Vehicles	Fog node selection
[57]	An Infrastructure-Assisted Workload Scheduling for Computational Resources Exploitation in the Fog-Enabled Vehicular Network	Fog node selection
[58]	Blockchain and Learning-Based Secure and Intelligent Task Offloading for Vehicular Fog Computing	Fog node selection Task offloading decision QoS assessment
[59]	A Novel Contract Theory-Based Incentive Mechanism for Cooperative Task-Offloading in Electrical Vehicular Networks	Fog node selection Task offloading decision
Continued on the next page		

Table 2.5 (Continued)

Paper	Paper title	Category
[60]	An Incentive Mechanism for Computing Resource Allocation in Vehicular Fog Computing Environment	Fog node selection Task offloading decision
[61]	Energy-Latency Tradeoff for Dynamic Computation Offloading in Vehicular Fog Computing	Fog node selection Task offloading decision
[62]	Mobility Aware Blockchain Enabled Offloading and Scheduling in Vehicular Fog Cloud Computing	Fog node selection Task offloading decision
[63]	Priority-Aware Task Offloading in Vehicular Fog Computing Based on Deep Reinforcement Learning	Fog node selection Task offloading decision
[64]	Real-time Task Offloading for Data and Computation Intensive Services in Vehicular Fog Computing Environments	Fog node selection Task offloading decision
[65]	Fuzzy Reinforcement Learning for energy efficient task offloading in VFC	Fog node selection
[66]	A Distributed Algorithm for Task Offloading in Vehicular Networks with Hybrid Fog/Cloud Computing	Fog node selection Task offloading decision
Continued on the next page		

Table 2.5 (Continued)

Paper	Paper title	Category
[67]	Adaptive-Learning-Based Vehicle-to-Vehicle Opportunistic Resource-Sharing Framework	Fog node selection
[68]	ARTNet: Ai-Based Resource Allocation and Task Offloading in a Reconfigurable Internet of Vehicular Networks	Fog node selection
[69]	Computing on Wheels: A Deep Reinforcement Learning-Based Approach	Task offloading decision
[70]	Delay-aware Intelligent Task Offloading Strategy in Vehicular Fog Computing	Task offloading decision
[71]	Fast Adaptive Task Offloading and Resource Allocation via Multiagent Reinforcement Learning in Heterogeneous Vehicular Fog Computing	Fog node selection Task offloading decision
[72]	Dynamic Many-to-Many Task Offloading in VFC: A Multi-Agent DRL Approach	Task offloading decision
[73]	Federated Deep Reinforcement Learning-Based Task Allocation in Vehicular Fog Computing	Fog node selection Task offloading decision
[74]	Joint Offloading Decision and Resource Allocation for Vehicular Fog-Edge Computing Networks: A Contract-Stackelberg Approach	Fog node selection Task offloading decision
End of Table 2.5		

Similarly, Vemireddy and Rout [65] combined RL with the fuzzy logic-based heuristic algorithm to propose an allocation approach to enhance the selection process of fog nodes. Fuzzy logic is used to calculate vehicles weights, and the agent uses reinforcement learning to learn vehicle scheduling policy. Reinforcement learning is also used by Gao et al. [72] to propose a decentralized task offloading method based on a transformer and policy decoupling-based multiagent actor-critic where each mobile vehicle agent can select the optimal computing node and allocate the corresponding resources for the arrival task. Ning et al. [45] used DRL based on queuing theory to comprise the arrival rate of both moving vehicles and offloading tasks with the number of parked vehicles. They then take the offloading decision based on those variables. Huang et al. [43] and Liao et al. [58] proposed learning-based intelligent task offloading frameworks where all vehicles are pre-registered on a blockchain-based VFC system. The available vehicle is selected by executing a smart contract with it. Iqbal et al. [29] used a blockchain to maintain reputation scores to choose the best vehicle to handle the offloaded task. Zhou et al. [36] proposed an adaptive learning-based task offloading algorithm. The authors developed an optimal policy for task assignment using task replication to handle incomplete tasks and allocate the available computing resources to them in the VFC system. In [67], the authors used an extended version of the multi-armed bandit algorithm with additional features, such as input aware, occurrence, location, and direction aware to propose an adaptive-learning-based task offloading mechanism that selects the most suitable vehicle to handle the task within the collaborative environment. The selection of the most suitable vehicle is based on speed, direction, and reward. The reward is calculated based on the offloading delay and the available CPU frequency of the candidate vehicles. The proposed algorithm allows vehicles to learn the performance of nearby vehicles without an explicit exchange of control messages and to learn from the mistakes made in the previous task offloading decision cycle.

In [68], the Markov decision process is used to select a suitable fog node for task offloading based on the current state of the fog nodes and the size of the task following each node task arrival. An approximate next state is obtained by computing the appropriate reward value for selecting a fog node for task computation. POMDP is used in [71] to formulate the trading process during many-to-many task offloading.

Other studies such as [20], [32], [34], [42], [51] and [52] used a matching algorithm to select the target node based on the preference list of the requester. However, such selection mechanisms may delay the process of task offloading when no matching nodes are available to carry out the offloaded tasks.

Prediction methods were used by Wu et al. [47] and Yang et al. [55] to predict vehicles' mobility to select a proper target fog node based on its future mobility. While this assists the nodes to guarantee their availability at the time of execution, it does not help in assuring that the available fog nodes will have sufficient computational resources to carry out the offloaded tasks.

Researchers have used classification and scheduling techniques to schedule the different tasks coming from other vehicles and find suitable vehicles in the required range that have enough resources to process these tasks within a minimal completion time [39], [48], [53], [56], [62] and [64].

Other types of approaches use allocation schemes based on different algorithms to select a node. For example, Wu et al. [21] proposed an allocation scheme to allocate the resource units (RUs) of vehicles for the offloading tasks based on SMDP solved by an iterative algorithm. Similarly, the Markov decision process (MDP) was used by Shi et al. [63] to handle the problem of task allocation in a dynamic VFC environment where a DRL method based on a soft actor critic was used to solve this problem. In their proposed task allocation scheme, a service vehicle is selected based on task priority and the service availability of vehicles. Liu et al. [25] studied

the energy-aware task allocation problem in vehicular networks. They developed a strategy to choose the best receiver fog node by calculating their capacities and comparing them with the required computation task. This information is used to select the optimal fog node that satisfies either energy efficiency or consumption. Ye et al. [26] proposed a resource allocation strategy for F-RAN to choose the optimal fog node by reducing the processing duration of the offloaded task. Wang et al. [37] proposed an application-aware allocation strategy that considers the delay requirements and priority of the computation task while selecting the target fog server. The proposed model uses a value iteration algorithm to maximize the long-term reward of the VFC system during the selection process. Ye et al. [49] proposed a hybrid fog architecture to combine F-RAN with VFC to handle the problem of resource constraints in eRRHs in IoV and then proposed an optimization algorithm for resource allocation to decrease task execution time during the offloading process. In the proposed scheme, deep learning methods enhance a heuristic algorithm to allocate the available resources of vehicles for the offloading process. Heuristic-based resource allocation was used by Yadav et al. [61] to propose an energy-efficient dynamic computation offloading and resource allocation scheme to address the issue of energy consumption and service latency in VFC systems. This proposed scheme is used to find the most efficient vehicular node to handle the offloading tasks. Contract theory is another method used by Kazmi et al. [59] for fog node selection, where RUS selects the available vehicle that matches its preferences outlined in the contract. Nazih et al. [60] also used contract theory to allocate vehicles' resources and manage the relationship between service operators and vehicles. Other methods, such as the bi-dimensional selection framework [35], are proposed using the hidden Markov model (HMM) to ensure the availability of fog resources by estimating the association states and the communication rate of vehicles. Then, the computation perception is completed using the Markov chain method. In [74], the authors

proposed an algorithm based on Stackelberg game and contract mechanism that involves two stages: an offloading decision stage and a resource allocation stage. As part of the offloading decision stage, the mobile user determines whether to offload tasks to the fog or edge servers according to their associated costs, and the optimal fog nodes are then selected according to these costs. During the resource allocation stage, the service provider tries to maximize its revenue by allocating computing resources while considering fairness among multiple users.

While the approaches detailed above assist in finding the target fog node, they assume that the available target fog nodes will have enough resources to execute the offloaded tasks. They do not propose mechanisms to guarantee or check if the available fog nodes are reliable and have sufficient computation resources to complete the intended tasks.

Approaches to deal with missing information in relation to the criteria required to select a fog node

In the VFC network, due to their mobile nature, the status of fog node vehicles, such as their available resources, vary with time. When there is no prior knowledge of the available fog nodes that are ready to share their idle resources, the uncertainty, complexity and time in selecting target fog nodes increases, conflicting with the latency constraints [31]. Only a few approaches have been proposed in the selected literature to assist in making an offloading decision when there is a lack of operational information about fog nodes.

Some approaches use learning-based methods to handle the uncertainty issue. For example, Liao et al. [20] and Zhou et al. [32] proposed a learning-based matching algorithm using the multiarmed bandit (MAB) framework to motivate vehicular fog servers (VFSs) to share their resources under the condition of information uncertainty where the side information of VFSs is unknown to user vehicle (UV). In the

proposed method, each UV sends its preference list to the edge server when offloading, which searches for any VFS that matches the list. Liao et al. [58] developed a learning-based intelligent offloading scheme named QUOTA-UCB for task offloading where queuing-delay awareness, handover cost awareness, and trustfulness awareness are achieved without proper knowledge of the fog node vehicles' information. A learning-based approach is also used in [67] and [69] to handle the uncertainty of fog nodes during the task offloading process. [44] adopted multi-armed bandit (MAB) models in which RSU learns the behaviour of vehicles passing its range. It then advises other RSUs that do not have enough information and helps them make their offloading decision. Xie et al. [35] used the hidden Markov model (HMM) and the Markov chain to deal with the incomplete information of fog nodes. HMM is used to estimate the association states and communication rate when fog node vehicles are in the coverage range of the RSU, and the Markov chain is used to complete computation perception. In [68], the Markov decision process is used to identify the current state of the fog nodes and the size of the task during each offloading process. Yang et al. [55] handled the uncertainty problem by developing a method to predict the future location of vehicles. This method assists them in selecting the proper fog service provider that will achieve a maximum success rate of offloading. Kazmi et al. [59] proposed a framework for task offloading under information asymmetry. In the proposed approach, RSU offers a collection of contracts for vehicles that seek to share their resources while their preference lists are unknown.

Even though several solutions have been proposed in the selected literature to handle the uncertainty issue during task offloading, these solutions do not provide prior information about the trustworthiness of the fog nodes that will be available at the time of offloading with enough resources to handle incoming tasks. Having previous knowledge of such information will help reduce the latency and build trust between nodes during task offloading.

Approaches that encourage nodes to be a part of the selection process

In the VFC system, vehicles sharing idle computation resources with other resource-poor vehicles do not occur unconditionally and need specific incentives [33]. Therefore, to ensure the availability of target fog nodes at the time of task offloading, incentive mechanisms have been proposed by several studies to motivate fog nodes to share their idle resources and receive rewards. For example, Zhou et al. [32] proposed an approach to increase the expected utility of fog service provider nodes in which different contracts are formed according to the various resource sharing capabilities of fog service provider vehicles. Zhao et al. [33] and Zhou et al. [34] use contract theory to incentivize the different fog nodes to participate in task offloading where a contract is designed for each fog node that agrees to participate in the VFC system. This contract contains additional contract items, including the number of resources to be shared with the corresponding rewards. Kazmi et al. [59] also used contract theory to design an incentive mechanism framework for task offloading to handle the problem of selfish nodes and deal with vehicles' mobility. In their proposed approach, the RSU offers a reward to encourage vehicles to synchronize their moving speed in respect to each other during the task offloading process to avoid the interruption of task offloading that might occur due to vehicles mobility. Nazih et al. [60] combined the Stackelberg game with contract theory to design an incentive mechanism. The Stackelberg game manages the price given for resource sharing, and the contract theory is used to design contracts with different items for the participating vehicles.

Other approaches were proposed by researchers in the literature using other techniques such as the approach proposed in [71] that develops an incentive mechanism based on the coalitional game and mid-market-rate pricing mechanism to allow vehicles to organize steady and well-defined coalitions for resource orchestration. During the offloading process, the proposed mechanism aims to take into account

the individual rationality of each vehicle. The approach in [73] is to design a novel reward function that considers both the performance of the individual vehicle and the overall performance of the system. This reward function is designed to encourage vehicles to optimize their local task allocation and to collaborate with other vehicles to achieve a globally optimal solution.

Other researchers utilize the smart contract technique to provide incentive mechanisms for fog nodes. For example, [43] proposed parked vehicle assisted fog computing (PVFC), in which the resources of parked vehicles are used to process the required tasks of other vehicles. PVFC utilizes blockchain technology to record and audit the communication between the vehicle nodes to address the security and privacy challenges arising from a centralized environment. Based on the recorded information, smart contracts are used to trigger activities that facilitate the formation and execution of offloading tasks between the service requester and service provider vehicles. Smart contract analysis is then undertaken to determine the payment to the node for processing the offloaded task using the Stackelberg game framework. [58] also utilized smart contracts to provide an incentive mechanism where the different fog nodes earn rewards for sharing their resources. In their proposed scheme, smart contracts ensure automatic task offloading where the transactions between vehicular nodes are settled based on identified vehicular behaviours. [29] proposed blockchain-based reputation management for task offloading, where an incentive is given to a fog node when it executes the offloaded task within predetermined deadline. The reward is given as a reputation value in the proposed incentive mechanism which is used in the next selection process. Another method was also introduced by [63] to incentivize vehicles to share their resources using a dynamic pricing scheme. The service price is determined at the time of selecting the service vehicle.

Game theory was also one of the approaches utilized to propose an incentive mechanism such as the mechanism proposed in [74] in which the authors combine the

Stackelberg game approach with the contract mechanism and proposed an approach that models the interaction between a mobile user (the leader) and a service provider (the follower). The mobile user aims to minimize its offloading cost and latency, while the service provider aims to maximize its revenue by allocating resources to multiple users effectively. A contract mechanism is designed to establish a win-win situation to ensure that both players, the leader and the follower, benefit from their interaction.

Although researchers have used incentive mechanisms, they lack detail on what rewards will be given to the nodes. Furthermore, they do not focus on defining what penalties should be given to a node if it fails to commit to the promised tasks. We discuss this issue in Section 6.3 as an open research gap that needs to be addressed.

2.5.3 Analysis of papers in the category of QoS assessment

In the VFC network, the requester fog node uses the available resources of the different target fog node vehicles. To assist the requester fog node in making an informed decision, it should know the trustworthiness of the target fog nodes to avoid offloading their tasks to selfish nodes. This can be achieved by adopting an assessment feedback mechanism after the task offloading process in which the requester fog node evaluates the target fog node's commitment to the promised QoS factors. In the selected literature, only three studies have used an assessment mechanism to evaluate the execution result of the offloaded task. Liao et al. [58] proposed a smart contract-based offloading method. A Merkle tree-based proof-of-computing check mechanism is used to verify the computation results where the transaction is automatically ended in the case of improper execution. Huang et al. [43] designed a smart contract to handle the different offloading processes for parked vehicles in VFC. A third-party method is selected to evaluate the execution result and determine whether the associated computation work is qualified. However,

the assessment proposed in this method will not help future offloading decisions as no assessment scores are given to fog service provider vehicles for future fog node selection decisions. While Iqbal et al. [29] give a reputation value as an incentive for a node to complete an offloaded task, this value is computed based on the historical performance of fog node vehicles by measuring the previous rewards given to this node after each successful execution of the offloaded task. However, using the reputation technique as an assessment mechanism to score the executed task and using this score for future fog node selection has not been widely covered in the selected literature.

Table 2.6 summarizes the selected studies. In the next section, we analyze the papers against the requirements defined in Section 1 and identify the gaps as open issues of future work.

Table 2.6 : Comparison of the selected papers against requirements R1-R3 for efficient task offloading.

Paper	Offloading decision		Uncertainty-driven	Fog node selection		After offloading assessment
	Reactive	Proactive		Fog node selection mechanism	Incentive mechanism	
[31]	✓		NA	Estimate the service time by the infrastructure fog node and choose the fog node with the shortest service time	NA	NA
[27]	✓		NA	Choose the nearest fog node to the client vehicle	NA	NA
[20]	✓		MI-based algorithm MV-UCB	Pricing-based matching algorithm	NA	NA
[32]	✓		ML mechanism using MAB framework	Pricing-based matching algorithm	Contract Theory using convex-concave procedure CCP	NA
Continued on the next page						

Continuation of Table 2.6						
Paper	Offloading decision		Fog node selection			After offloading assessment
	Reactive	Proactive	Uncertainty-driven	Fog node selection mechanism	Incentive mechanism	
[33]	✓		NA	DRL	Contract Theory-based	NA
[34]	✓		NA	Pricing-based matching algorithm	Contract Theory-based	NA
[28]	✓		NA	NA	NA	NA
[21]	✓		NA	Resource allocation using Semi-Markov decision process (SMDP)	NA	NA
[35]	✓		HMM Markov chain	A bi-dimensional selection principle using HMM and Markov chain	NA	NA
[36]	✓		NA	Learning-based using task replication	NA	NA

Continued on the next page

Continuation of Table 2.6						
Paper	Offloading decision		Fog node selection			After offloading assessment
	Reactive	Proactive	Uncertainty-driven	Fog node selection mechanism	Incentive mechanism	
[25]	✓		NA	By the proposed task allocation mechanism	NA	NA
[26]	✓		NA	By the proposed task allocation mechanism	NA	NA
[37]	✓		NA	By the proposed task allocation mechanism	NA	NA
[38]	✓		NA	Device selection and offloading module	NA	NA
[39]	✓		NA	Greedy scheduling algorithm using queuing model	NA	NA
[42]	✓		NA	Pricing-based matching approach	NA	NA
Continued on the next page						

Continuation of Table 2.6						
Paper	Offloading decision		Fog node selection			After offloading assessment
	Reactive	Proactive	Uncertainty-driven	Fog node selection mechanism	Incentive mechanism	
[43]	✓		NA	Smart contract	Game theory and smart contract	Smart contract
[44]	✓		MAB	Online learning	NA	NA
[45]	✓		NA	Learning-based	NA	NA
[23]	✓		NA	RL-based	NA	NA
[47]	✓		NA	Min-Max rule, prediction-based	NA	NA
[48]	✓		NA	Scheduling-based task assignment mechanism	NA	NA
[49]	✓		NA	Resource allocation mechanism using heuristic algorithm enhance by DL methods	NA	NA

Continued on the next page

Continuation of Table 2.6						
Paper	Offloading decision		Fog node selection			After offloading assessment
	Reactive	Proactive	Uncertainty-driven	Fog node selection mechanism	Incentive mechanism	
[29]	✓		NA	Reputation-based	Reputation-based	Reputation-based
[51]	✓		NA	Matching-based	NA	NA
[52]	✓		NA	Greedy-algorithm-based matching approach	NA	NA
[53]	✓		NA	Task allocation mechanism using heuristic task scheduling strategy	NA	NA
[55]	✓		Prediction-based	Mobility-prediction, Replication-based	NA	NA
[56]	✓		NA	Delay-Driven classification policy and resource driven division policy	NA	NA

Continued on the next page

Continuation of Table 2.6						
Paper	Offloading decision		Fog node selection			After offloading assessment
	Reactive	Proactive	Uncertainty-driven	Fog node selection mechanism	Incentive mechanism	
[57]	✓		NA	Greedy scheduling algorithm	NA	NA
[58]	✓		Learning-based	Learning-based and smart contract	Smart contract	Merkle Tree-based proof-of-computing check mechanism
[59]	✓		Contract theory-based	Contract theory-based	Contract Theory-based	NA
[60]	✓		NA	Contract theory with Stackelberg game	Task allocation using Contract Theory	NA
[61]	✓		NA	Heuristic-based resource allocation	NA	NA
[62]	✓		NA	Task scheduling	NA	NA

Continued on the next page

Continuation of Table 2.6						
Paper	Offloading decision		Fog node selection			After offloading assessment
	Reactive	Proactive	Uncertainty-driven	Fog node selection mechanism	Incentive mechanism	
[63]	✓		NA	MDP based on task priority and service availability	Dynamic pricing scheme	NA
[64]	✓		NA	Task classification using resource-driven division policy	NA	NA
[65]	✓		NA	RL-based scheduling algorithm combined with fuzzy logic based greedy heuristic	NA	NA

Continued on the next page

Continuation of Table 2.6						
Paper	Offloading decision		Fog node selection			After offloading assessment
	Reactive	Proactive	Uncertainty-driven	Fog node selection mechanism	Incentive mechanism	
[66]	✓		NA	Vehicles identify nearby fog nodes and exchange information to build a neighbour table	NA	NA
[67]	✓		Learning-based	Learning-based	NA	NA
[68]	✓		Markov decision process	Markov decision process	NA	NA
[69]	✓		Learning-based	NA	NA	NA
[70]	✓		DRL-based	Allocating computing resources to service nodes based on their availability and proximity to client vehicle	NA	NA

Continued on the next page

Continuation of Table 2.6						
Paper	Offloading decision		Fog node selection			After offloading assessment
	Reactive	Proactive	Uncertainty-driven	Fog node selection mechanism	Incentive mechanism	
[71]	✓		DRL	A partially observable Markov decision process (POMDP)	Coalitional game and mid-market-rate pricing mechanism	NA
[72]	✓		RL	Multiagent reinforcement learning (transformer and policy decoupling based multiagent actor-critic)	NA	NA
[73]	✓		RL	Reinforcement Learning	Reward function	NA
[74]	✓		Contract mechanism	A cost-based decision process using Stackelberg game and contract mechanism	Stackelberg game and contract mechanism	NA
End of Table 2.6						

2.6 Open research issues

From the discussion and analysis presented in Table 2.6, it can be seen that while many approaches address the issue of task offloading in VFC, the existing solutions still have gaps and open issues. Further work needs to be done in the following areas to address these gaps:

1. **The source node vehicle (SNv) proactively determines when to offload the task**

Table 2.6 shows that none of the approaches determine in a proactive way when the SNv needs to offload a task. Existing approaches consider a reactive approach which does not guarantee the timely processing of the tasks nor the availability of TNvs with the required resources. To have a proactive approach, prior knowledge of the future workload of the SNv is needed, which will ascertain when it will be overloaded and what task/s it needs to offload. Future research in this area should use workload prediction for the intelligent handling of the offloading decision and alleviate issues related to latency and timeliness completion constraints in the VFC environment.

2. **Dynamic update of information in a registry that assists source node vehicles (SNv) in the offloading process**

Due to the high mobility of vehicle nodes, there is a need for a registry that updates dynamically and assists the SNv to accurately know the specifics of the available fog nodes to which it can offload its tasks. While existing approaches in the literature select a fog node based on criteria such as their distance, pricing-based matching, learning-based mechanism, etc., they assume that the available fog nodes will agree to carry out the offloaded task.

In a real-world scenario, this may not always be possible. For example, even

though a fog node vehicle may have the available resources, it may not want to share these, which will affect the offloading efficiency and the time required to decide. This can be addressed by a registry that dynamically updates itself with the information needed by the requester node to make an informed offloading decision.

3. Providing incentives to target node vehicles (TNvs) to be a part of the selection pool

An essential factor for the success of the offloading process is the availability of computing resources in the form of fog node vehicles (FNvs) that can process the required tasks. One way this can be guaranteed is to incentivize the available nodes to participate in the offloading process. From Table 2.6, it can be seen that most of the existing approaches do not use incentive mechanisms and assume that fog nodes will be available at offloading and ready to take incoming tasks. Furthermore, while incentivizing, it is also essential to monitor any malicious behaviour exhibited by nodes to obtain rewards and introduce penalties if a node does not complete the offloaded tasks as promised. For example, Liao et al. [58] proposed an incentive mechanism that gives a coin as a reward for executing another node's tasks. However, this may lead to malicious behaviour among nodes to compete to obtain coins but not complete the tasks they have taken from the previous nodes.

To successfully handle the incentive issue in the VFC environment, researchers should look at utilizing game theory. Game theory can decide on incentives as task offloading is considered a multiplayer decision problem. The FNvs are competing to maximize their benefits. This approach can also be used to determine if FNvs are acting selfishly or in a malicious way and prevent this from occurring.

2.7 Limitations of this SLR

This SLR has several limitations that should be considered when viewing this chapter. Firstly, the selected studies were obtained from only four data sources, which are well-known databases in Engineering and IT. Secondly, the search strings used in this chapter are limited due to the time constraints of this research project, which may result in some related studies being missed. Thirdly, some issues related to task offloading in VFC were not covered in this SLR, including, but not limited to, security and privacy, load balancing, and service interruption during the offloading process. Finally, task offloading has applications in other domains such as fog computing [75]. As this chapter focuses only on VFC, we only consider those articles that match our inclusion criteria. Thus, it excludes research that discusses task offloading but is not related to VFC.

2.8 Conclusion

This SLR proposes three essential requirements that should be considered in the task offloading process in VFC. After a thorough review of the existing literature, this SLR analyses the selected literature against these three requirements and is the first attempt to address the issues observed in task offloading in a VFC system related to these requirements.

Based on the result of the analysis of the selected literature, the next chapter identifies the research problem, identifies the research question and formulates the research objectives.

Chapter 3

Problem Definition

3.1 Introduction

In the previous chapter, a comprehensive systematic literature review was conducted to identify the research gaps. Based on these identified research gaps, this chapter defines the research problem and identifies the research questions as well as the research objectives.

This chapter is organized as follows: Section 3.2 identifies the key terms and concepts used in this thesis. Section 3.3 identifies the research problem. Section 3.4 outlines the research questions and section 3.5 identifies the research objectives. Section 3.6 concludes this chapter.

3.2 Key definitions

In this section, the definitions of the key terms and concepts used in this thesis are as follows:

- a) Fog node vehicle (FNv): We define a FNv as any interested vehicle which downloads the iVFC application and completes the registration process to become an iVFC client and identify as a FNv.
- b) Source node vehicle (SNv): We define an SNv as one with a computationally intensive task or latency-sensitive task that needs to be offloaded to a fog-node vehicle for its execution.
- c) Target node vehicle (TNv): We define a TNv vehicle as one that has idle or

unused computation resources (such as compute or memory) and can share its idle computing resources to execute the other vehicle's tasks.

- d) Load or utilization capacity: We define the load or utilization capacity as the past utilization profile or capacity of a vehicle (in terms of its computational resources). The past load or utilization capacity profile can be used to make intelligent judgements about the future need for offloading.
- e) Task offloading: This is the process of transferring a task from a resource-constrained SNv to a more powerful device (such as a cloud server or TNv) that can handle the task more efficiently [16].
- f) Task: We define the term “task” in the task offloading process as a specific unit of computation or processing that needs to be performed by a FNv in the VFC system. This task can be any kind of computation, such as data processing, data analysis, or data storage, and can be initiated by any client node in the VFC system.
- g) Reactive-based task offloading: This is a strategy where tasks are offloaded to a TNv in response to a specific event. In this approach, the decision to offload a task is made when a need arises. For example, the client node vehicle decides to offload a task when it reaches its capacity and is unable to execute its tasks locally [13].
- h) Proactive-based task offloading: This is a strategy where tasks are offloaded based on future events or requirements. In this approach, the decision to offload a task to a suitable TNv is made proactively, based on predicted workload, where the SNv is a priori aware that it is going to be overloaded in the future. Proactive-based task offloading is typically planned ahead of time and is based on the expected future system state [13].

- i) Target node selection: We define target node selection as the process of choosing the most optimal TNv from a group of available fog nodes in a VFC system to offload a particular task. A target node is a physical vehicular-node that provides computing, networking, and storage services in a VFC system. Target node selection involves considering various factors, such as the computing capacity of the fog nodes (its CPU utilization and memory usage), and their proximity to the devices generating the tasks. The goal of target node selection is to find the best possible TNv that can perform the task efficiently and effectively.
- j) Incentive-mechanisms: These are mechanisms designed to encourage and motivate FNvs to participate in the VFC system during task offloading. Incentives involve offering participants rewards or benefits for their contributions. [17].
- k) Smart cars: These are vehicles that are equipped with advanced technology and connectivity features such as sensors, communication systems, and computing power to enable them to communicate with each other and with the environment to collect and process data to enhance safety, efficiency, and overall driving experience [76].
- l) Machine learning (ML): This is an artificial intelligence (AI) subfield that enables computer systems to automatically learn from and make predictions or decisions based on data by applying algorithms and statistical models, without the need for explicit instructions from humans [77].
- m) Deep Learning (DL): This is a subset of machine learning that involves learning and representing complex patterns in data using artificial neural networks with multiple layers. It allows the network to learn hierarchical representations of features [78].

- n) Fuzzy logic: This is a mathematical framework that deals with reasoning and decision making in situations that involve uncertainty, ambiguity, and imprecision. It is a form of logic that allows for degrees of truth instead of the usual binary true or false values in classical logic [79].
- o) Time-series: This is a set of observations or data points collected at regular intervals over time. The data points can be collected at equal or unequal time intervals [80].

3.3 Problem definition

In the context of VFC, the SNvs are mobile nodes with intensive computation tasks that need to be executed within a specified time frame. Executing these tasks locally on the source node is challenging due to the resource limitations of the node and the time constraints of the job. Therefore, offloading these tasks to another fog node can solve such problems. However, the offloading decision-making process is a complex task that initiates an agreement between the SNv with the computation tasks and the TNv that executes the offloaded tasks. The SNv is responsible for identifying the need to offload, identifying an appropriate TNv, communicating and negotiating with the TNv, offloading the task to the TNv, and receiving the completed task from the TNv.

One of the major challenges in task offloading is the decision as to where, how much, and when to offload tasks. The decision as to where to offload tasks involves determining the best TNv to offload the task to. The decision as to how much workload to offload is crucial in managing resource utilization, while the decision as to when to offload the task is critical in meeting the job's time constraints. The effectiveness of the task offloading decision depends on the accuracy of these decisions as well as the availability of fog nodes that are ready to share their computation resources.

However, the existing literature fails to provide intelligent approaches or mechanisms for carrying out the offloading decision-making process. Therefore, there is a need for intelligent approaches and mechanisms to improve the decision-making process in task offloading. These approaches should include a predictive mechanism to support the SNv to priori determine the future offloading decision, a TNv selection mechanism to help the overloaded SNv to select an optimal TNv for task offloading, and an incentive mechanism to encourage fog nodes to participate in the task offloading process.

In this thesis, to address the research gaps that were identified in the existing literature, we propose an intelligent framework for handling the task offloading process in a VFC system. This framework will help the SNv in the VFC system to priori identify the need to offload its next task, select the most optimal TNv to carry out the task and to incentivise fog nodes to participate in the task offloading process.

Based on the discussion above, the research problem that is addressed in this thesis is:

How to develop a framework that can intelligently handle the decision-making problem during task offloading in a VFC system to reduce latency and increase the quality of service (QoS)?

3.4 Research Questions

Based on the systematic literature review reported in chapter two and the research problem identified in section 3.2, the main research question is identified as:

How can the source node vehicle (SNv) proactively offload a task to the most optimal target node vehicle (TNv) to minimize execution delay in VFC?

The main research question is divided into the following research sub-questions:

3.4.1 Research Question 1 (RQ1)

How can we use the prior load of the source node to proactively predict the need to carry out task offloading in VFC?

3.4.2 Research Question 2 (RQ2)

How can incentive mechanisms be used to motivate fog node vehicles (FNvs) to take the incoming task while guaranteeing that the required level of participation is met?

3.4.3 Research Question 3 (RQ3)

How to best select the most optimal target node vehicle (TNv) for task offloading in which execution delay will be minimized?

3.4.4 Research Question 4 (RQ4)

How to evaluate and validate the effectiveness of our proposed method based on simulation results?

3.5 Research objectives

On the basis of the above research question and sub-questions, the research objectives are formulated as follows:

3.5.1 Research Objective 1

To develop a proactive-based task offloading methodology based on predictive techniques and the prior utilization-profile of the fog node vehicle (FNv).

To achieve this objective, we develop a predictive analytic model using machine learning methods (ML) to predict the future utilization capacity (i.e., the CPU utilization and the memory usage) of each FNv based on its prior CPU utilization and memory usage to help the node to have prior knowledge of when it is going

to be overloaded and when it needs to offload its next task. Due to the dynamic nature of fog nodes in the VFC environment, the workload of FNvs varies with time (i.e., time series workload). Therefore, we propose a framework that can handle the dynamic nature of time series data efficiently. Based on the predicted workload, each FNv will be able to proactively predict when it is going to be overloaded in the future.

3.5.2 Research Objective 2

To develop an incentive mechanism that considers a reward to motivate fog node vehicles (FNvs) to take the incoming task and a penalty to ensure that the required level of participation is met.

To achieve this objective, we develop a game theory-based incentive mechanism to encourage fog nodes to participate in the VFC system. The incentive mechanism helps to increase the level of participation in the VFC system. To develop our proposed incentive module, we choose game theory as the incentive mechanism to be used in our framework. Due to the selfishness and rationality of the vehicles that intend to increase their outcomes in the VFC environment, the VFC environment is considered to be a non-cooperative game. Therefore, in our framework, we consider vehicles as players in the game and we follow the solution concept of the Stackelberg Nash equilibrium game theory.

3.5.3 Research Objective 3

To develop an effective mechanism for the SNv to intelligently select an optimal TNv to handle the task.

To achieve this objective, we build a TNv selection model using three methods, statistical methods, a machine learning (ML) method and deep learning (DL) methods. Of the statistical methods, we use the analytic hierarchy process (AHP) method to

calculate the weights of the criteria of the TNvs (CPU utilization and memory usage) and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method for ranking the different TNvs based on their criteria. This ranking is based on the relative closeness of a particular alternative to the ideal solution. we use other methods, namely the ML method and the DL methods to compare the result with TOPSIS to choose the best performing method for selecting the most optimal TNv.

3.5.4 Research Objective 4

To validate and evaluate the developed methods using simulation experiments.

To achieve this objective, our proposed framework is validated by building a prototype software of the framework and implementing the approaches proposed to achieve objectives (1-3). This is done by conducting different experiments to test the effectiveness of our research objectives (research objective 1 to research objective 3) in addressing our research questions (research question 1 to research question 3).

3.6 Conclusion

In this chapter, the research problem is identified and the main research question and research sub-questions are presented. Based on the identified research problem and the outlined research questions, four research objectives are identified. This chapter also presented the definitions of the main key terms used through this thesis.

In the next chapter, an overview of the proposed solution to address the outlined research objectives (research objective 1 to research objective 4) is presented along with a detailed description of the research methodology used to design and develop the proposed solution.

Chapter 4

Research Methodology and Solution Overview

4.1 Introduction

In this chapter, the research methodology applied in this thesis is explained in detail. This methodology addresses the research gaps that were identified in the literature review in chapter two. This chapter also includes an overview of the proposed solution and how each research question is addressed.

This chapter is organized as follows: section 4.2 provides the definitions of the key terms used in the proposed solution. Section 4.3 gives an overview of the research methodology used to develop the proposed framework. A step-by-step overview of the proposed iVFC prototype is given in section 4.4. Solutions for research questions (RQ1-RQ4) are presented in sections 4.5, 4.6, 4.7 and 4.8, respectively. Finally, section 4.9 concludes the chapter.

4.2 Key definitions

In our iVFC proposed framework, we use the following key terms:

- a) Fog Server Node (FSN): a node that is part of the iVFC infrastructure and is specifically designed to provide computing and storage resources to vehicles and other devices on the road. It is typically located at the road side unit (RSU) and can communicate with nearby vehicles and devices using short-range wireless communication technologies like Wi-Fi or Bluetooth. It can provide various services, such as data caching, data processing, and data filtering, to improve the performance of the iVFC system.

- b) Fog Service Providers (FSPs): a number of different fog service nodes (also known as road side units (RSUs) in our proposed framework). Each one of these fog service nodes is part of the fog service consortium (FSC) and all of these fog service nodes collaborate and coordinate with each other and act as FSPs. Their job is to collect all the data originating from the vehicles and store them (and subsequently process them through the AI layer).
- c) Fog Service Consortium (FSC): is a subset of the iVFC fog service providers (FSPs). The primary purpose of the FSC is to provide a distributed approach for managing all the FSPs which are geographically dispersed. The FSC is responsible for activities such as adding a new fog node to the FSC, removing an existing fog node from the FSC, managing all the iVFCs such as enrolling a new iVFC and managing iVFC's access to the distributed fog repository. FSC works on the basis of consensus and decisions are arrived at based on majority vote.
- d) Fog Repository (FR): a storage system that is designed to store and manage data and applications that are used by fog node vehicles (FNvs) and devices. The FR provides a central location for storing and distributing data, software, and other resources, which can be accessed by FNvs and devices in the network. It is typically located in a centralized data centre and is connected to the FNvs and devices through high-speed networks. It can be managed and configured by the iVFC computing system administrators, who can control access and permissions for the stored data and applications.
- e) Roadside Units (RSUs): RSU in our proposed framework are the edge servers that are deployed close to vehicles to gather, process and store data in a timely manner.
- f) CPU utilization of FNv: represents the percentage of processing power that

is being used by a FNV's CPU at any given time. CPU utilization is an important metric for evaluating the performance and efficiency of a FNV, as it indicates how much of the available processing power is being utilized by the system. High CPU utilization can indicate that the FNV is processing a large amount of data or running complex computations and is about to become overloaded, which can lead to increased latency and decreased performance. On the other hand, low CPU utilization can indicate that the FNV is underutilized and has idle resources that can be used to process other FNVs' tasks.

- g) Memory usage of FNV: represents how much of a FNV's RAM is being used at any given time. Memory usage is considered an important metric to evaluate the performance and efficiency of a FNV. It indicates how much of the available memory resources are being used by the vehicular fog computing (VFC) system. High memory usage can indicate that the FNV is processing a large amount of data or running complex applications that require a significant amount of memory and is about to become overloaded. On the other hand, low memory usage can indicate that the FNV is not fully utilizing the available memory resources and has idle resources that can be used to process other FNVs' tasks.

4.3 Selected Research Methodology

To achieve the research objectives identified in chapter 3, the design science research methodology (DSRM) is used in our research. The DSRM approach is a widely used research methodology for validating and testing a newly designed prototype system [81].

The objective of this research is to develop an intelligent framework on top of VFC for task offloading. Subsequently, in our proposed framework, we propose a novel algorithm to provide proactive-based task offloading by developing three mod-

ules: predictive analytic, target fog-node selection and incentives for participation. To validate our proposed framework, we develop a prototype system using various software frameworks. Using the developed software framework, we simulate the VFC network and evaluate the algorithms that are proposed in this research to determine if the initial prototype system addresses the identified research objectives. This process will be iterated until the identified research objectives are achieved. Non of the existing research methodologies, except DSRM, are suited to develop a prototype or a system as proof of concept. furthermore, non of the exiting research methodologies, except DSRM, carry out lab-based testing as an approach to validate the developed proof of concept. Hence, we used DSRM in our research.

Given the focus of this research is on the development of a software framework, and the simulation and evaluation of VFC using the software framework, the DSRM is the most appropriate methodology to be used.

Based on the DSRM, our proposed solution implements the following steps in a nominal sequence. These steps are presented in Figure 4.1.

1. **Problem identification and motivation:** During this step, we conducted an initial literature review on VFC and related issues on resource management to build knowledge and to identify the research problem and motivation. Specifically, task offloading was identified as one of the resource management issues of VFC.
2. **Literature review:** In this step, we conducted a systematic literature review (SLR) based on the problem identified in the initial literature review. This SLR aims to identify the research gaps by conducting a critical analysis of the existing works that have been proposed in the literature to address the problem of task offloading in VFC. This step is documented in chapter 2 of this thesis.

3. **Objective of the solution:** In this step, we identified the research questions and objectives based on the outcomes from steps one and two. The main objective of this research is to develop an intelligent framework on top of VFC for task offloading. To achieve this objective, we proposed iVFC: a proactive-based framework for task offloading in VFC by developing three modules: a predictive analytic module to assist the participating FNvs in the VFC environment to have prior knowledge of the overloading decision, a target node vehicle (TNv) selection module to select the most optimal TNv to efficiently handle the offloaded task, and an incentive module to encourage FNvs to participate in the task offloading process to guarantee the availability of TNvs at the time of task offloading to handle the offloaded task. This framework helps to reduce execution delay during the task offloading process and improves the performance of the VFC system. This step is documented in chapter 3 of this thesis.
4. **Design and development:** In this step, the iVFC framework is built using various artificial intelligent methods. These artificial intelligent schemes are used to develop the solutions to research questions one to four. This step is documented in chapter five, six, and seven of this thesis.
5. **Demonstration:** In this step, the efficiency of the designed iVFC solution is demonstrated through the development of a prototype system to achieve each objective (research objective one to research objective three) to check the efficiency of the proposed iVFC solution in solving the problem identified in step 1. This step is documented in chapters four, five, and six as part of research objectives one to three.
6. **Evaluation:** In this phase, we evaluate and validate our proposed iVFC solution by conducting experiments and a number of metrics are used to decide

whether to continue to the next step or to go back to step three to improve the effectiveness of our proposed solution. This corresponds to research question four. This step is documented in the validation section in chapters four to six to evaluate and validate research objectives one to three.

- 7. Communication:** In this step, the outcomes of the previous steps are disseminated through publications in peer-reviewed journals and conferences. This step is iterated during this thesis.

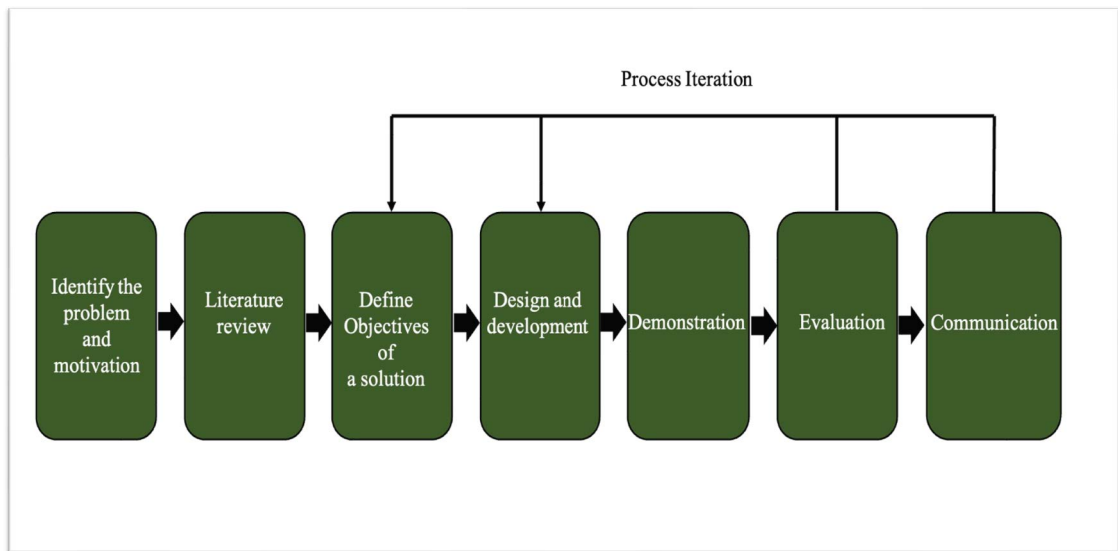


Figure 4.1 : The proposed Design Science Research Methodology

4.4 Overview of the proposed iVFC solution

This section discusses the overall solution of the proposed iVFC system that is developed to provide an intelligent framework on top of VFC for the task offloading process to achieve the research objectives identified in chapter 3.

4.4.1 Architecture of the iVFC

iVFC is an intelligent framework to support fog nodes during the task offloading process in the VFC environment. The architecture of the proposed iVFC, as shown

in Figure 4.2, consists of the following three layers:

Data collection Layer: This is the base layer that is responsible for collecting data from the different vehicular fog nodes and passing it to the nearest iVFC FSP. When each vehicle installs the iVFC application, the data layer is installed as well. There can be a number of FSPs within the iVFC FSP layer.

iVFC Fog Service Providers (FSPs) layer: This layer comprises a number of fog service nodes (also known as road side units (RSU) in our proposed framework). Each of these fog service nodes is part of the (FSC) and all of these fog service nodes collaborate and coordinate with each other and act as FSPs. Their job is to collect all the data originating from the vehicles and store them (and subsequently process them through the AI layer). In our research, the iVFC FSPs store all the information in the distributed FR. The AI intelligent modules and algorithms are run on the FSC.

FSC may be defined as:

$FSC = FN1, FN2, FN3, FN4, \dots, FNn$, where FN denotes as a Fog Node.

In the iVFC layer, there are many different FSPs, and FSC is a subset of FSPs which manages the membership of these FSPs such as their registration to the iVFC system, execution of the AI modules on top of their frameworks and so on. This means the FSC will manage the whole network of the FSPs.

AI layer: is used for performing the following three AI modules:

- Predictive analytic module to predict the need for offloading (corresponding to research question 1 (RQ1)).
- Incentives module for participation (corresponding to research question 2 (RQ2)).
- TNv selection module to determine which target node vehicle to select (corresponding to research question 3 (RQ3)).

Once the data is collected, it is processed in these three modules to obtain analytical insights. Figure 4.3 overviews the proposed iVFC conceptual framework where all vehicular fog nodes that become part of the iVFC (that is, they complete the registration process) are known as FNvs. Any node that needs to offload a task is known as a source node vehicle (SNv) and the node which agrees to receive and process the offloaded task is known as a target node vehicle (TNv).

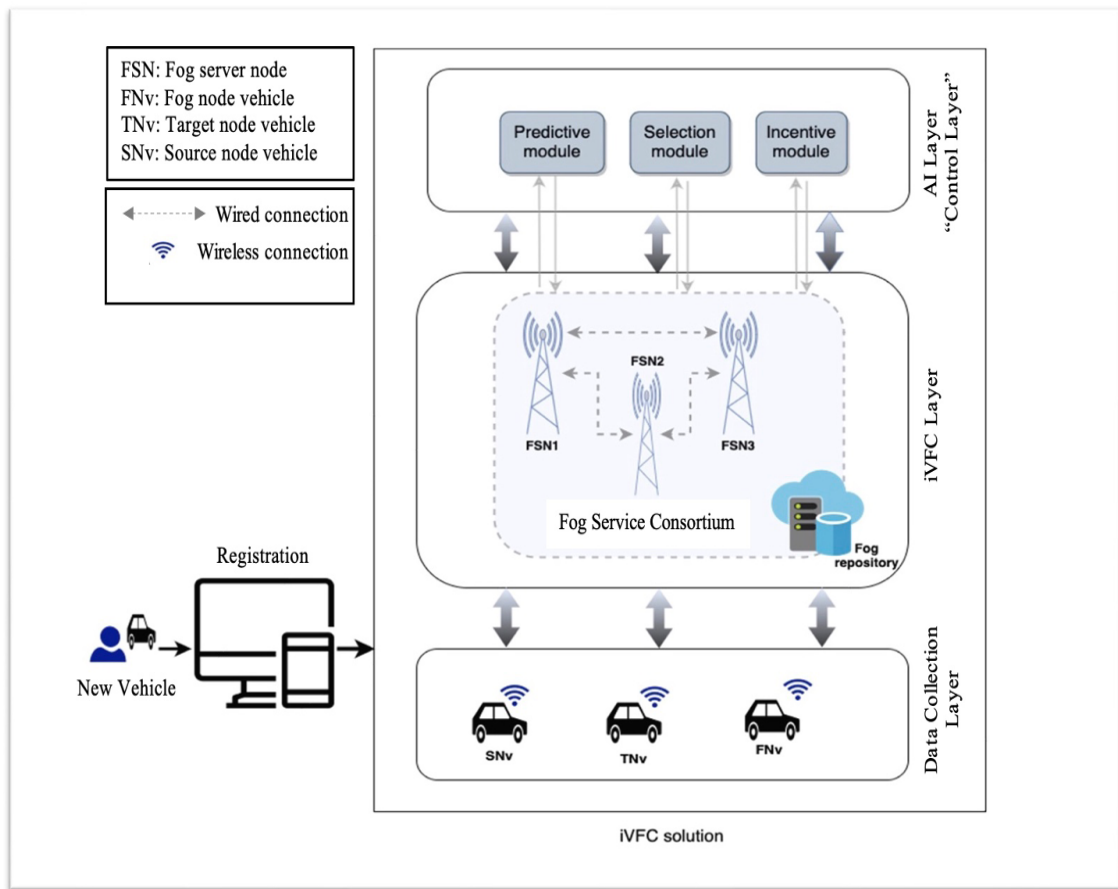


Figure 4.2 : The architecture of the proposed iVFC system

4.4.2 Overview of the proposed iVFC framework

The solution steps of the proposed iVFC and its working are as follows:

Step 1: Each interested vehicle downloads the iVFC application (becoming an

iVFC client and is identified as an FNV) and completes the registration process. The iVFC application is given access to the vehicles' global positioning system (GPS) coordinates. The following parameters are provided during the registration process: vehicle's ID, vehicle's actual CPU and memory, and the resources that the FNV is willing to share. Subsequently, the Client ID is generated by the FSP and is unique across all iVFC clients. The information of the registered FNVs is stored in the FR of the FSC.

Step 2: Once the registered vehicle becomes an iVFC client, its historical CPU utilization and memory usage is recorded periodically by the iVFC FSN and stored and periodically updated in the FR of the FSC. This historical data is used by the predictive analytic module to predict future CPU utilization and the future memory usage of each FNV to predict the overloading condition of FNVs to facilitate the proactive handling of task overloading.

Step 3: The iVFC fog server nodes constantly monitor the iVFC nodes to proactively and intelligently determine when they will be overloaded. When the iVFC fog server nodes identify that a FNV is about to be overloaded, they contact the vehicle on the iVFC client and the following information on the requested service is provided: the ID of the vehicle, service start time, service finish time, and the resources needed for the task. This information is uploaded through the iVFC system to the nearest FSN (which is the RSU in our proposed framework) and stored in the FR. The overloaded FNV is then known as a SNV after this step.

Step 4: The offloading request is handled by the FSN in the incentive module through the iVFC system to obtain the list of the TNVs that agreed to participate in the task offloading process. The incentive module works in parallel with the other modules (predictive analytic module and TNV selection module) to encourage fog nodes to participate in the task offloading process by offering them a reputation

value for participating and sharing their resources to execute other FNVs' tasks. In this step, when an offloading request is received, the FSN contacts all the available FNVs and provides information of the task execution time and the needed resources and offers them a reputation for sharing their resources.

Step 5: The FNVs which accepted the offered reputation value and agreed to participate are known as TNVs and a list of all the TNV is prepared by the incentive module and sent to the selection module for ranking.

Step 6: The selection of the most suitable TNV for handling the task occurs in the selection module after receiving the TNVs list. The service provider selection method is applied to rank the TNVs list based on their current CPU utilization and memory usage and the top TNVs is ranked from the lowest to the highest workload.

Step 7: From the selected TNV list, the most optimal (can be more than one and is usually the top of the list) TNV is chosen by the FSN to handle the task. Subsequently, the chosen TNVs will be ready to receive the agreed task via the FSP.

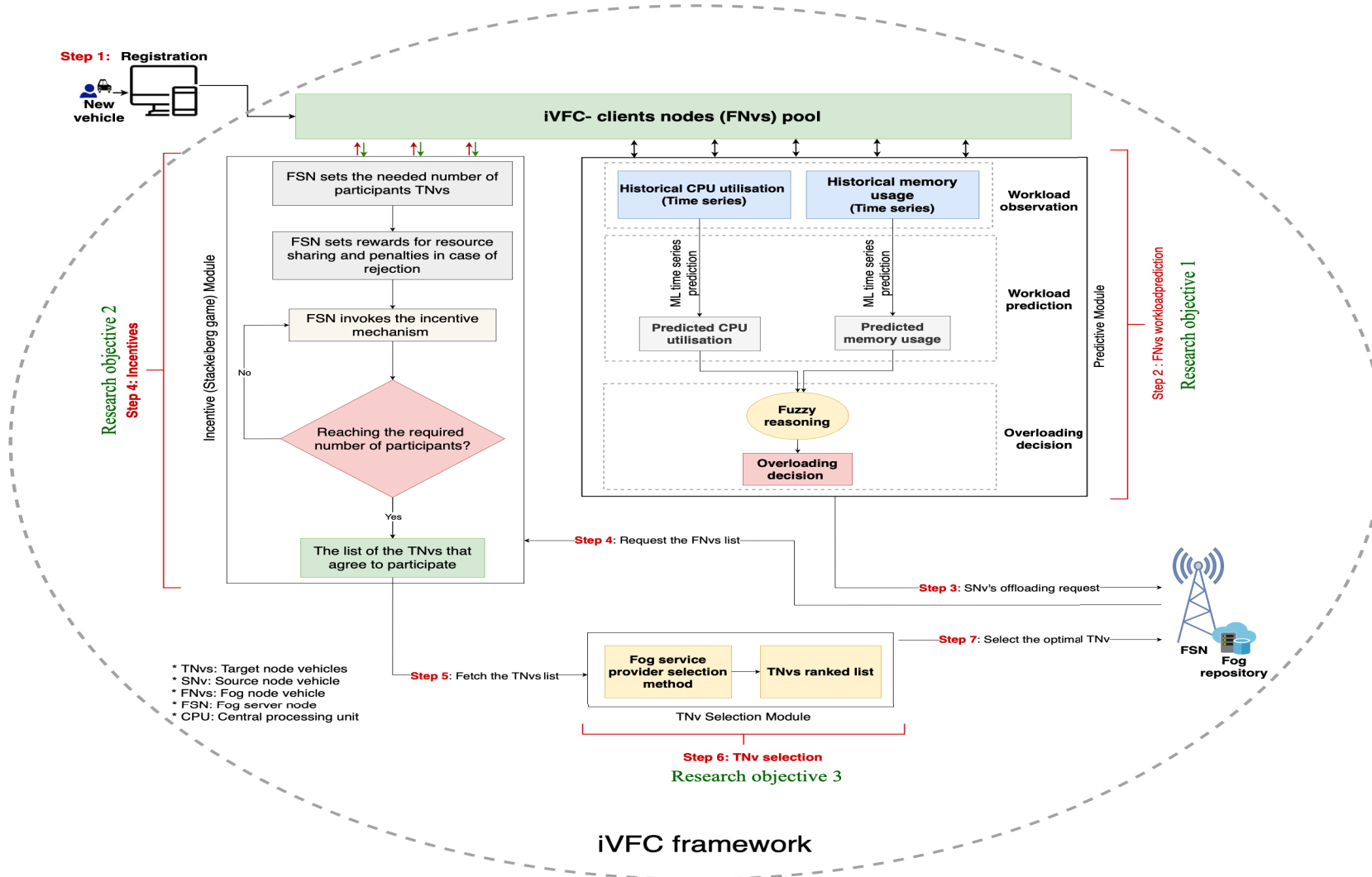


Figure 4.3 : Overview of the iVFC framework

4.5 Overview of the solution for research objective 1 (RO1)

RO1 is to develop a proactive-based offloading methodology based on a prior utilization-based prediction technique to accurately predict the various workload metrics (CPU utilization and memory usage) of FNVs running in a VFC environment. This means that the future workload of FNVs can be predicted based on historical observation.

To achieve this objective, we develop a predictive analytic module to predict the future CPU utilization and memory usage of each FNV based on its prior CPU utilization and memory usage. This prediction will help the node to have prior knowledge of when it is going to be overloaded and when it needs to offload its next task. Due to the dynamic nature of fog nodes in the VFC environment, the workload of the different FNVs varies with time (time series workload). Therefore, we propose a framework that can handle the dynamic nature of time series data efficiently. Based on the predicted workload, each FNV will be able to proactively predict when it is going to be overloaded in the future. Figure 4.4 shows the workflow of the solution for research objective 1 (the iVFC-predictive analytic module).

The framework of this objective comprises three phases: (1) workload observation to identify the workload patterns of each FNV (CPU utilization and memory usage); (2) workload prediction to predict the future workload of FNVs; and (3) overloading decision, joining the two prediction models to make an overloading decision. Figure 4.4 shows the working steps of the proposed iVFC-predictive analytic module and Figure 4.5 shows the three phases of the proposed framework.

Further explanation of the development process of the proposed iVFC-predictive analytic module is given in detail in chapter 5 of this thesis.

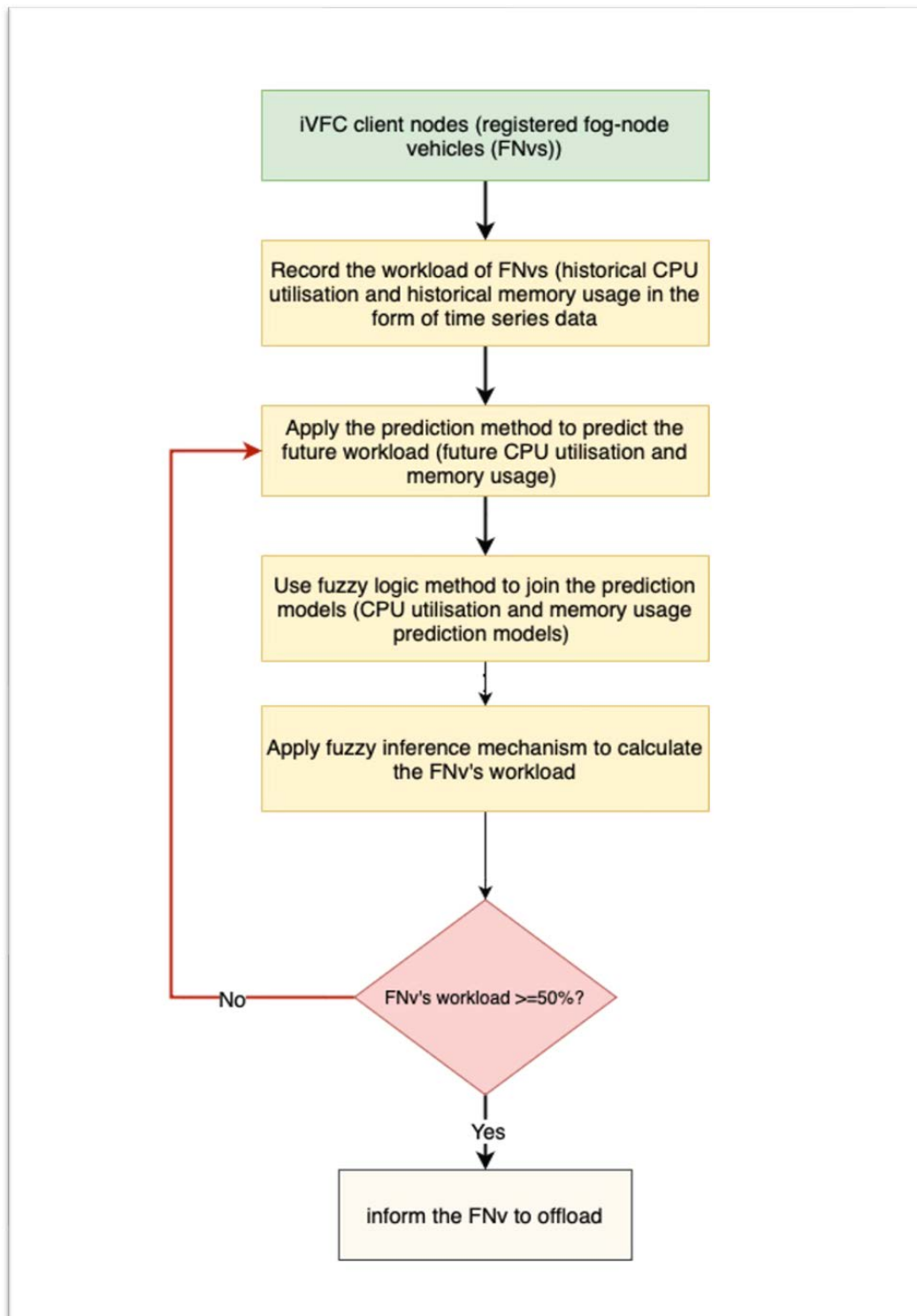


Figure 4.4 : Working steps of the proposed iVFC-predictive analytic module

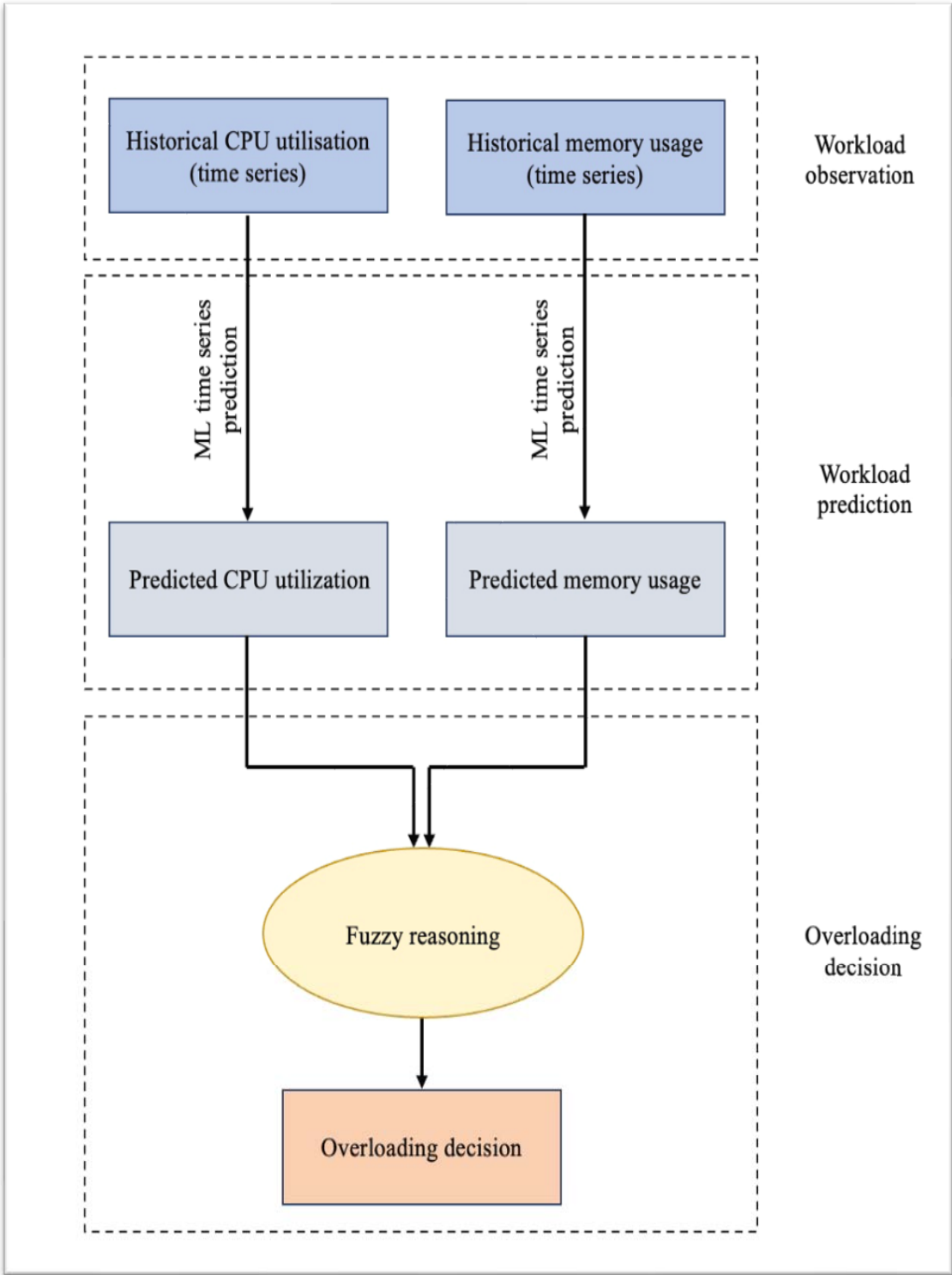


Figure 4.5 : Framework of the proposed iVFC-predictive analytic module

4.6 Overview of the solution for research objective 2 (RO2)

RO2 is to develop an incentive mechanism that considers a reward to motivate FNvs to take an incoming task and a penalty for declining to participate. The reason for using incentive mechanisms is to motivate FNvs to share their idle resources and improve their profitability and productivity as well as increase the participation level in the VFC environment. We chose game theory as an incentive mechanism to develop the incentive framework for the following reasons:

- As task offloading in VFC is considered a multiplayer decision problem in which the SNvs and the TNvs are competing to maximize their own benefits, this scenario can be viewed as a game [82].
- When the different node vehicles are competing in a task offloading game, some of them may act selfishly. This can be prevented using game theory which provides various efficient tools and mechanisms [82].
- Some nodes may not use system resources efficiently. Such cases can be avoided using the various mechanisms provided by game theory.

There are two types of game theory: cooperative game theory and non-cooperative game theory. Due to the selfishness and rationality of the vehicles that want to increase their outcomes in the VFC environment, the VFC environment is considered to be a non-cooperative game [82]. Therefore, in our framework, we consider vehicles as players in the game and we follow the solution concept of the Nash equilibrium (NE) game theory. Figure 4.6 outlines the working steps of the proposed incentive module. The development process of the proposed incentive module is given in detail in chapter 6 of this thesis.

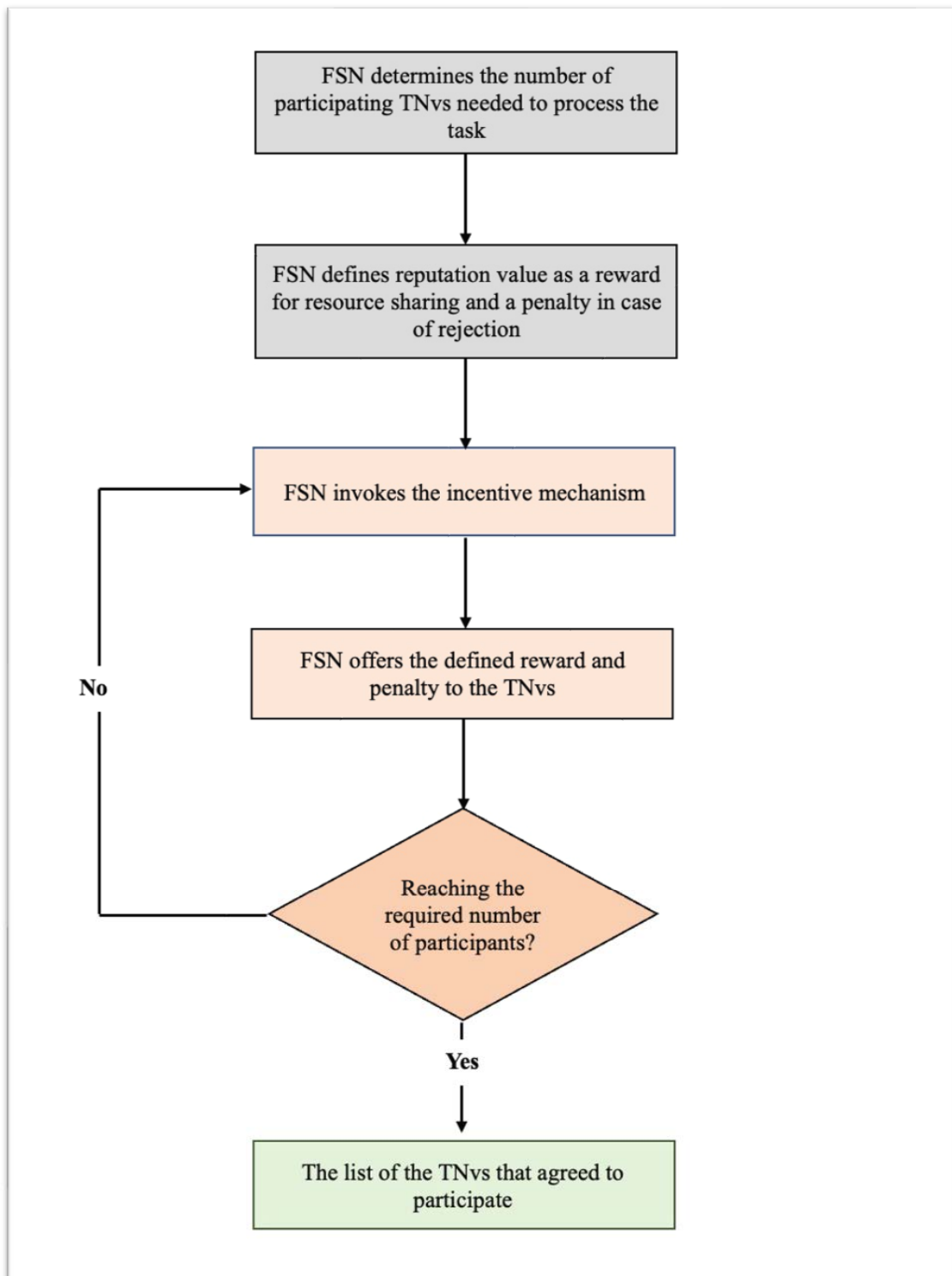


Figure 4.6 : Working steps of the proposed iVFC-incentive module

4.7 Overview of the solution for the research objective 3 (RO3)

RO3 is to develop an effective mechanism for service provider node selection (TNv selection) in the VFC environment. To achieve this objective, we develop a TNv selection module using three methods, classical method (multi-criteria decision-making (MCDM) method), machine learning (ML) method and deep learning (DL) method to compare the results of the three methods and choose the one that gives the best accuracy. These methods are used to rank the available TNvs that have agreed to participate in the task offloading process. TNvs ranking is based on their current workload (the percentage of their current CPU utilization and memory usage) from lowest to highest (the lower the percentage of the TNv current CPU utilization and memory usage, the higher the ranking of the node).

Figure 4.7 shows the working steps of the iVFC-TNv selection module and Figure 4.8 shows the proposed framework of the iVFC-TNv selection module. A detailed explanation of the iVFC-TNv selection module is given in chapter 7.

4.8 Evaluation and validation of the proposed iVFC solution (RO 4)

our proposed framework is validated by building the iVFC prototype system of the framework and the approaches developed for objectives (1-3). Subsequently, we use different simulation platforms to test and validate each of the three research objectives and we apply different evaluation metrics to evaluate the working of the developed approaches. The validation process for each of our research objectives is summarized in the following subsections.

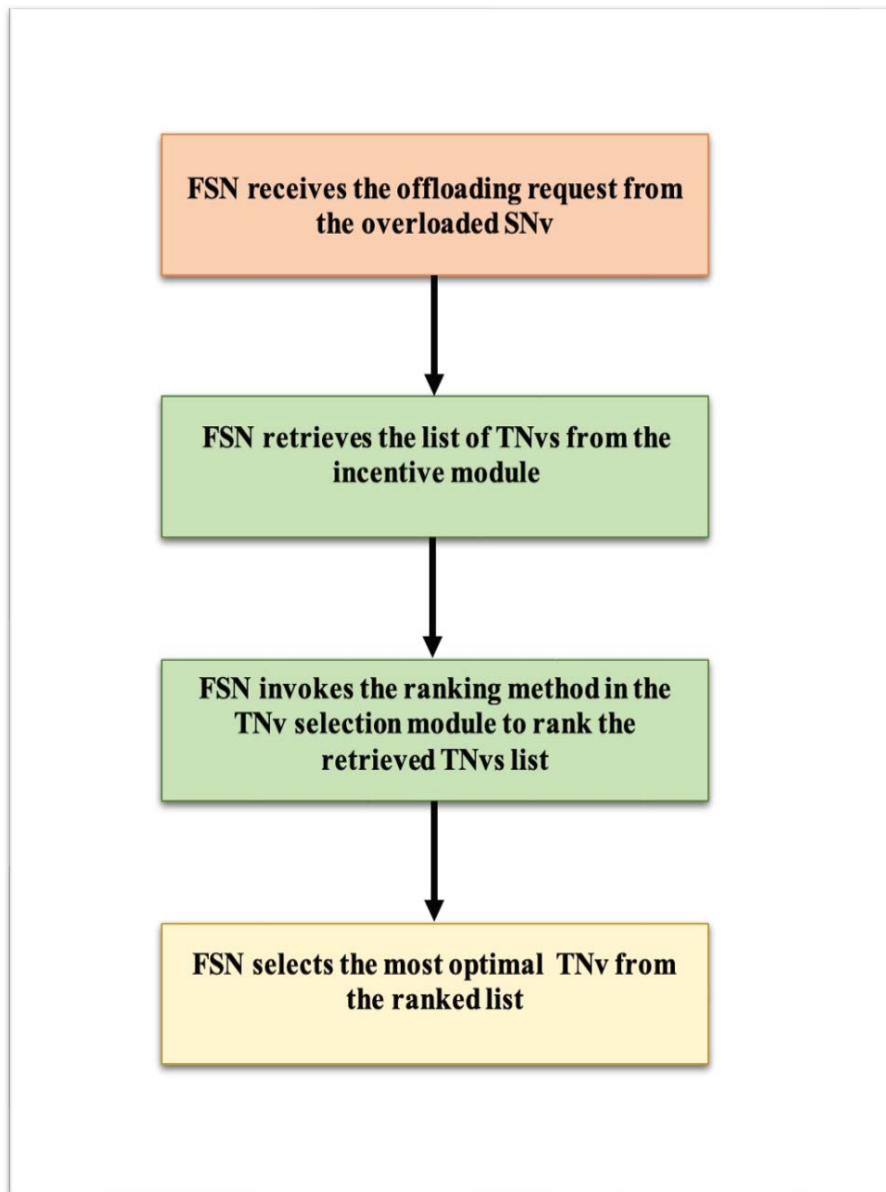


Figure 4.7 : Working steps of the proposed iVFC-TNv selection module

4.8.1 The validation steps for the solution to research objective 1 (RO1)

To validate the solution to RO1, we choose the Azure platform to fit the prediction models on our dataset, train and test the model, and choose the best model based on the evaluation metrics.

To evaluate and validation the solution to RO1 we implement the following steps:

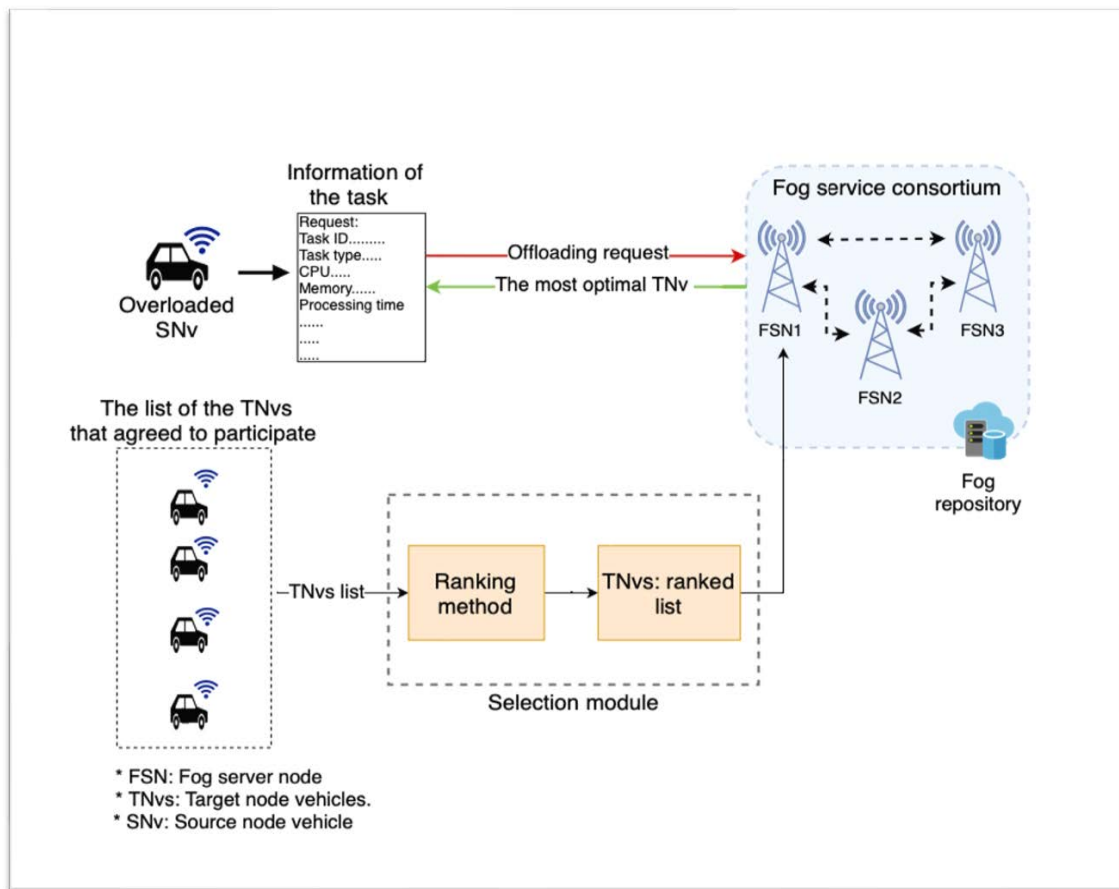


Figure 4.8 : The proposed framework of the iVFC-TNv selection module

Step 1: We build two prediction models on the Azure platform using ML time series prediction methods to predict CPU utilization and memory usage of the TNvs.

Step 2: We evaluate the two prediction models, CPU utilization prediction model and memory usage prediction model, based on the Normalized Root Mean Squared Error (NRMSE) metric. The NRMSE metric is a statistical metric that has become widely used to evaluate the accuracy of certain predictive models and is expressed as percentage [83]. Further details on the NRMSE metric are given in chapter 5.

Step 3: Using MATLAB (R2021b), we develop a fuzzy logic model to join the two prediction models, the CPU utilization prediction model and the memory usage prediction model that were generated in step one to make the overloading decision.

Step 4: We evaluate and validate the fuzzy logic model based on the following metrics: accuracy, precision and recall. These metrics are calculated based on true positive (TP), false positive (FP), true negative (TN) and false negative (FN) values obtained from the prediction data. These values are identified based on a comparison of the workload prediction (CPU utilization and memory usage) resulting from the fuzzy logic model with those resulting from the ML prediction models.

Chapter 5 includes further details on the evaluation and validation process for the RO1.

4.8.2 The validation steps for the solution to research objective 2 (RO2)

To validate the solution to RO2, we develop an incentive module using Stackelberg game theory. We implement Stackelberg game theory on Google Collaboratory.

The validation of the developed iVFC-incentive module includes the following steps:

Step1: Set up two parallel simulation scenarios as follows:

- **Scenario A:** Using the proposed incentive mechanism. This scenario uses the proposed incentive mechanism framework presented to address RO2 to encourage FNvs to participate in the task offloading process by sharing their idle resources.
- **Scenario B:** Without using the proposed incentive mechanism. This scenario runs without using the proposed incentive mechanism.

Step2: We run the simulation in both scenarios for a pre-defined interval of time. During this interval of time, there is a consortium node which always asks fog nodes running on the iVFC system to join and take tasks from the overloaded nodes.

Step3: We measure how many times the join request is accepted (i.e., compute the level of participation of FNvs in both scenarios) using the following formula:

$$\text{Level of Participation} = \frac{\text{Number of times the join request is accepted}}{\text{Total number of join requests}} \quad (4.1)$$

Chapter 6 includes further details on the evaluation and validation of the solution to RO2.

4.8.3 The validation steps for the solution to research objective 3 (RO3)

To evaluate and validate the solution to RO3, we develop a TNv selection module based on three methods: a statistical method, a machine learning method (ML), and a deep learning method (DL). Then, we evaluate the developed models using different metrics to choose the best working model for selecting the optimal TNvs.

A) Statistical method: multi-criteria decision-making (MCDM) method

MCDM methods have been widely used in cloud and fog computing systems to solve the problem of service provider (FSP) selection. There are many effective MCDM methods that have been used by researchers for FSP selection. We used two of these: the analytic hierarchy process (AHP) method to calculate the weights of the criteria of the TNvs and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method for ranking the different FNvs based on their criteria. [84].

To evaluate the working of the TNv selection module using the TOPSIS method, we implement TOPSIS using R (RStudio 2022.02.3+492). To measure the accuracy of the TOPSIS method in selecting the best TNv, we use the mean absolute error (MAE) metric, which is the absolute value of the difference between the forecasted value and the actual value. MAE measures the accuracy of the continuous variables and gives an indication of how large an error can be expected from the forecast on average [83]. We calculate MAE

using the following formula [83]:

$$MAE = \frac{\sum_{i=1}^n (|y_i - x_i|)}{n} \quad (4.2)$$

where y_i is the predicted value, x_i is the actual (observed) value and n is the total number of data points.

B) Machine learning method (ML)

We use the ML method as a second method to compare the ranking result with the TOPSIS method ranking. We choose a regression XGBoost (Extreme Gradient Boosting) method which is a popular and efficient open-source implementation of the gradient boosted trees algorithm to accurately predict the rank variable based on CPU utilization and memory usage variables of the TNvs [85].

We implement the XGBoost method on Google Collaboratory to predict the rank variable of the TNvs based on their CPU utilization and memory usage criteria. We conduct experiments by varying the number of estimators and learning rates, and we evaluate the performance of the XGBoost model using the MAE metric.

C) Deep learning (DL) methods

We implement DL methods as a third method to compare the result with the TOPSIS and XGBoost methods. We implement DNNs on Google Collaboratory to predict the rank variable of the TNvs based on their CPU utilization and memory usage criteria using different hidden layers and varying number of learning rates and we evaluate the performance of the DNNs model using the MAE metric. Chapter 7 provides further details on the evaluation and validation of the solution to RO3.

4.9 Conclusion

This chapter overviewed the solutions proposed to address research objectives 1-4 respectively and discussed the methodological approach used in this research, namely the design science research approach.

Also, this chapter included the different steps to build the proposed iVFC system and gave a brief overview of the different modules included in the iVFC system. These modules are the predictive analytic module (to address research objective 1), the incentive module (to address research objective 2), and the TNv selection module (to address research objective 3). It also gave an overview of the testing and validation of the proposed solution (to address research objective 4).

The next chapter discusses how the iVFC-predictive analytic module is developed to assist FNvs running on the iVFC system to make the overloading decision.

Chapter 5

A proactive-based task offloading in VFC using machine learning prediction techniques

5.1 Introduction

This chapter overviews the steps involved in developing a proactive-based framework for task offloading in vehicular fog computing (VFC), corresponding to research objective one. The purpose of this framework is to enable the proactive-based handling of task offloading for the VFC system in which the workload of each fog node vehicle (FNv) is predicted to monitor and control its overloading condition. Predicting the workload of FNvs to provide a proactive-based handling of the overloading decision is carried out by a predictive analytic module in our proposed iVFC framework. This module is responsible for monitoring the workload of the FNvs running on the VFC environment and this monitored data is used to continually predict the future overloading condition of the FNvs.

The design and development phases of the predictive analytic module and the experiment setup to evaluate and validate the proposed framework are explained in detail in this chapter. The proposed framework of the predictive analytic module comprises three phases, which are discussed in detail in this chapter.

This chapter is outlined as follows: section 5.2 explains in detail the different phases involved in developing the proposed framework of the predictive analytic module. Section 5.3 describes the different steps conducted to evaluate and validate the proposed predictive analytic module. Section 5.4 discusses the evaluation results and section 5.5 concludes this chapter and outlines the work to be covered in the next

chapter.

5.2 The proposed framework of the iVFC-predictive analytic module

The main purpose of the predictive analytic module is to develop a framework to accurately predict the various workload metrics (i.e., CPU utilization and memory usage) of the FNvs running in the VFC environment to make a prior overloading decision (i.e., proactive-based task offloading), which means that the future workload of FNvs can be predicted based on historical observation. We considered only CPU utilization and memory usage parameters to address their impact on the system's overall performance due to their significant role in determining the computational and memory capabilities of the system. These parameters were chosen because they are crucial factors that often contribute to overloading situations in many computing environments.

Due to the dynamic nature of FNvs in the VFC environment, the workload of the different FNvs varies with time (time series workload). Therefore, we propose a framework that can handle the dynamic nature of time series data efficiently. Based on the predicted workload, each FNv will be able to proactively predict when it is going to be overloaded in the future.

Figure 5.1 shows the three phases of our proposed iVFC-predictive analytic module framework, which are explained in more detail in the next subsections.

The framework of the iVFC-predictive analytic module comprises three phases. The first phase is the workload observation phase, which identifies the workload patterns of each FNv by monitoring its CPU utilization and memory usage. The second phase is the workload prediction phase, which includes predicting the future workload of FNvs using their historical data obtained from the workload observation

phase to generate two prediction models: one model to predict the CPU utilization and the other model to predict the memory usage of FNvs. The third phase is the overloading decision phase, which involves combining the two prediction models generated in the workload prediction phase to make an overloading decision.

5.2.1 Workload observation

Due to the dynamic nature of the VFC environment, the workload observation of different FNvs varies with time. Therefore, the data collected from different FNvs must be in the form of time series data. Once the vehicle has registered as a VFC client and starts participating as a fog node in the VFC environment, its CPU utilization and its memory usage data will be recorded and collected periodically by the nearest fog server node (FSN) and stored in the fog repository. The FSN in our proposed framework is a node located at the nearest road side unit to this FNv. This periodic data will be used by the predictive analytic module to predict the future workload of this FNv in the next time slot (i.e., short-term prediction).

5.2.2 Workload prediction

In this step, the periodic data (i.e., CPU utilization and memory usage) collected in the workload observation phase is used to predict the future workload of each FNv. In this step, two prediction models are created for each FNv using machine learning (ML) time series prediction methods; one model to predict the CPU utilization and the other model to predict the memory usage for each FNv based on its historical recorded workload. The workload prediction is generated on a 15-minute basis, which means the workload of the FNvs is continually predicted to decide their future overloading states every 15 minutes.

5.2.3 The overloading decision

In this phase, the output of the two models created in the workload prediction phase (the CPU utilization prediction model and the memory usage prediction model) are combined to make the overloading decision. Additional details on the development of the prediction models as well as the evaluation and validation are explained in the next section.

5.3 Evaluation of the proposed iVFC-predictive analytic module

5.3.1 Dataset

To build the prediction models for our proposed predictive analytic module, a publicly available workload trace, named rnd trace is used. Due to the unavailability of real vehicle workload datasets, we use a dataset of the observed workload of different virtual machines (VMs) hosted on the cloud which is collected from a typical data centre managed by a service provider called Bitbrains which specializes in hosting and business computation management for enterprises [86] [87]. This dataset comprises time series data involving eleven attributes, namely timestamp, CPU cores, CPU capacity provisioned in MHz, CPU usage in MHz, the percentage of CPU usage, memory capacity provisioned, memory used, hard disk reading speed, hard disk writing speed, network received throughput [KB/s] and network transmitted throughput [KB/s], in frequencies of 5 minutes for an interval of one month. We modified this dataset to suit our models as follows:

- Some of the attributes have been modified: the timestamp attribute has been converted from Epoch to datetime and an ID column has been added to be used as an identifier for each FNv.

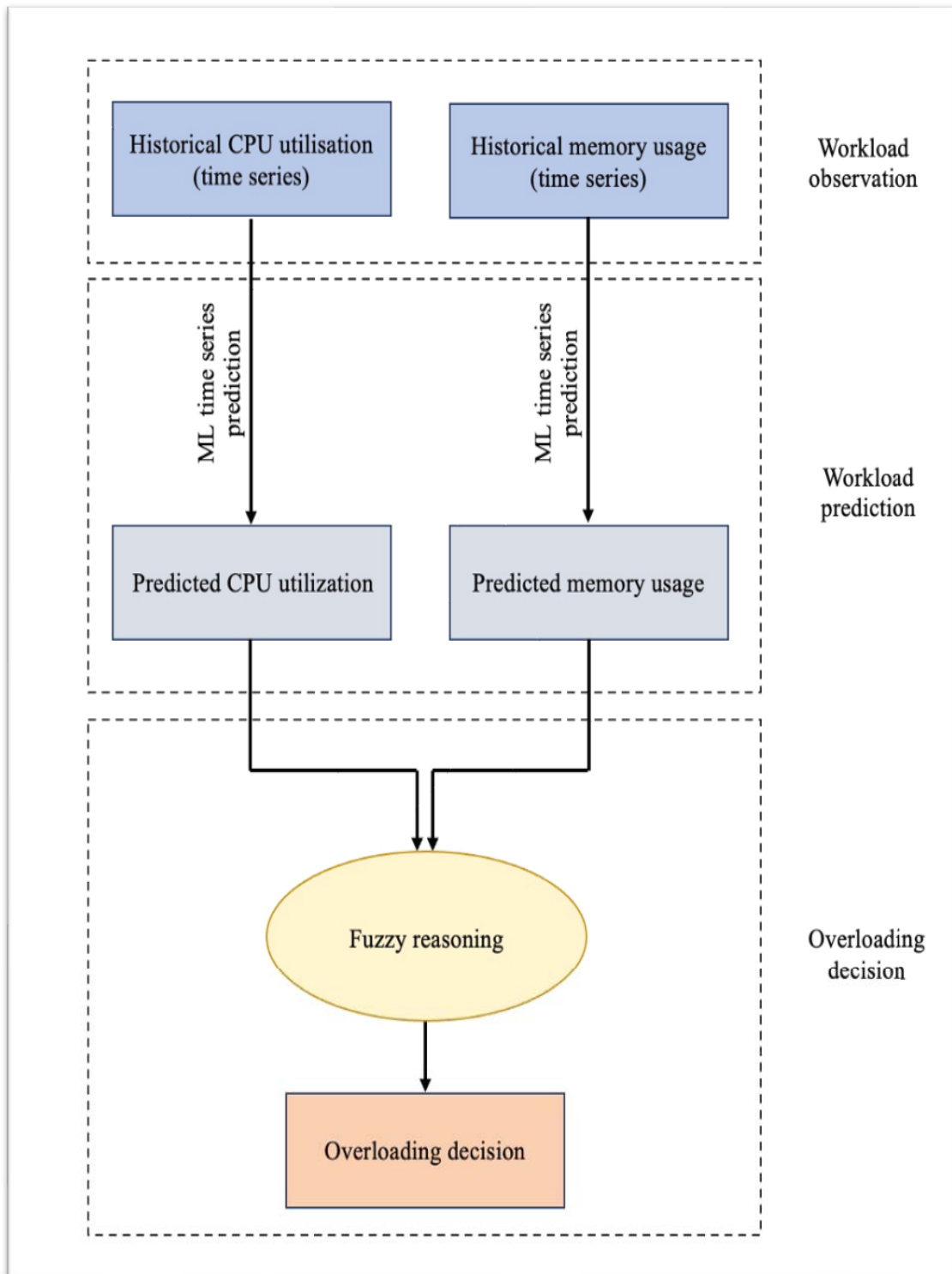


Figure 5.1 : The proposed framework of the iVFC-predictive analytic module

- Since our objective is to predict the workload of FNvs, we used only two columns from the dataset to train our prediction models, namely the percentage of CPU utilization and the memory used. These attributes represent the workload of the FNvs.
- We use a frequency of 15 minutes instead of 5 minutes as this is the lowest frequency that can be accepted by the Azure platform to build the prediction models.
- This dataset is split into two files, the CPU utilization file and the memory usage file.
- We use VMs as a representation of FNvs in our experiments so we rename the VMs column to FNvs.

Figure 5.2 shows a snapshot of the dataset that we used to develop our proposed prediction models. The modified dataset is available at the following GitHub link: <https://github.com/alhamedy/Dataset-used-to-build-the-time-series-prediction-models>.

5.3.2 The experimental setup and implementation

The platform used to build and train the prediction models

We chose Microsoft Azure as the platform to build the time series prediction models to predict the CPU utilization and memory usage for each FNv. Azure is a web-based console that replaces command-line tools with a very flexible graphical user interface [88]. We chose Azure platform due to its flexibility in accepting the time series intervals of our dataset and its ability to apply a large number of machine learning time series prediction algorithms when building the prediction models.

100 FNvs (CPU utilization)			100 FNvs (Memory usage)		
Node-ID	Timestamp	CPU usage [%]	Node-ID	Timestamp	Memory usage [%]
FNv-1	2013-08-12 13:30:00	93.23333333	FNv-1	2013-08-12 13:30:00	9.133330584
FNv-1	2013-08-12 13:45:00	90.75	FNv-1	2013-08-12 13:45:00	17.13333024
FNv-1	2013-08-12 14:00:00	93.18888889	FNv-1	2013-08-12 14:00:00	10.82221932
FNv-1	2013-08-12 14:15:00	90.46111111	FNv-1	2013-08-12 14:15:00	20.2666631
FNv-1	2013-08-12 14:30:00	50.13333333	FNv-1	2013-08-12 14:30:00	13.7555537
FNv-1	2013-08-12 14:45:00	0.61111111	FNv-1	2013-08-12 14:45:00	0.733332634
FNv-1	2013-08-12 15:00:00	0.605555556	FNv-1	2013-08-12 15:00:00	0
FNv-1	2013-08-12 15:15:00	0.583333333	FNv-1	2013-08-12 15:15:00	0
FNv-1	2013-08-12 15:30:00	0.594444444	FNv-1	2013-08-12 15:30:00	0
FNv-1	2013-08-12 15:45:00	0.538888889	FNv-1	2013-08-12 15:45:00	0
FNv-1	2013-08-12 16:00:00	0.527777778	FNv-1	2013-08-12 16:00:00	0
FNv-1	2013-08-12 16:15:00	0.538888889	FNv-1	2013-08-12 16:15:00	0
FNv-1	2013-08-12 16:30:00	0.572222222	FNv-1	2013-08-12 16:30:00	0
FNv-1	2013-08-12 16:45:00	0.55	FNv-1	2013-08-12 16:45:00	0
FNv-1	2013-08-12 17:00:00	0.55	FNv-1	2013-08-12 17:00:00	0
FNv-1	2013-08-12 17:15:00	0.583333333	FNv-1	2013-08-12 17:15:00	0
FNv-1	2013-08-12 17:30:00	0.572222222	FNv-1	2013-08-12 17:30:00	0
FNv-1	2013-08-12 17:45:00	0.622222222	FNv-1	2013-08-12 17:45:00	0
FNv-1	2013-08-12 18:00:00	0.594444444	FNv-1	2013-08-12 18:00:00	0.155555407
FNv-1	2013-08-12 18:15:00	0.577777778	FNv-1	2013-08-12 18:15:00	0
FNv-1	2013-08-12 18:30:00	0.583333333	FNv-1	2013-08-12 18:30:00	0
FNv-1	2013-08-12 18:45:00	0.572222222	FNv-1	2013-08-12 18:45:00	0
FNv-1	2013-08-12 19:00:00	0.566666667	FNv-1	2013-08-12 19:00:00	0
FNv-1	2013-08-12 19:15:00	0.744444444	FNv-1	2013-08-12 19:15:00	0
FNv-1	2013-08-12 19:30:00	0.594444444	FNv-1	2013-08-12 19:30:00	0

Figure 5.2 : Snapshot of the dataset used to train the prediction models

Build and train the prediction models

The two files of the dataset (the CPU utilization data file and the memory usage data file) were exported to the Azure portal on the Azure platform. Then, the two files were used to build two prediction models, the CPU utilization prediction model and the memory usage prediction model. To build the prediction models, auto ML experiments were conducted twice, once to predict the CPU utilization of all FNVs using the CPU utilization data file and the second to predict the memory usage of all FNVs using the memory usage data file.

When building the CPU utilization model, the CPU utilization column was chosen as the target column and when building the memory usage model, the memory usage column was chosen as the target column to be predicted. During each experiment, we choose ML time series prediction as the prediction method to be applied by Azure to build the prediction models because our dataset varies with time (time series data). We conducted these experiments three times when building each model to obtain an accurate prediction using a different number of FNV datasets.

Table 5.1 shows the various Azure Auto ML experiments that were implemented and Table 5.2 details the parameters used in the experiment evaluation.

As clearly shown in Table 5.1, when we use a dataset of a large number of FNVs (363 FNVs), the experiment takes longer to execute, at around 6 hours. After 6 hours of training the model with a dataset of 363 FNVs, Azure stops training the model on the dataset and only 6 ML time series prediction algorithms were applied in both the CPU utilization prediction experiment and the memory usage prediction experiment.

Therefore, from the three experiments, we chose to use experiment 2 for the next step of the workload prediction process as it takes less time to complete the training and the testing of the models, more ML time series prediction algorithms

were applied when training the model in experiment 2 compared to experiment 3 and the comparison metric of the best model chosen by Azure (NRMSE, as explained in the next section) in experiment 2 is better than experiment 1 and experiment 3.

Table 5.1 : ML time series prediction experiments to build and train the prediction models

Experiment	# of FNvs	Duration	# of ML algorithms	CPU utilization model		Memory usage model	
				Best Model	NRMSE	Best Model	NRMSE
Experiment 1	15	48.37 mins	41	Naive	0.05675	Naive	0.04583
Experiment 2	100	43.67 mins	41	Exponential Smoothing	0.01738	Exponential Smoothing	0.06178
Experiment 3	363	6 hrs	6	Exponential Smoothing	0.274	Exponential Smoothing	0.432

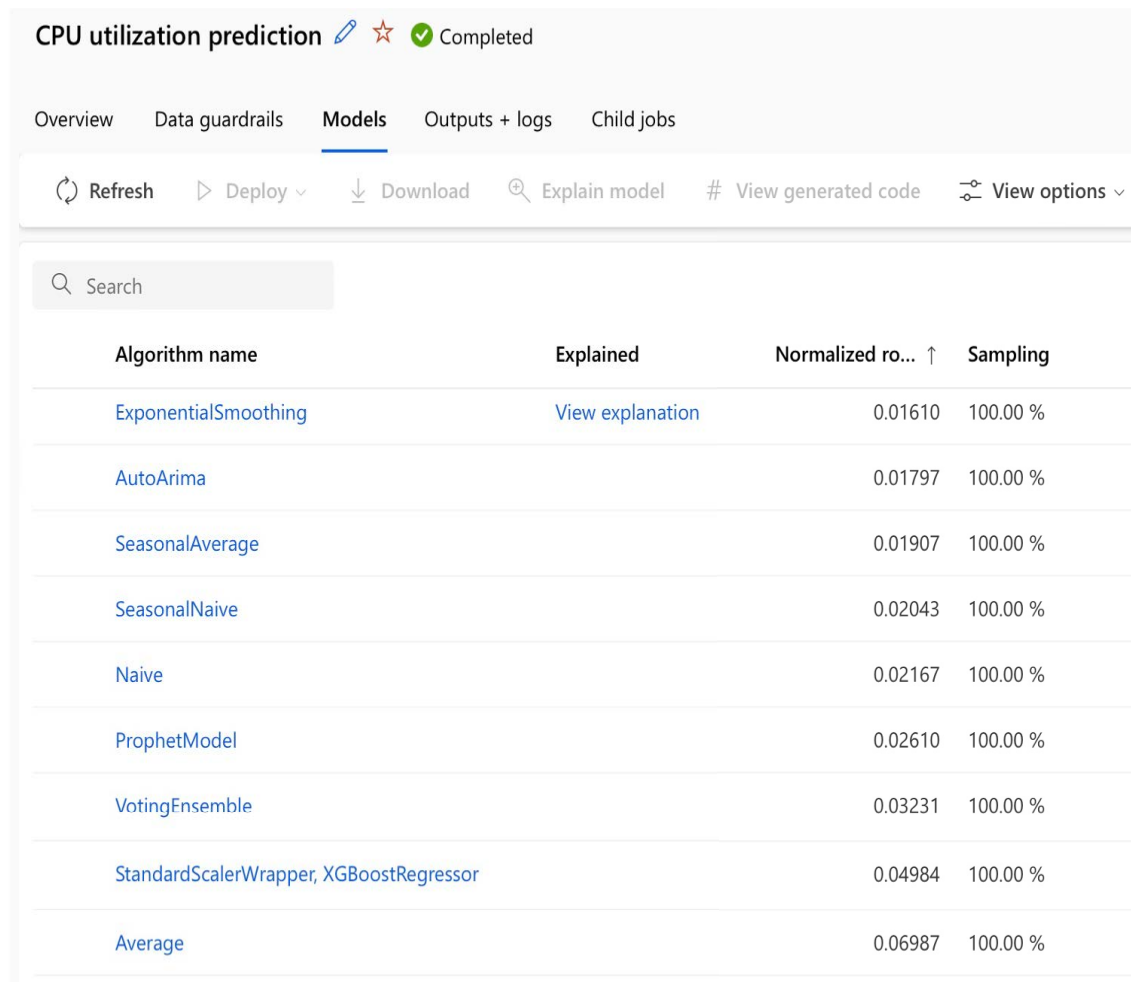
Table 5.2 : The parameters used in the time series prediction experiments

Parameter	Definition
# of FNvs	The number of fog node vehicles in the dataset used in each experiment.
# of ML algorithms	The total number of machine learning time series prediction methods applied by Azure during model training.
Best Model	The best machine learning time series prediction method chosen by Azure after training the model.
NRMSE	The normalized root mean square error metric which is used to evaluate the resulting prediction models and select the best performing method in building the prediction models.

Choosing the best model for workload prediction

During the running of the auto ML experiments, 41 ML time series prediction methods were applied by Azure to build the workload prediction models on experiments 1 and 2, as shown in Table 5.1. The best model in each experiment is chosen automatically by Azure according to the NRMSE metric which ranks all models from lowest to highest (i.e., the lower the NRMSE, the better the model). Further details

about the NRMSE metric are given in section 5.3.3. Figures 5.3 and 5.4 show the different ML methods applied by Azure in experiment 2.



CPU utilization prediction [✎](#) [★](#) [✔](#) Completed

Overview Data guardrails **Models** Outputs + logs Child jobs

[↻ Refresh](#) [▶ Deploy](#) [↓ Download](#) [🔍 Explain model](#) [# View generated code](#) [⚙️ View options](#)

🔍 Search

Algorithm name	Explained	Normalized ro... ↑	Sampling
ExponentialSmoothing	View explanation	0.01610	100.00 %
AutoArima		0.01797	100.00 %
SeasonalAverage		0.01907	100.00 %
SeasonalNaive		0.02043	100.00 %
Naive		0.02167	100.00 %
ProphetModel		0.02610	100.00 %
VotingEnsemble		0.03231	100.00 %
StandardScalerWrapper, XGBoostRegressor		0.04984	100.00 %
Average		0.06987	100.00 %

Figure 5.3 : CPU utilization prediction experiment on Azure portal

Using the chosen model for prediction

After choosing the best CPU utilization prediction model, we downloaded the model file to my personal device (MacBook Pro 13, version: macOS 13.4) to retrieve the prediction output for each FNv. The model file is then uploaded to the Jupiter notebook on the Azure portal to obtain the prediction output. To retrieve the prediction output, we chose one hour (12:00 – 01:00) for the specific date (01/09/2013) to ob-

Algorithm name	Explained	Normalized ro... ↑	Sampling
ExponentialSmoothing	View explanation	0.06178	100.00 %
VotingEnsemble		0.06289	100.00 %
AutoArima		0.06412	100.00 %
StandardScalerWrapper, XGBoostRegressor		0.06428	100.00 %
ProphetModel		0.06815	100.00 %
StandardScalerWrapper, XGBoostRegressor		0.06842	100.00 %
SeasonalAverage		0.07217	100.00 %
StandardScalerWrapper, XGBoostRegressor		0.07833	100.00 %
StandardScalerWrapper, LightGBM		0.08138	100.00 %

Figure 5.4 : Memory usage prediction experiment on Azure portal

tain the prediction output each 15 minutes for CPU utilization then for memory usage.

The following link is the Python code used on the Jupiter notebook on the Azure portal to obtain the prediction output from both models, the CPU utilization prediction model and the memory usage prediction model: <https://github.com/alhamedy/Retrieve-prediction-values>.

This step resulted to two files containing CPU utilization and memory usage prediction output for the next time slot (the next 15 minutes). Table 5.3 shows a sample of the prediction values for CPU utilization and memory usage after re-

trieving the values on the Jupiter notebook. The prediction values (CPU utilization and memory usage) of all FNVs that were retrieved from the chosen prediction models using the Jupiter notebook can be found at the following GitHub link:<https://github.com/alhamedy/Predicted-values-dataset>.

Table 5.3 : Prediction values for an interval of 15 minutes for CPU utilization and memory usage of the 100 FNVs

Node-ID	Timestamp	CPU utilization [%]	Memory usage [%]
FNv1	1/9/2013 00:00:00	31.16	0.07
FNv1	1/9/2013 00:00:15	31.16	0.06
FNv1	1/9/2013 00:00:30	31.16	0.01
FNv1	1/9/2013 00:00:45	31.16	0.04
FNv1	1/9/2013 01:00:00	31.16	0.07
FNv2	1/9/2013 00:00:00	30.56	19.46
FNv2	1/9/2013 00:00:15	30.56	19.48
FNv2	1/9/2013 00:00:30	30.56	19.93
FNv2	1/9/2013 00:00:45	30.56	20.14
....
....
....
....
....
....
FNv100	1/9/2013 00:00:15	1.27	0.49
FNv100	1/9/2013 00:00:30	1.27	0.52
FNv100	1/9/2013 00:00:45	1.3	0.68
FNv100	1/9/2013 01:00:00	1.28	0.60

Combining the two prediction models to predict the workload of FNvs and make the overloading decision

To use the prediction models to predict the workload of a FNv and make the overloading decision, the output of the CPU utilization prediction model and the memory usage prediction model have to be combined. To do so, a fuzzy logic method is used. Building a fuzzy logic model based on the two time series prediction models will assist in the accurate prediction of the workload of FNvs. The purpose of using a fuzzy logic model is to combine the prediction output resulting from training the CPU utilization and the memory usage prediction models to build one accurate prediction model that can be used to make the overloading decision for the FNvs.

Fuzzy logic is a computation approach on which the modern computer is based. Rather than the usual “true or false”, this approach uses “degrees of truth” of the inputs and produces outputs based on the states of the inputs and the values of change of these states [89].

The fuzzy logic approach was first introduced by Zedah in the 1960s and this method has been used in artificial intelligent (AI) systems to imitate human decision making [89]. This method considers all the available data to solve a problem and then makes the best decision for the given input.

As shown in Figure 5.5, we use the Mamdani method to design and implement our fuzzy inference systems. Mamandi is a well-known method introduced by Ebrahim Mamdani in 1975 and has become one of the most popular approaches in fuzzy logic control systems [90].

In the Mamdani method, we choose methods for the ”and” operator, ”or” operator, implication, aggregation, and defuzzification as following:

- ”And” operator: we used the minimum ’min’ operator to capture the least

dominant input, where the minimum value among the fuzzy sets or membership values is selected.

- "Or" operator: we used the maximum operator 'max' to capture the most dominant input where the maximum value among the fuzzy sets or membership values is selected.
- Implication: we used the minimum operator 'min' to capture the minimum of the firing strengths of the antecedent and consequent parts of the rules.
- Aggregation: we used the maximum operator 'max' to select the maximum value among the fuzzy outputs.
- Defuzzification: we used the 'centroid' method to compute the centre of gravity of the fuzzy output.

We employed fuzzy rules to model the sources from both prediction models. The rule base consists of linguistic variables and membership functions which determine how the controller will perform actions based on the if-then rules. To produce a fuzzy set output, an inference mechanism has to be applied to the set of if-then rules. In this process, the input fuzzy set is matched with the premise of the rules, the rules are activated to determine which conclusion is deduced by each rule and the activated conclusions are combined using the fuzzy set union to generate the fuzzy set output. We used the CPU utilization and memory usage prediction values as the input fuzzy set, as discussed below and shown in Figure 5.5.

- Fuzzify Input and Output Parameters: a fuzzifier is used to map the data values to the fuzzy sets and then a degree of membership is assigned for each fuzzy set. CPU utilization and memory usage are the input parameters and

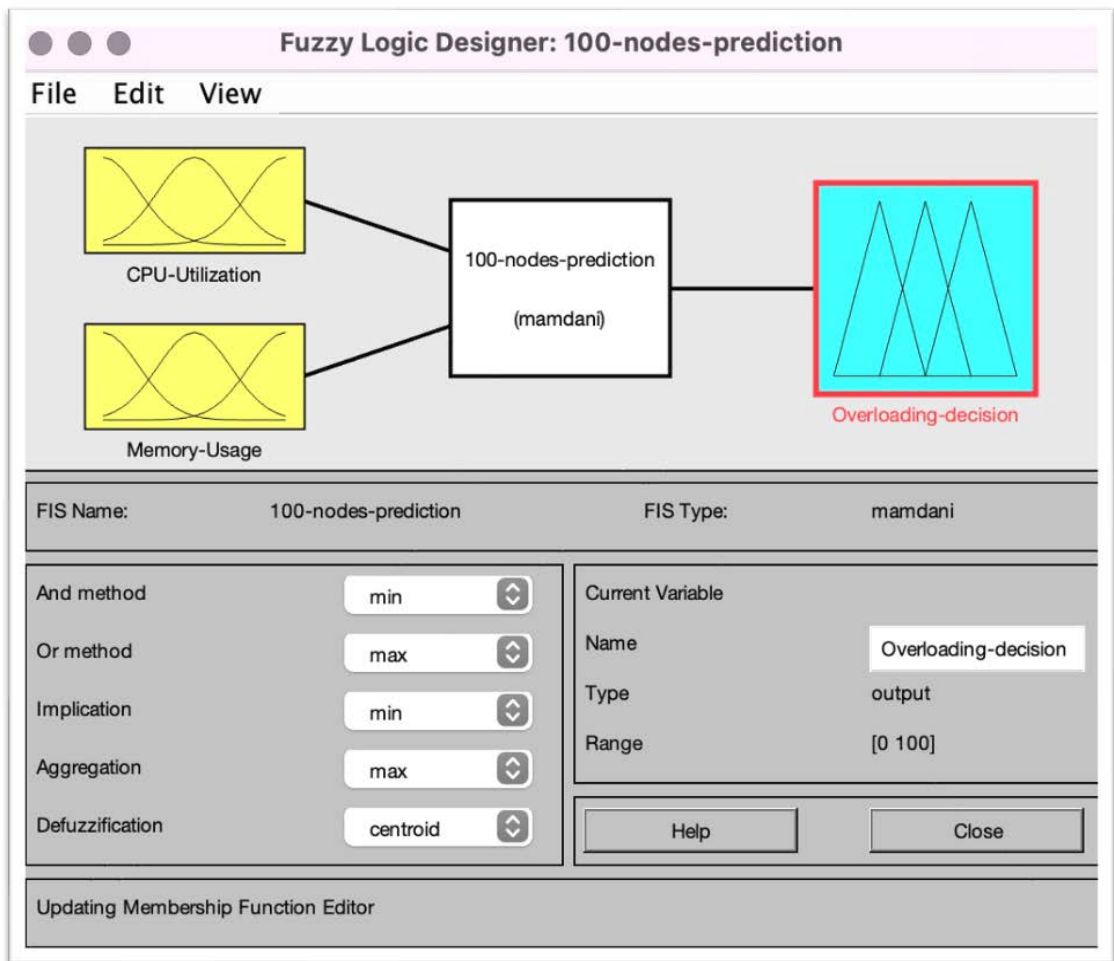


Figure 5.5 : Fuzzy logic model for predicting the workload of FNvs

the overloading decision is the output, each of which is fuzzified to four fuzzy sets.

- Fuzzify Node Workload Input: The fuzzy sets for the CPU utilization and memory usage input variables have the following names: Low, Medium, High and Very High. Hence, CPU utilization is fuzzified between Low= 0-15 %, Medium= 15-30%, High= 30-60%, and Very High $\geq 60\%$. Memory usage is fuzzified between Low= 0-15 %, Medium= 15-30%, High =30-60%, and Very High $\geq 60\%$, as shown in Figures 5.6 and 5.7.
- Fuzzify Offloading Decision Output: Fuzzy sets for the overloading output

variables have the following names: Overloaded, Prepare-to-overload, and Not-overloaded. The offloading decision is fuzzified between Not-overloading= 0-15 %, Prepare-to-overload= 15-49.9 %, and Overloaded= 50-95%, as shown in Figure 5.8.

- Fuzzy Rules and Fuzzy Inference: we propose eight fuzzy rules to be used by the fuzzy inference to map the fuzzy input sets: Low, Medium, High, Very High (V-high), and the fuzzy output sets: Not-overloaded, Prepare-to-overload and Overloaded, as shown in Figure 5.9
 - Rule 1: IF (CPU utilization is low) and (memory usage is low) THEN the overloading decision is: not-overloaded.
 - Rule 2: IF (CPU utilization is low) and (memory usage is medium) THEN the offloading decision is: not-overloaded.
 - Rule 3: IF (CPU utilization is medium) and (memory usage is low) THEN the offloading decision is: not-overloaded.
 - Rule 4: IF (CPU utilization is medium) and (memory usage is medium) THEN the offloading decision is: not-overloaded.
 - Rule 5: IF (CPU utilization is high) and (memory usage is high) THEN the offloading decision is: prepare-to-overload.
 - Rule 6: IF (CPU utilization is high) or (memory usage is high) THEN the offloading decision is: prepare-to-overload.
 - Rule 7: IF (CPU utilization is V-high) or (memory usage is V-high) THEN the offloading-decision is: overloaded.
 - Rule 8: IF (CPU utilization is V-high) and (memory usage is V-high) THEN the offloading decision is: overloaded.

The fuzzy inference process evaluates all eight fuzzy rules (RULE 1 to RULE 8) and

finds their antecedent part firing strength and then applies this firing strength to the consequent part of the rules. Finally, the weighted average defuzzification method is applied to obtain a single offloading decision output.

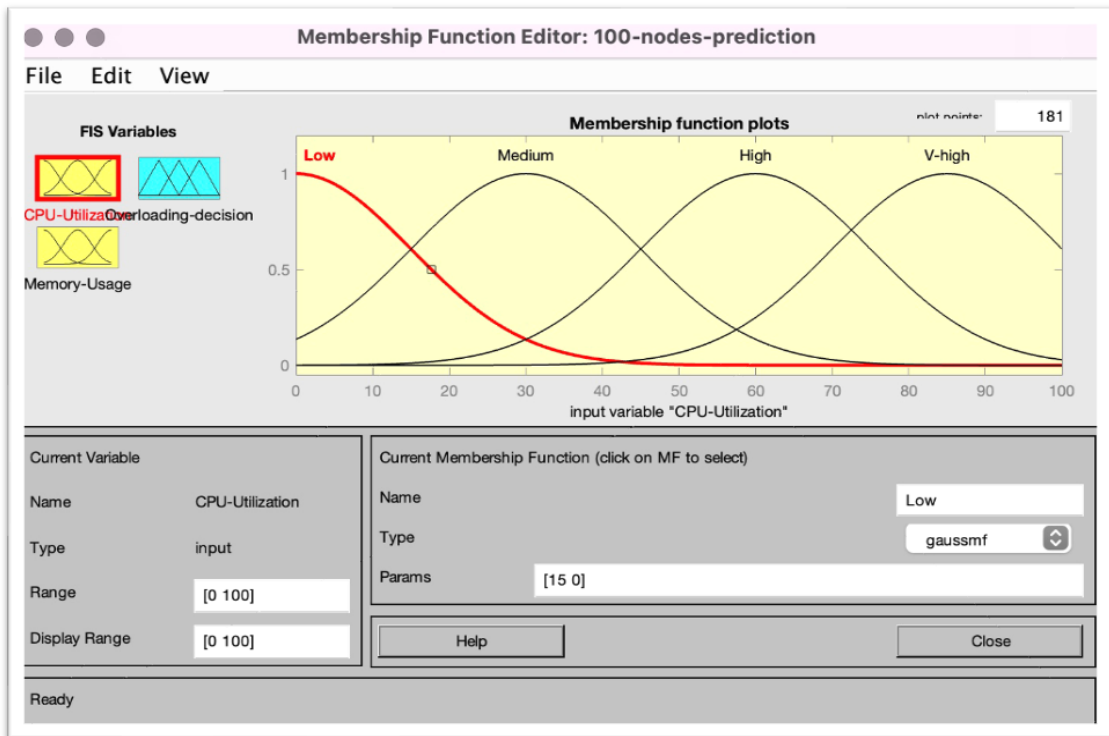


Figure 5.6 : The fuzzy sets for the CPU utilization input variables

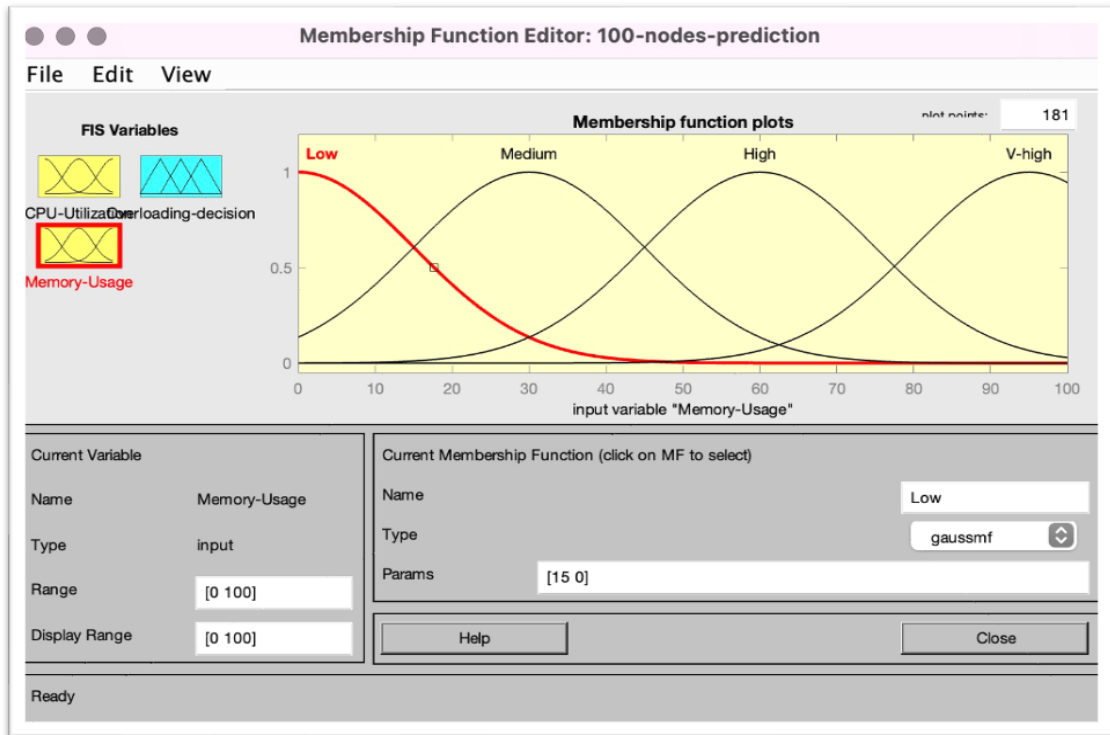


Figure 5.7 : The fuzzy sets for the memory usage input variables

After building a fuzzy logic prediction model, the fuzzy logic model file must be saved in the same folder along with the prediction output files retrieved from the prediction models (the CPU utilization and the memory usage prediction models). Then, we used Version (R2021b) of MATLAB to obtain the prediction output using the developed fuzzy logic model. The MATLAB code used to obtain the prediction output from the developed fuzzy logic model can be accessed via the following GitHub link: <https://github.com/alhamedy/MATLAB-code-for-fuzzy-output>.

After obtaining the prediction output from the developed fuzzy logic model, we compare the prediction output obtained from the time series prediction models (the CPU utilization and the memory usage prediction models) detailed in section 5.3.2 with the fuzzy logic prediction output, as shown in Figure 5.14, to obtain the confusion matrix table. The confusion matrix is a square matrix in which the

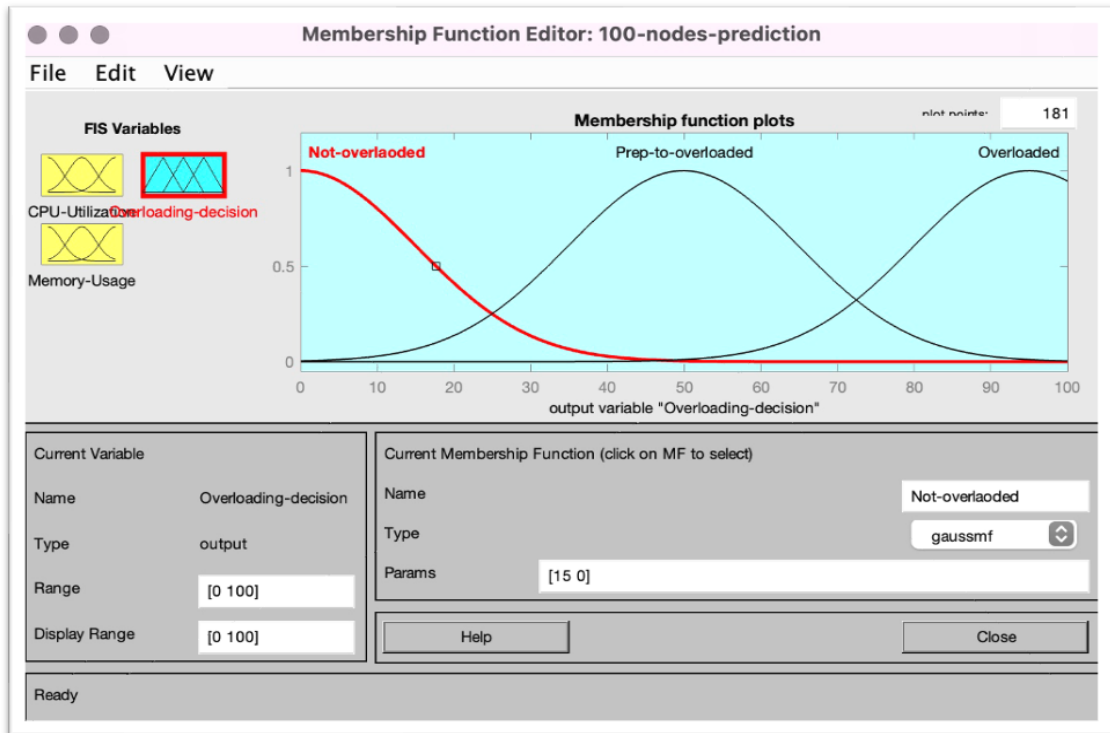


Figure 5.8 : The fuzzy sets for the overloading decision input variables

rows represent the actual values and the columns represent the predicted values [91]. The confusion matrix reports the number of true positive (TP), false positive (FP), true negative (TN) and false negative (FN) parameters. The goal is to evaluate the performance of the fuzzy logic model in predicting the overloading status of FNVs based on their workload (i.e., their CPU utilization and their memory usage).

Based on the prediction output obtained from the time series prediction models for the CPU utilization and for the memory usage, we manually recorded the overloading status of each FNV based on its predicted CPU utilization and memory usage as follows:

- If the CPU utilization is low (from 0 to $< 60\%$) AND the memory usage is low (from 0 to $< 60\%$), then the decision is that the node is not overloaded.
- If the CPU utilization is high ($\geq 60\%$) AND the memory usage is high (\geq

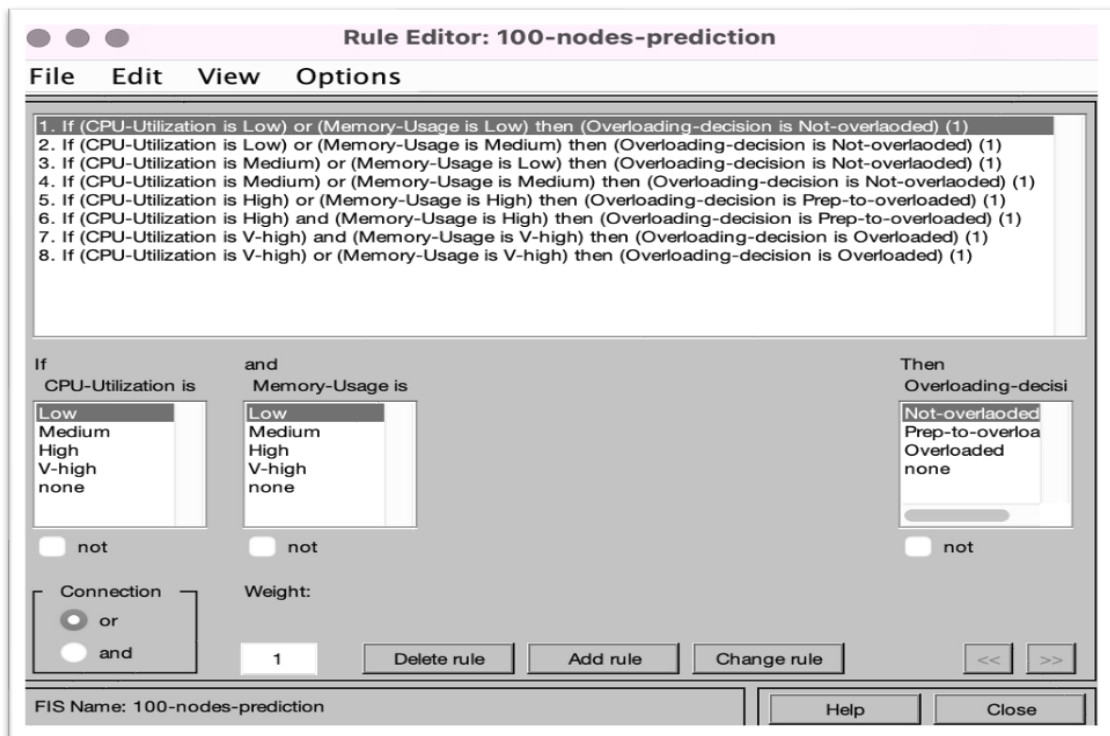


Figure 5.9 : The fuzzy logic model rules to obtain the offloading decision outputs

60%), the decision is that the node is overloaded.

- If the CPU utilization is high ($\geq 60\%$) OR the memory usage is high ($\geq 60\%$), the decision is that the node is overloaded.

Which are the same values assumed to fuzzify the node workload input when building the fuzzy logic model. Then we compare the manually recorded predictions with predictions obtained from the fuzzy logic model to calculate the values of the true positive (TP), false positive (FP), true negative (TN) and false negative (FN) as follows:

- A TP is the number of outcomes where the model correctly predicts the CPU utilization and the memory usage of the FNv (i.e., correctly predicts that the node is overloaded).

- A TN is the number of outcomes where the model correctly predicts the negative CPU utilization and memory usage of the FNv (i.e., correctly predicts that the node is not overloaded).
- A FP is the number of outcomes where the model incorrectly predicts the CPU utilization and the memory usage of the FNv (i.e., incorrectly predicting the nodes as being overloaded).
- A FN is the number of outcomes where the model incorrectly predicts the negative CPU utilization and memory usage of the FNv (i.e., incorrectly predicting the nodes as not overloaded).

Then we use the recorded TP, TN, FP and FN values to evaluate the performance of the developed fuzzy logic model in predicting the overloading status of FNvs based on their workload as shown in the next section.

5.3.3 Evaluation metrics

To evaluate the two prediction models generated from the proposed predictive analytic module, we use the following metrics:

Normalized root mean square error (NRMSE) metric

NRMSE is calculated by taking the square root of the average squared difference between the predicted and the actual values, dividing it by the range of the actual values, and multiplying the result by 100 [83].

NRMSE is calculated using the following formula:

$$NRMSE = \frac{\sum (S_i - O_i)^2}{\sum O_i^2} \quad (5.1)$$

where O_i are the observed (i.e., actual) values and S_i are the predicted values. The closer the NRMSE value is to zero, the more accurate the model.

Predicted vs true

The predicted vs. true chart illustrates the relation between the target feature (actual values) and the predictions made by the model [88]. On the x-axis, the true values are categorized into bins. A plot with error bars shows the average predicted value for each bin to detect any bias in the model's predictions. Shaded areas represent the variance of predictions around the average, and the line represents the overall average prediction. The model tends to make accurate predictions for the true value that commonly occurs in the dataset. Additionally, the lower variance (or spread) of these predictions is indicating a higher level of consistency. In regions with fewer true values, the distance between the trend line and the ideal $y = x$ line is a good indicator of the model performance on outliers [88]. An outlier is a point of data that deviates significantly from the overall pattern or distribution of the dataset.

Residuals chart

The residuals chart is a histogram of the prediction errors (residuals) generated for the prediction experiments. These residuals are obtained by subtracting the predicted values from the true values for each sample [88]. To validate the fuzzy logic model, we used the following metrics:

1. Accuracy is calculated using the following formula:

$$Accuracy = \frac{TN + TP}{total\ number\ of\ prediction\ values} \quad (5.2)$$

2. Precision is the proportion of instances that correctly predict the node as an overloaded node out of all the nodes that are predicted as overloaded nodes.

Precision is calculated using the following formula:

$$Precision = \frac{TP}{(TP + FP)} \quad (5.3)$$

3. Recall is the proportion of instances that correctly predict the node as not an overloaded node out of all the nodes that are predicted as not overloaded nodes. Recall is calculated using the following formula:

$$Recall = \frac{TP}{(TP + FN)} \quad (5.4)$$

These metrics are calculated based on the TP, FP, TN and FN values obtained from the prediction data. These values are obtained based on the comparison that was reported in section 5.3.2, as shown in Figure 5.14.

5.4 Results and discussion

Table 5.4 shows the top 10 ML time series prediction methods applied by Azure to build the prediction models for the CPU utilization and the memory usage that we reported in section 5.3.2 in relation to experiment 2, ranked according to the NRMSE metric where the best prediction model for CPU utilization and memory usage is chosen based on the NRMSE metrics in which the closer the NRMSE value is to zero, the more accurate the model.

Table 5.4 shows that the exponential smoothing algorithm has the lowest NRMSE for both models, hence we choose it for the prediction for both experiments (CPU utilization and memory usage prediction). Figures 5.10, 5.11, 5.12 and 5.13 show the metric charts of the best model (exponential smoothing) for both the CPU utilization and the memory usage models.

Based on the residuals histogram shown in Figures 5.11 and 5.13, the CPU utilization prediction model and the memory usage prediction model are considered good models as both have a residuals distribution that peaks at zero with few residuals at the extremes.

By analysing the trend line's distance from the ideal line in the outlier regions of the predicted vs. true chart presented in Figures 5.10 and 5.12, we can see that

Table 5.4 : Comparison of the top-10 ML algorithms applied on experiment 2

CPU utilization prediction experiment		Memory usage prediction experiment	
Algorithm name	NRMSE	Algorithm name	NRMSE
ExponentialSmoothing	0.01738	ExponentialSmoothing	0.06178
AutoArima	0.01917	VotingEnsemble	0.06323
SeasonalAverage	0.02053	AutoArima	0.06412
SeasonalNaive	0.02245	StandardScalerWrapper, XGBoostRegressor	0.06463
Naive	0.02286	ProphetModel	0.06820
ProphetModel	0.02720	SeasonalAverage	0.07217
VotingEnsemble	0.04090	StandardScalerWrapper, LightGBM	0.08112
StandardScalerWrapper, XGBoostRegressor	0.05958	SeasonalNaive	0.08440
Average	0.07056	Naive	0.08730
StandardScalerWrapper, LightGBM	0.013184	Average	0.09234

the trend line almost aligns perfectly with the diagonal line $y = x$, which means the model's predictions almost match the exact true values.

Table 5.5 and Figure 5.14 show the accuracy, precision and recall calculations of the fuzzy logic model that was detailed in section 5.3.2. As can be seen, the fuzzy logic model achieves excellent accuracy (94.4%) which indicates that the overall performance of the model is good and the model can be used to predict the future workload of vehicular fog nodes.

The precision of the model is 93%, which means that of all the predictions where the model classifies a node as overloaded, 93% are actually true. This is a good result, as it indicates that the model is able to correctly identify an overloaded node with a high degree of confidence.

The recall of the model is 62%, which means that the model only correctly predicted 62% of nodes as not being overloaded. This low score may be due to difficulty in correctly predicting the workload of FNvs that belong to the not overloaded FNv category. This could be due to a range of factors, such as a lack of sufficient training

data or the need to pre-process data.

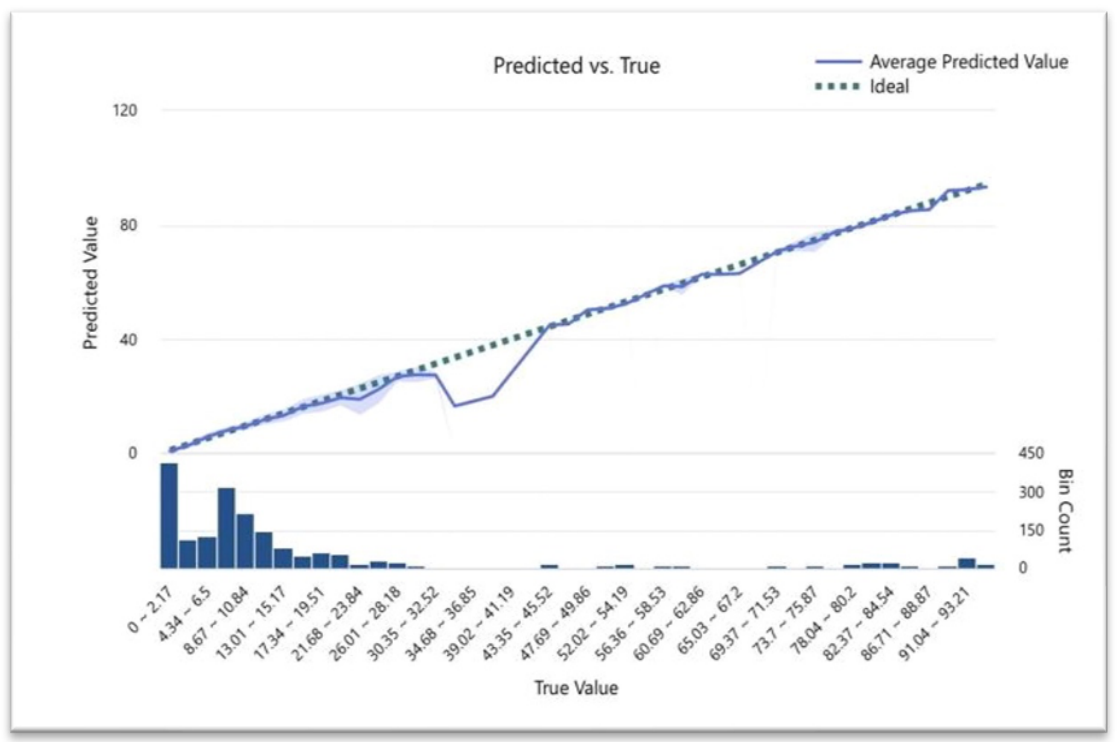


Figure 5.10 : Predicted vs. true for the exponential smoothing algorithm for CPU utilization prediction

Table 5.5 : Calculations of the evaluation metrics of the fuzzy logic model

Total prediction values	TN	TP	FN	FP
500	432	40	25	3
Accuracy	$432+40/500= 94.4\%$			
Precision	$40/(40+3) = 40/43 = 0.93 (93\%)$			
Recall	$40/(40+25) = 0.62 (62\%)$			

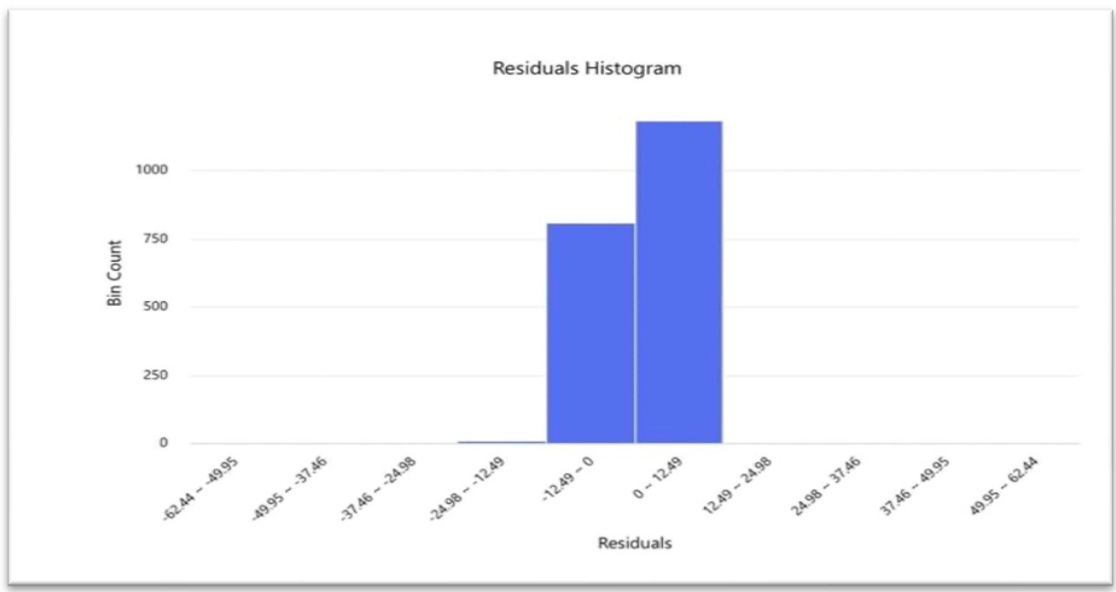


Figure 5.11 : Residuals histogram for the exponential smoothing algorithm for CPU utilization prediction

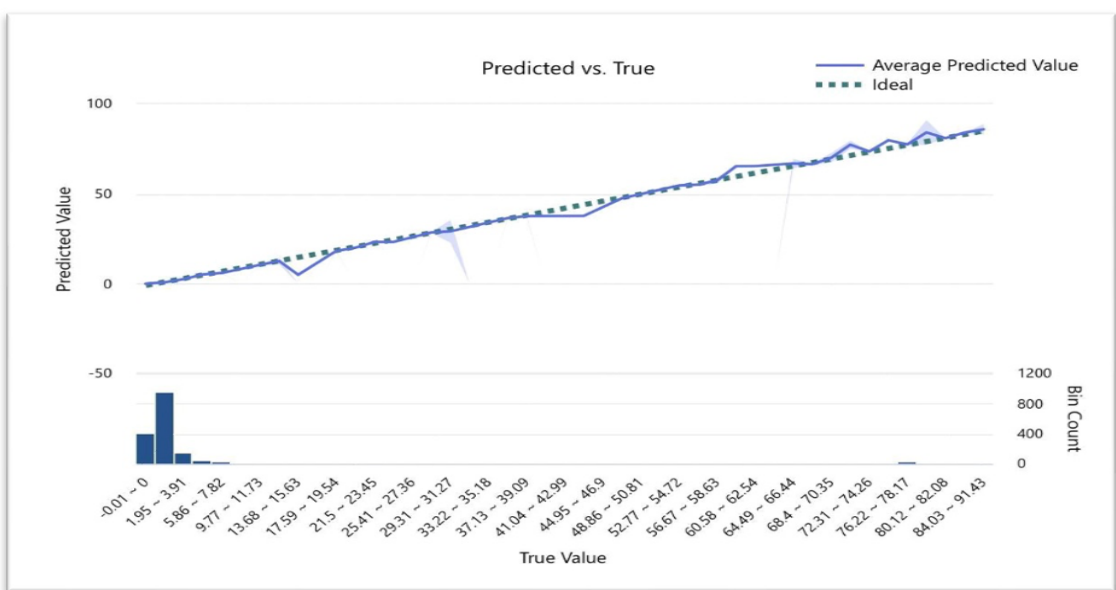


Figure 5.12 : Predicted vs. true for the exponential smoothing algorithm for memory usage prediction

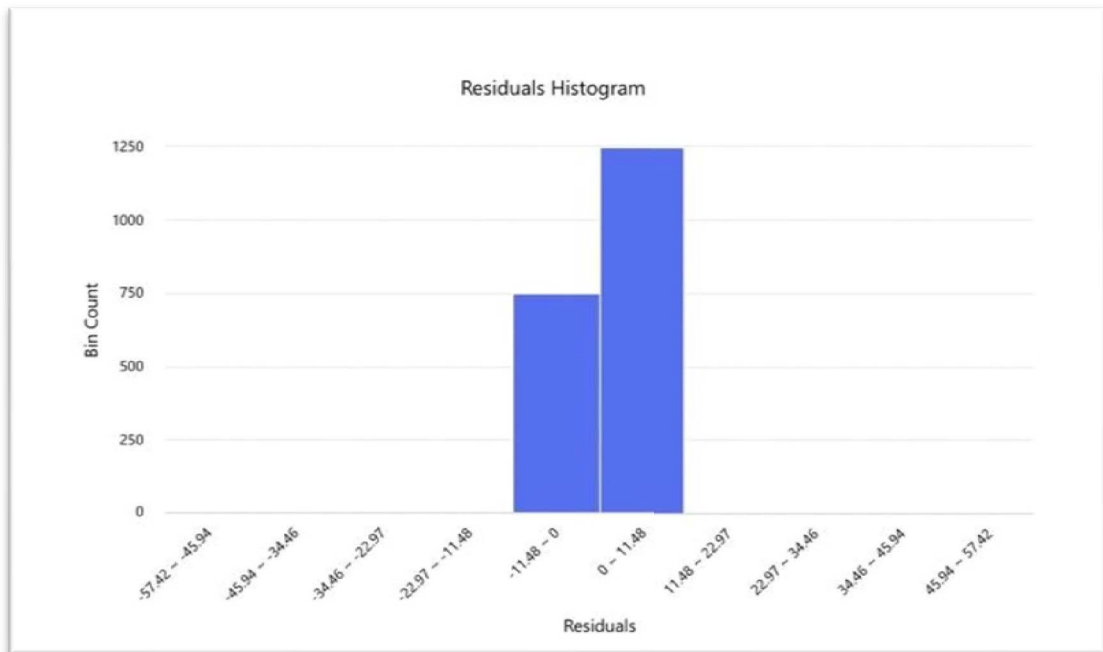


Figure 5.13 : Residuals histogram for the exponential smoothing model for memory usage prediction

ML-Time series prediction				Manually-Overloading decision	Fuzzy logic prediction		Total prediction values	TN	TP	FP	FN
Node-ID	Timestamp	CPU-Utilization Prediction	Memory-Usage Prediction	Overloading decision	Fuzzy logic prediction	Fuzzy Overloading decision					
FNv-1	01-09-2013 00:00:00	31.2	0.1	0	20.1	0	500	432	40	3	25
FNv-1	01-09-2013 00:15:00	31.2	0.1	0	20.1	0	Accuracy	TN+TP/total predictions			94.4
FNv-1	01-09-2013 00:30:00	31.2	0.0	0	20.1	0	Precision	TP/(TP+FP)			93.02325581
FNv-1	01-09-2013 00:45:00	31.2	0.0	0	20.1	0	Recall	TP/(TP+FN)			61.53846154
FNv-1	01-09-2013 01:00:00	31.2	0.1	0	20.1	0					
FNv-2	01-09-2013 00:00:00	30.6	19.5	0	19.6	0					
FNv-2	01-09-2013 00:15:00	30.6	19.5	0	19.6	0					
FNv-2	01-09-2013 00:30:00	30.6	19.9	0	19.6	0					
FNv-2	01-09-2013 00:45:00	30.6	20.1	0	19.6	0					
FNv-2	01-09-2013 01:00:00	30.6	19.5	0	19.6	0					
FNv-3	01-09-2013 00:00:00	73.5	91.7	1	67.2	1					
FNv-3	01-09-2013 00:15:00	73.6	91.7	1	67.2	1					
FNv-3	01-09-2013 00:30:00	73.5	91.7	1	67.2	1					
FNv-3	01-09-2013 00:45:00	73.5	91.7	1	67.2	1					
FNv-3	01-09-2013 01:00:00	73.5	91.7	1	67.2	1					
FNv-4	01-09-2013 00:00:00	67.1	81.4	1	60.3	1					
FNv-4	01-09-2013 00:15:00	65.9	81.3	1	59.8	1					
FNv-4	01-09-2013 00:30:00	65.7	81.4	1	59.8	1					
FNv-4	01-09-2013 00:45:00	66.2	81.3	1	59.9	1					
FNv-4	01-09-2013 01:00:00	67.1	81.4	1	60.3	1					
FNv-5	01-09-2013 00:00:00	57.2	57.7	0	50.0	1					
FNv-5	01-09-2013 00:15:00	57.1	57.7	0	49.9	0					
FNv-5	01-09-2013 00:30:00	57.1	57.7	0	50.0	1					

Figure 5.14 : Snapshot of the calculations of the accuracy of the fuzzy logic model using Excel

5.5 Conclusion

This chapter provides details on the two time-series prediction models that were built to predict the future CPU utilization and memory usage of FNV using ML time series prediction algorithms. The two models are combined using the fuzzy logic model to predict the future workload of the participating FNVs to help those nodes have prior knowledge of when they are going to be overloaded and when they need to offload their next task. Our fuzzy logic model achieves an accuracy of 96%, indicating that the model can be used for future workload prediction.

In the next chapter, the prediction output of FNVs obtained from the prediction models detailed in this chapter are used as input data to develop an incentive module, which is detailed in the next chapter.

The next chapter details the development of an incentive mechanism to encourage fog nodes to participate in the VFC system by sharing their resources to help other resource-constrained fog nodes. The incentive mechanism includes a reward for participation and a penalty for declining to participate. This mechanism helps to increase the number of participating FNVs in the VFC environment and guarantees the availability of sufficient fog nodes at the time of task offloading with enough resources to process the offloaded task.

Chapter 6

An incentive-based framework for task offloading in VFC

6.1 Introduction

Ensuring the availability of a sufficient number of target node vehicles (TNvs) at the time of task offloading is important to ensure the successful execution of the task and to provide a satisfactory level of quality of service (QoS).

Therefore, this chapter details the development of an incentive module to encourage fog node vehicles (FNvs) to participate in the task offloading process. The purpose of the incentive module is to increase the level of participation of FNvs in the task offloading process. This is to ensure the availability of a sufficient number of TNvs that have computation resources and are ready to share their idle resources to execute another vehicle's task

This chapter is outlined as follows: section 6.2 details the proposed iVFC-incentive module. Section 6.3 includes the mathematical formulation of the proposed iVFC-incentive module. Section 6.4 demonstrates the evaluation and validation of the proposed iVFC-incentive module. Section 6.5 discusses the experiments results and section 6.6 concludes this chapter and outlines the work to be covered in the next chapter.

6.2 The proposed framework of the iVFC-incentive module

When an overloaded source node vehicle (SNv) decides to offload its task, it sends the offloading request to the nearest fog server node (FSN) through the iVFC system.

The request contains information related to the task (such as task type, service start time, service end time and needed resources to execute the task). The offloaded task is a specific unit of computation or processing that needs to be performed by a FNV in the iVFC system. This task can be any kind of computation, such as data processing, data analysis, or data storage. The FSN receives the offloading request from the overloaded SNv through the iVFC system, and the following steps are implemented by the corresponding FSN in the incentive module to encourage FNVs that are running on the iVFC system at the time of the offloading request to participate in the task offloading process:

Step 1: The FSN checks the request received from the overloading SNv to determine the computation resources (i.e., the number of the TNvs) needed to process the task based on the information of the task provided in the request.

Step 2: The FSN sets the optimal number of the TNvs needed and determines the range of the reputation value (rewards and penalties) to be offered to the FNVs to encourage them to participate. The offered reputation value is in a form of a reward and penalty formed within a predefined range. We assumed that the optimal number of TNvs determined by the FSN will be more than the actual number of TNvs needed to process the task to guarantee the availability of enough computation resources to process the task.

Step 3: The FSN contacts the FNVs that are available in the system at the time of the offloading request through the iVFC system and offers them the determined reputation (a positive reputation as a reward for agreeing to participate and a negative reputation or zero as a penalty for declining to participate).

Step 4: The FNV checks the offered reputation and calculates its payoff (its new reputation using the offered reputation reward).

Step 5: If the new reputation is greater than or equal to its old reputation, the

FNv agrees with the offered reputation and responds to the FSN request with an approval throughout the iVFC system. A FNv that agrees to participate is known as a TNv after this step. Otherwise, the FNv rejects the participation request and receives the offered penalty.

Step 6: The FSN continues incentivising the FNvs to participate by offering them a different combination of rewards and penalties within the predefined ranges each time until it reaches the optimal number of participating TNvs needed to process the offloaded task.

Step 7: When the FSN reaches the required resources, it prepares a list of all the TNvs that agree to participate and forwards this list to the selection module in preparation for the ranking process.

Figure 6.1 shows the working steps of the proposed iVFC-incentive module. To develop our proposed incentive module, we choose game theory as the incentive mechanism to be used in our framework. There are two types of game theory: cooperative game theory and non-cooperative game theory [82]. Due to the selfishness and rationality of the vehicles that intend to increase their outcomes in the VFC environment, the VFC environment is considered to be a non-cooperative game. Therefore, in our framework, we consider vehicles as players in the game and we follow the solution concept of the Stackelberg Nash equilibrium game theory. The next section presents the mathematical formulation of the proposed incentive module using Stackelberg game theory.

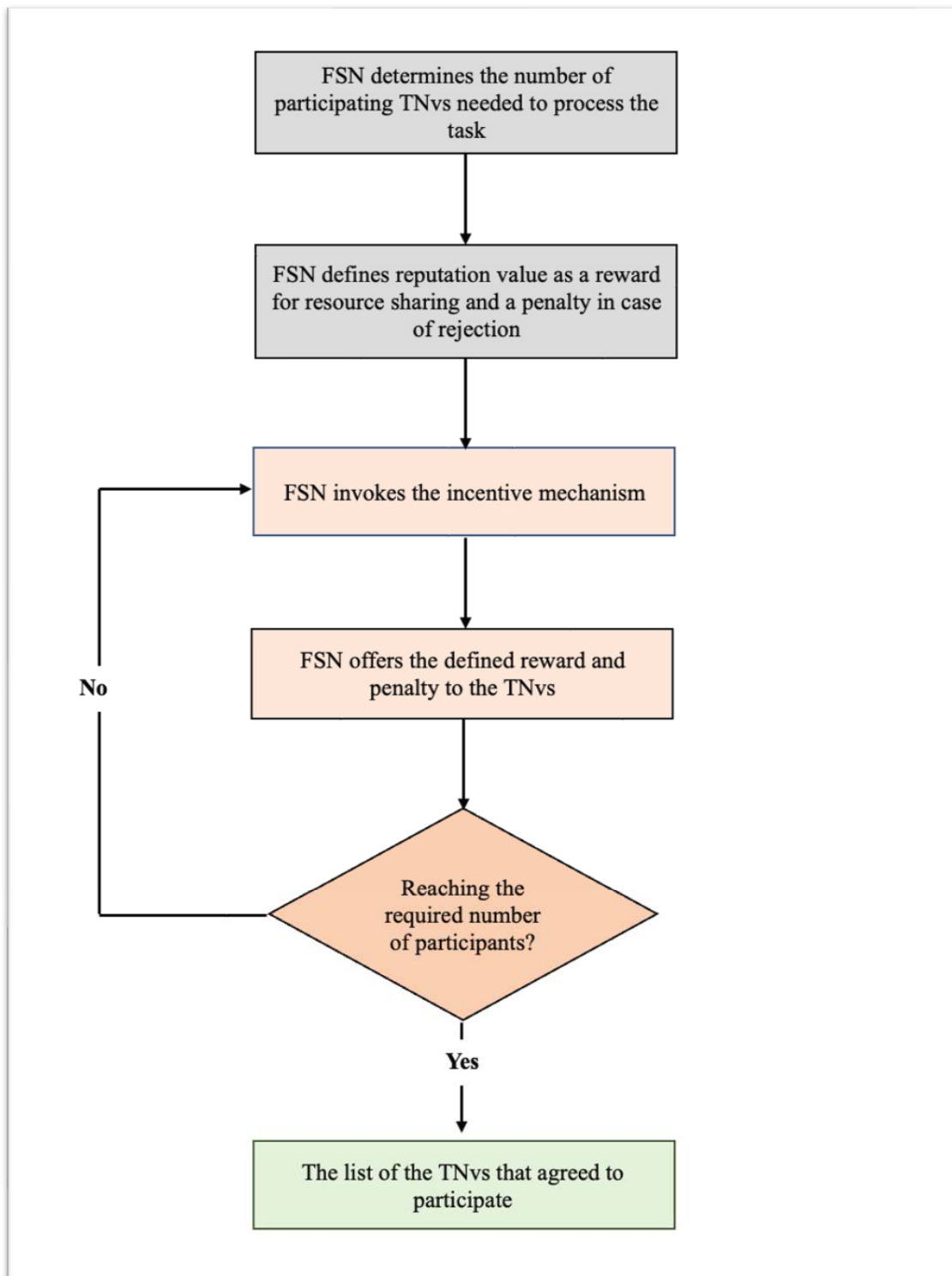


Figure 6.1 : Working steps of the proposed iVFC-incentive module

6.3 Mathematical formulation of the solution for the proposed iVFC-incentive module using Stackelberg game theory

To formulate the mathematical model for the proposed Stackelberg game-based incentive module, we need to define the following:

1. Decision Variables:

- a) Number of participating TNvs: denoted as n_i , where n is the number of participants needed to execute task i and can range from 0 to the total number of needed TNvs.
- b) The offered reputation value: denoted as r for reward and p for penalty, where r is a positive value and p can be 0 or negative.

2. Objective function:

- a) Leader (FSN): The objective of the leader is to minimize the total cost of the task offloading process, which includes the reputation rewards and penalties offered to the participating TNvs. The leader aims to set the reputation value in a way that maximizes the number of participating TNvs.
- b) Followers (FNvs): The objective of the followers is to maximize their own reputation value. They will decide to participate if the offered reputation value will increase their current reputation value or reject otherwise.

3. Constraints:

- a) The number of participating TNvs cannot exceed the total number of FNvs available in the system at the time of task offloading.

- b) The reputation value offered by the leader as a reward cannot be negative, as it is a reward for participation.
- c) The reputation value received by the participating TNv cannot exceed a maximum value, denoted as r_{\max} .
- d) The reputation value offered by the leader as a penalty for rejection is a negative value or 0.

With these definitions, we write the mathematical formulation for the Stackelberg game as follows:

1. Leader's decision:

- a) The leader (FSN) decides on the reputation value to offer to the participating TNvs as following: r for the offered reward, p for the offered penalty.
- b) The leader aims to minimize the total cost of the task offloading process, which can be expressed as follows:
 - $\text{Cost} = n * r$, where n is the number of the required participating TNvs and r is the reputation value offered by the leader.
 - The leader aims to minimize the cost by maximizing the number of participating TNvs.
 - The leader's decision can be expressed as follows:
 - $\min n*r$
 - s.t. $0 < n \leq \text{total number of FNvs}$
 - $1 < r < 6$
 - $r \leq r_{\max}$
 - $-1 \leq p \leq 0$

2. The Follower's decision:

- a) Each follower decides to participate or not based on the offered reputation value and their current reputation value. Each follower calculates its new reputation score based on its old reputation score and the current offered reward using the following formula:

$$new_score = \min(old_score * 0.8 + reward * 0.2, 6) \quad (6.1)$$

,where the new_score does not exceed 6.

To calculate the new reputation score, we use one common approach, which is using a weighted average based on the relative importance of the old and new reputation score [92]. We assign weights for the old and new reputation scores, which can be subjective and depends on the specific system. We assume we assign a weight of 0.8 to the old reputation score and a weight of 0.2 to the new reputation score.

- b) The follower's decision can be expressed as follows:
- If the new reputation value \geq current reputation value, then participate.
 - Otherwise, reject.

3. Equilibrium:

- a) The equilibrium of the Stackelberg game can be found by solving the leader's decision problem and the follower's decision problem simultaneously.
- b) The equilibrium is reached when the leader's decision and the follower's decisions are mutually consistent.
- c) In other words, the leader's decision results in the maximum number of participating TNvs, and each participating TNv has accepted the offered

reputation value. The Nash equilibrium can be expressed as (r, p) where r is the optimal reward and p is the optimal penalty offered by the FSN and gives the optimal number of TNvs.

6.4 Evaluation of the proposed iVFC-incentive module

To evaluate and validate the proposed framework of the incentive module, we implement two scenarios in Google Collaboratory [93]; one scenario using the proposed Stackelberg game-based incentive mechanism to incentivise FNvs to participate, and the other scenario without using the proposed incentive mechanism. We run both scenarios and calculate the level of participation in each scenario for comparison. The following sections explain the different steps involved in the experimental setups for both scenarios and the evaluation of the two scenarios to validate the working of the proposed iVFC-incentive module.

6.4.1 Dataset

The incentive module in this chapter relies on the reputation values of FNvs to determine their decision to accept or reject the offered reward. We assigned a random reputation value to each FNv ranges from 0 to 6, with 6 representing the highest reputation and 0 the lowest. To incorporate this information into the dataset, an additional column is added to include the reputation value for each FNv. This dataset is derived from the workload prediction data of the 100 FNvs generated by the predictive analytic module discussed in Chapter 5.

6.4.2 The selected implementation platform

We implement our experiments on Google Collaboratory. Google Collaboratory is a cloud-based Jupyter notebook environment provided by Google. It allows users to write and execute Python code directly in their web browsers without the need for

local installations. We chose Google Collaboratory to implement our experiments due to its ease of use and accessibility. Google Collaboratory provides a free GPU or TPU (tensor processing unit) runtime, which can significantly speed up computations for machine learning tasks [94]. It also provides an interactive and dynamic environment for executing code and documenting our experiment steps. The seamless integration of Google Collaboratory with Google Drive allows our notebooks and datasets to be stored and accessed, which simplifies data management and helps to back up and access our work from anywhere [94].

6.4.3 The experimental setup and implementation

In Google Collaboratory, we implemented two simulation scenarios. In both scenarios, we use ‘import pandas as pd’ to imports the pandas library and assign it the name pd. Pandas is a powerful library in Python used for data manipulation and analysis [94]. we also use the ‘random’ module from the Python standard library to randomly generate a range of numbers and choose random elements from these ranges.

The first scenario: This scenario uses the proposed incentive mechanism to motivate the FNvs to participate. We implement this scenario using Stackelberg game theory where there is a leader that interacts with a group of followers.

We implement this scenario as follows:

- The leader is the FSN that receives the offloading request, and the followers are the FNvs that are registered in the iVFC system and are available at the time of the offloading request.
- The leader offers a certain reputation for the FNvs, and the followers have their own reputation scores. The goal is to find the optimal combination of reward and penalty values that maximize the number of followers that accept

the leader's offer.

- We assume that the number of the FNvs available at the time of the offloading request are m and the number of TNvs needed to execute the offloaded task i are n_i , where n can be set as required based on the computation resources needed for processing the offloaded task.
- We assume that the leader offers a reputation value in the range $(1.0, 6.0)$ as a reward for the FNv that agrees to participate in the task execution process. We also assume that the leader offers a reputation value in the range $(-1.0, 0.0)$ as a penalty for the FNv that declines to participate.
- We define the desired number of followers to execute the task as the optimal participants, where the optimal participants = n_i .
- We define the maximum number of iterations the model will run as 100 times. The code initializes the number of participants to 0 and creates empty lists to store the results of rewards, penalties, number of accepts, and behaviours (i.e., the behaviours are the reactions of the FNvs in response to the offered reputation, behaviours are: accept or reject).
- A loop is set up to iterate `max_iterations` times or until the optimal number of participants is reached. Inside the loop, a new reward and penalty score for the leader are randomly generated within the ranges specified in the fourth step.
- The behaviour of each follower is determined based on their current reputation score and the offered reward/penalty values. The behaviour is stored in the behaviours list, and the updated reputation scores are stored in the `new_scores` list.

- After the loop, the results (reward, penalty, and number of accepts) for each iteration are appended to their respective lists (rewards, penalties, nu_accepts).
- If the optimal number of participants is reached (i.e., the number of accepts is greater than or equal to optimal_participants) or the final iteration is reached, the loop breaks and the optimal combination of reward and penalty that gave the optimal number of TNvs is printed.

In summary, the code runs a simulation to find the optimal reward and penalty values that maximize the number of followers accepting the leader's offer.

The code of the incentive-based scenario using Stackelberg game can be accessed using the following link: <https://github.com/alhamedy/Stackelberg-game-based-incentive-scenario>.

Algorithm 6.1 shows the logical steps of the simulation for the first scenario using Stackelberg game theory.

The second scenario: This scenario runs without using the proposed incentive mechanism. In Google Collaboratory, we implement another scenario without using rewards and penalties to incentivize the FNvs to participate. Instead, the followers' acceptance is solely based on their reputation score, where a FNv with a score of 4-6 represents acceptance, and a FNv with a score of less than 4 and greater than 0 represents rejection. The followers with different reputation scores make decisions (accept or reject) based on their scores. The goal is to find the optimal number of participants by iterating until the desired number is reached or a maximum number of iterations is reached. Depending on the old score, the behaviour of the FNvs is determined as follows:

- If $4 \leq \text{old score} \leq 6$, the behaviour is set to "accept".
- If $0 \leq \text{old score} < 4$, the behaviour is set to "reject".

Algorithm 6.1 Simulation scenario using Stackelberg game theory-based incentive mechanism

Require: number of FNvs: m , minimum reputation:0, maximum reputation:6, reward range:(1.0,6.0), penalty range:(-1.0,0.0), optimal participants: ni , iterations:100

Ensure: Optimal participants, Nash equilibrium (Reward, Penalty)

```
1: Initialize num_participants, rewards, penalties, num_accepts, behaviours
2: for  $i$  in range(max_iterations) do
3:   reward = random.uniform(*reward_range)
4:   penalty = random.uniform(*penalty_range)
5: end for
6: reputation_scores = [0, 1, 5, ...,  $m$ ]
7: for  $j$  in range(num_participants) do
8:   old_score = reputation_scores[ $j$ ]
9:   new_score = min(old_score * 0.8 + reward * 0.2, 6)
10:  new_scores.append(new_score)
11:  if new_score  $\geq$  old_score then
12:    behaviour is "accept"
13:  else
14:    behaviour is "reject"
15:  end if
16: end for
17: num_accept = behaviours.count("accept")
18: rewards.append(reward)
19: penalties.append(penalty)
20: num_accepts.append(num_accept)
21: if num_accept  $\geq$  optimal_participants then
22:  num_participants = num_accept
```

Simulation scenario using Stackelberg game theory-based incentive mechanism (continued)

```
23: print("Reward =", reward, "and Penalty =", penalty, "is the optimal")
24: break
25: else if num_participants < num_accept then
26:   num_participants = num_accept
27: end if
28: print("Optimal participants:", num_participants)
29: for i in range(len(num_accepts)) do
30:   if num_accepts[i] == num_participants then
31:     print("Reward =", " {:.2f}".format(rewards[i]), "and Penalty =",
           " {:.2f}".format(penalties[i]))
32:   end if
33: end for
```

We assume the number of available FNvs at the time of the offloading request are m and the number of TNvs needed to execute the offloaded task i are n_i , where n can be set as required. We define the desired number of followers to execute the task as the optimal-participants. We define the maximum number of iterations the model will run as 100 times.

In summary, the code runs a simulation to find the optimal number of followers accepting to participate.

The code of the non-incentive scenario can be accessed using the following link: <https://github.com/alhamedy/non-incentive-scenario>.

Algorithm 6.2 shows the logical steps of the simulation for the second scenario without using the incentive mechanism.

Algorithm 6.2 Simulation scenario without using the proposed incentive mechanism

Require: number of FNvs: m , reputation_scores : $r = [0, 1, 2, \dots, rm]$, optimal participants: n , iterations: 100

Ensure: The optimal number of participants n

```
1: Initialize num_participants: 0, num_accepts: empty list, behaviours: empty list
2: for  $i$  in range(max_iterations) do
3:   behaviours = []
4:   for  $j$  in range(num_participants) do
5:     old_score = reputation_scores[ $j$ ]
6:     if old_score  $\geq 4$  then
7:       behaviour is "accept"
8:     else
9:       behaviour is "reject"
10:    end if
11:    behaviours.append(behaviour)
12:  end for
13:  num_accept = behaviours.count("accept")
14:  num_accepts.append(num_accept)
15:  if num_accept  $\geq$  optimal_participants then
16:    num_participants = num_accept
17:    break
18:  else if num_participants < num_accept then
19:    num_participants = num_accept
20:  end if
21: end for
22: print("Optimal participants: num_participants")
```

6.4.4 Evaluation metrics

To evaluate the working of the proposed Stackelberg game-based incentive module, we implement the following steps:

Step 1: We calculate the level of participation in the first scenario that applies the proposed incentive module to encourage FNvs to participate and share their resources to execute the offloaded task. We use the following formula to calculate the level of participation.

$$\text{Level of Participation} = \frac{\text{Number of times the join request is accepted}}{\text{Total number of join requests}} \quad (6.2)$$

Step 2: We calculate the level of participation in the second scenario that does not apply any incentive mechanism to encourage the FNvs to participate using Formula 6.2.

Step 3: We compare the level of participation in both scenarios to evaluate the effectiveness of the proposed Stackelberg game-based incentive module.

6.5 Results and discussion

Table 6.1 summarises the different experiments of the two scenarios that were implemented on Google Collaboratory, the incentive-based scenario using Stackelberg game theory and the non-incentive scenario. We attempted each scenario three times using a different number of m variables (FNvs) and n variables (TNvs) to calculate the level of participation in each scenario using Formula 6.2.

Additionally, Nash equilibrium values are provided for each attempt, indicating the optimal combination of reward and penalty that gives the optimal number of participants.

In the incentive-based scenario, we employed the proposed Stackelberg game-based incentive mechanism. This mechanism aims to encourage FNvs to participate

Table 6.1 : Comparison of Experiments: Stackelberg Game theory-based incentive vs. Non-incentive Scenario

Experiment attempts	Scenario 1: Stackelberg game-based incentive			Scenario 2: non-incentive mechanism		
	Attempt 1	Attempt 2	Attempt 3	Attempt 1	Attempt 2	Attempt 3
The number of available FNvs (m)	10	15	20	10	15	20
The optimal number of needed TNvs (n)	8	13	17	8	13	17
Reward range	(1, 6)	(1, 6)	(1, 6)	NA	NA	NA
Penalty range	(-0.1, 0)	(-0.1, 0)	(-0.1, 0)	NA	NA	NA
The number of iterations to run the model	100	100	100	100	100	100
Iteration when the optimal number of needed TNvs is reached TNvs	3rd	6th	4th	100th	100th	100th
The number of FNvs that agree to participate	8	13	17	3	7	6
The number of the FNvs that decline to participate	2	2	3	7	8	14
Nash equilibrium (optimal reward, optimal penalty)	(2.92, -0.95)	(4.10, -0.74)	(4.92, -0.55)	NA	NA	NA
Level of participation	100%	100%	100%	37.50 %	53.85%	58.82%

in the network by offering them rewards and penalties.

Initially, in the first attempt, we assumed that there are 10 available FNvs at the time of task offloading with a requirement of 8 TNvs for task processing. Upon running the code, random combinations of reward and penalty, within the predetermined range of reward (1 to 6) and penalty (-0.1 to 0), were generated in each iteration and the number of participants was calculated using these combinations for the purpose of reaching the optimal number of participants. We obtained the optimal number of participants in the third iteration with reward = 2.92 and penalty = -0.95, as shown in Figure 6.2 and Figure 6.3. From these figures, it is clear that increasing the reward consistently leads to an increase in the number of participants. However, increasing the penalty also has an impact on participation, causing a reduction in the optimal number of participants. In the first attempt, as shown in Figure 6.2, the reward decreased while the penalty increased in the first iteration, which caused in a reduction in the number of participants. Then, in the second iteration, when the penalty decreased, the reward increased, which caused to increase the number of participants. The number of participants continued to increase until

reaching the optimal participants at the beginning of the third iteration.

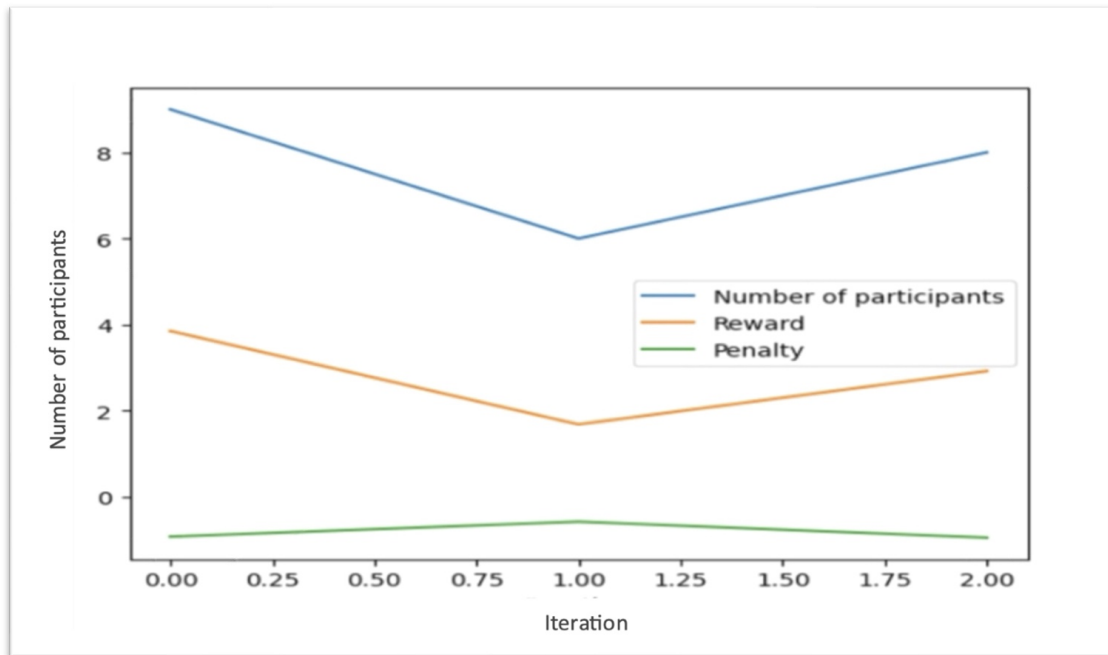


Figure 6.2 : Effect of using the proposed Stackelberg game-based incentive mechanism on the participation level of FNvs (attempt 1, FNvs =10, TNvs =5)

Subsequently, in the second attempt, we increased the number of FNvs to 15 and the number of TNvs needed to process the task to 13. In this case, the optimal number of TNvs was reached by the fourth iteration at a reward = 4.10 and a penalty = -0.74, as shown in Figure 6.4 and Figure 6.5, which gives 100% participation.

Based on Figures 6.4 and 6.5, it is evident that decreasing the reward consistently leads to a reduction in the number of participants, as is clear from the first iteration. Then, when the reward started to increase during the second iteration and the penalty was almost stable, the number of participants increased to more than 14 participants. However, as the reward almost stabilized during the third and the fourth iteration, with a slight reduction in the penalty range, the level of participation also stabilized. Ultimately, in the beginning of iteration 5, the reward started to increase, which resulted to an increase in the participation level. The

```

Iteration: 3
Reward: 2.92, Penalty: -0.95
Node FNV-1: CPU utilisation =31.2%, Memory usage=0.1%, New reputation value=0.58, Decision=accept
Node FNV-2: CPU utilisation =30.6%, Memory usage=19.5%, New reputation value=6.00, Decision=accept
Node FNV-5: CPU utilisation =57.2%, Memory usage=57.7%, New reputation value=2.18, Decision=accept
Node FNV-6: CPU utilisation =0.0%, Memory usage=8.8%, New reputation value=1.38, Decision=accept
Node FNV-7: CPU utilisation =0.0%, Memory usage=12.7%, New reputation value=2.98, Decision=reject
Node FNV-8: CPU utilisation =47.5%, Memory usage=50.4%, New reputation value=4.58, Decision=reject
Node FNV-10: CPU utilisation =0.0%, Memory usage=12.1%, New reputation value=6.00, Decision=accept
Node FNV-12: CPU utilisation =1.0%, Memory usage=7.3%, New reputation value=0.58, Decision=accept
Node FNV-13: CPU utilisation =0.0%, Memory usage=18.6%, New reputation value=1.38, Decision=accept
Node FNV-14: CPU utilisation =0.0%, Memory usage=8.1%, New reputation value=2.18, Decision=accept
Level of participation: 100.00%
-----
Reward: 2.92, Penalty: -0.95, Participants: 8
-----
Reward=2.92 and Penalty=-0.95 is the optimal Nash equilibrium
Optimal participants: 8
Reward = 2.92 and Penalty = -0.95
CPU Utilization, Memory Usage, and Reputation of Participating Nodes:
Node FNV-1: CPU=31.2%, Memory=0.1%, Reputation=0
Node FNV-2: CPU=30.6%, Memory=19.5%, Reputation=6
Node FNV-5: CPU=57.2%, Memory=57.7%, Reputation=2
Node FNV-6: CPU=0.0%, Memory=8.8%, Reputation=1
Node FNV-10: CPU=0.0%, Memory=12.1%, Reputation=6
Node FNV-12: CPU=1.0%, Memory=7.3%, Reputation=0
Node FNV-13: CPU=0.0%, Memory=18.6%, Reputation=1
Node FNV-14: CPU=0.0%, Memory=8.1%, Reputation=2
Level of participation: 100.00%

```

Figure 6.3 : Snapshot of the implementation of proposed the Stackelberg game-based incentive mechanism in Google Collab., attempt 1

optimal number of participants was achieved in the beginning of iteration 6, with a reward = 4.10 and a penalty = -0.74, which gives 100% participation.

In our final attempt, we further increased the number of available FNVs to 20 and the number of TNVs needed to process the task to 17. In this attempt, the number of optimal TNVs was reached at iteration 4 with a reward = 4.92 and a penalty = -0.55, as shown in Figure 6.6 and Figure 6.7.

In this attempt, the reward initially increased during the first iteration while the penalty remained relatively constant, resulting in an increase in the number of participants. During the second iteration, both the reward and penalty were almost stabilized, leading to a stabilization of the level of participation. During the third iteration, there was an increase in the reward and penalty, which resulted in an increase in the level of participation. Ultimately, at the beginning of iteration 4, the

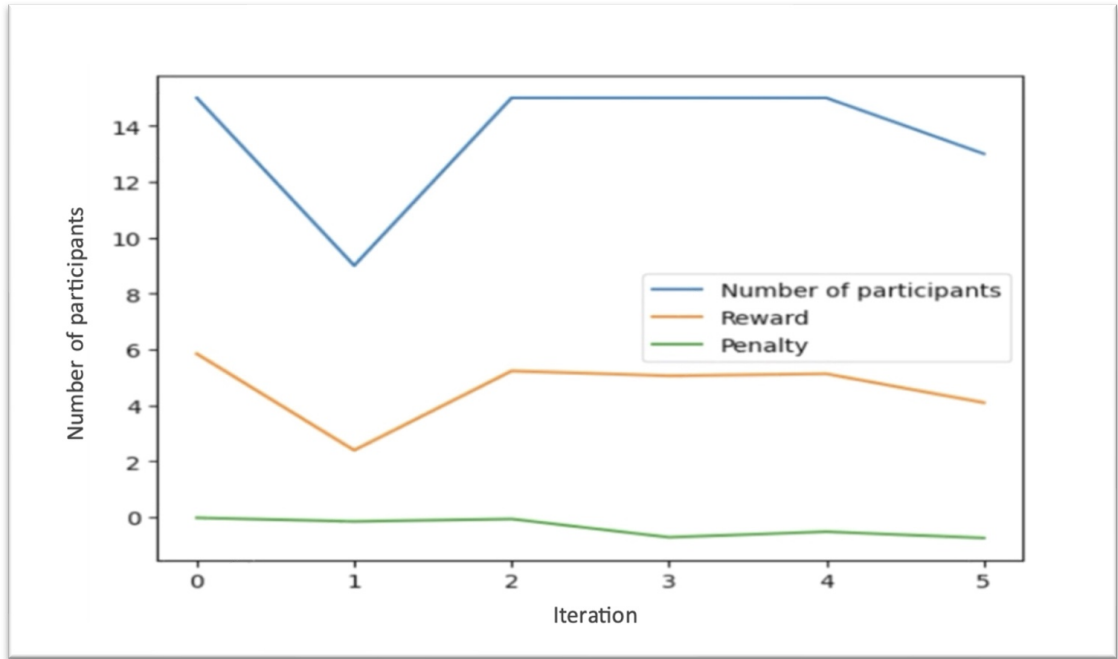


Figure 6.4 : Effect of using the proposed Stackelberg game-based incentive mechanism on the participation level of FNvs (attempt 2, FNvs =15, TNvs =10)

optimal number of participants was achieved, with a reward = 4.92 and a penalty = -0.55 with 100% participation.

As shown in Table 6.1 and Figures 6.2 to 6.7, it is clear that applying the incentive mechanism positively affects the level of participation. Using a different number of FNvs and TNvs with different combination of rewards and penalties, our proposed incentive-based mechanism has proven its effectiveness in incentivising the required number of participants to process the offloaded task and achieving a level of participation = 100% in all experiments.

In the second scenario, the proposed Stackelberg game based-incentive mechanism is not used, which means that the FNvs do not have any incentive to participate. In this scenario, the participation of the FNvs is based on their reputation value where the nodes that have a high reputation value in the network will always

```

Iteration: 6
Reward: 4.10, Penalty: -0.74
Node FNV-1: CPU utilisation =31.2%, Memory usage=0.1%, New reputation value=0.82, Decision=accept
Node FNV-2: CPU utilisation =30.6%, Memory usage=19.5%, New reputation value=1.62, Decision=accept
Node FNV-5: CPU utilisation =57.2%, Memory usage=57.7%, New reputation value=6.00, Decision=accept
Node FNV-6: CPU utilisation =0.0%, Memory usage=8.8%, New reputation value=3.22, Decision=accept
Node FNV-7: CPU utilisation =0.0%, Memory usage=12.7%, New reputation value=4.02, Decision=accept
Node FNV-8: CPU utilisation =47.5%, Memory usage=50.4%, New reputation value=4.82, Decision=reject
Node FNV-10: CPU utilisation =0.0%, Memory usage=12.1%, New reputation value=6.00, Decision=accept
Node FNV-12: CPU utilisation =1.0%, Memory usage=7.3%, New reputation value=0.82, Decision=accept
Node FNV-13: CPU utilisation =0.0%, Memory usage=18.6%, New reputation value=1.62, Decision=accept
Node FNV-14: CPU utilisation =0.0%, Memory usage=8.1%, New reputation value=2.42, Decision=accept
Node FNV-15: CPU utilisation =2.9%, Memory usage=8.4%, New reputation value=4.82, Decision=reject
Node FNV-16: CPU utilisation =0.0%, Memory usage=7.5%, New reputation value=6.00, Decision=accept
Node FNV-17: CPU utilisation =0.0%, Memory usage=7.9%, New reputation value=4.02, Decision=accept
Node FNV-18: CPU utilisation =0.0%, Memory usage=24.8%, New reputation value=3.22, Decision=accept
Node FNV-19: CPU utilisation =0.0%, Memory usage=10.7%, New reputation value=2.42, Decision=accept
Level of participation: 100.00%

-----
Reward: 4.10, Penalty: -0.74, Participants: 13
-----
Reward=4.10 and Penalty=-0.74 is the optimal Nash equilibrium
Optimal participants: 13
Reward = 4.10 and Penalty = -0.74
CPU Utilization, Memory Usage, and Reputation of Participating Nodes:
Node FNV-1: CPU=31.2%, Memory=0.1%, Reputation=0
Node FNV-2: CPU=30.6%, Memory=19.5%, Reputation=1
Node FNV-5: CPU=57.2%, Memory=57.7%, Reputation=6
Node FNV-6: CPU=0.0%, Memory=8.8%, Reputation=3
Node FNV-7: CPU=0.0%, Memory=12.7%, Reputation=4
Node FNV-10: CPU=0.0%, Memory=12.1%, Reputation=6
Node FNV-12: CPU=1.0%, Memory=7.3%, Reputation=0
Node FNV-13: CPU=0.0%, Memory=18.6%, Reputation=1
Node FNV-14: CPU=0.0%, Memory=8.1%, Reputation=2
Node FNV-16: CPU=0.0%, Memory=7.5%, Reputation=6
Node FNV-17: CPU=0.0%, Memory=7.9%, Reputation=4
Node FNV-18: CPU=0.0%, Memory=24.8%, Reputation=3
Node FNV-19: CPU=0.0%, Memory=10.7%, Reputation=2
Level of participation: 100.00%

```

Figure 6.5 : Snapshot of the implementation of proposed the Stackelberg game-based incentive mechanism in Google Collab., attempt 2

participate, while other nodes with a very low reputation mostly declined to participate. We implement this scenario using the same number of FNvs and TNvs that we assumed in the incentive-based scenario without any offered rewards or penalties. As shown in Table 6.1, it can be clearly seen that we did not reach the optimal number of participants at any attempts and the level of participation changed according to the reputation score of the available FNvs. That is, if the reputation score of the FNV is high, the node always agrees to participate. On the other hand, the nodes with low reputation scores declines to participate.

As shown in Figures 6.8 to 6.13, the number of participants remains the same in all iterations for all 3 attempts. the optimal number of participants was not reached in any iteration for any attempt.

Overall, in the incentive scenario, the optimal number of needed TNvs is reached

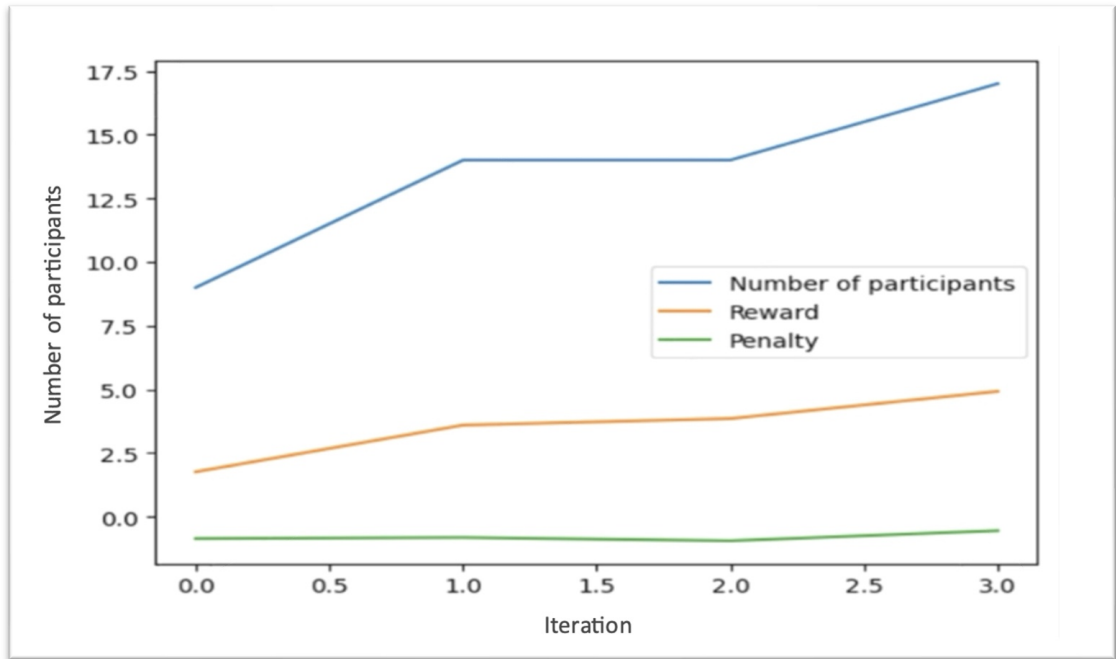


Figure 6.6 : Effect of using the proposed Stackelberg game-based incentive mechanism on the participation level of FNvs (attempt 3, FNvs =20, TNvs =13)

in all attempts. Generally speaking, the proposed incentive mechanism motivates FNvs to join the network, resulting in a 100% participation rate. On the other hand, the level of participation in the second scenario that does not have an incentive mechanism is mostly low as it depends on the reputation score of the FNvs, where only the nodes that have a high reputation value in the network will always agree to participate and share their resources.

```

Iteration: 4
Reward: 4.92, Penalty: -0.55
Node FNV-1: CPU utilisation =31.2%, Memory usage=0.1%, New reputation value=0.98, Decision=accept
Node FNV-2: CPU utilisation =30.6%, Memory usage=19.5%, New reputation value=1.78, Decision=accept
Node FNV-5: CPU utilisation =57.2%, Memory usage=57.7%, New reputation value=6.00, Decision=accept
Node FNV-6: CPU utilisation =0.0%, Memory usage=8.8%, New reputation value=3.38, Decision=accept
Node FNV-7: CPU utilisation =0.0%, Memory usage=12.7%, New reputation value=4.18, Decision=accept
Node FNV-8: CPU utilisation =47.5%, Memory usage=50.4%, New reputation value=4.98, Decision=reject
Node FNV-10: CPU utilisation =0.0%, Memory usage=12.1%, New reputation value=6.00, Decision=accept
Node FNV-12: CPU utilisation =1.0%, Memory usage=7.3%, New reputation value=0.98, Decision=accept
Node FNV-13: CPU utilisation =0.0%, Memory usage=18.6%, New reputation value=1.78, Decision=accept
Node FNV-14: CPU utilisation =0.0%, Memory usage=8.1%, New reputation value=2.58, Decision=accept
Node FNV-15: CPU utilisation =2.9%, Memory usage=8.4%, New reputation value=4.98, Decision=reject
Node FNV-16: CPU utilisation =0.0%, Memory usage=7.5%, New reputation value=6.00, Decision=accept
Node FNV-17: CPU utilisation =0.0%, Memory usage=7.9%, New reputation value=4.18, Decision=accept
Node FNV-18: CPU utilisation =0.0%, Memory usage=24.8%, New reputation value=3.38, Decision=accept
Node FNV-19: CPU utilisation =0.0%, Memory usage=10.7%, New reputation value=2.58, Decision=accept
Node FNV-20: CPU utilisation =8.7%, Memory usage=9.2%, New reputation value=3.38, Decision=accept
Node FNV-24: CPU utilisation =0.1%, Memory usage=52.3%, New reputation value=4.18, Decision=accept
Node FNV-26: CPU utilisation =0.1%, Memory usage=45.0%, New reputation value=4.98, Decision=reject
Node FNV-27: CPU utilisation =0.1%, Memory usage=2.2%, New reputation value=6.00, Decision=accept
Node FNV-28: CPU utilisation =1.0%, Memory usage=0.7%, New reputation value=0.98, Decision=accept
Level of participation: 100.00%

-----
Reward: 4.92, Penalty: -0.55, Participants: 17
-----
Reward=4.92 and Penalty=-0.55 is the optimal Nash equilibrium
Optimal participants: 17
Reward = 4.92 and Penalty = -0.55
CPU Utilization, Memory Usage, and Reputation of Participating Nodes:
Node FNV-1: CPU=31.2%, Memory=0.1%, Reputation=0
Node FNV-2: CPU=30.6%, Memory=19.5%, Reputation=1
Node FNV-5: CPU=57.2%, Memory=57.7%, Reputation=6
Node FNV-6: CPU=0.0%, Memory=8.8%, Reputation=3
Node FNV-7: CPU=0.0%, Memory=12.7%, Reputation=4
Node FNV-10: CPU=0.0%, Memory=12.1%, Reputation=6
Node FNV-12: CPU=1.0%, Memory=7.3%, Reputation=0
Node FNV-13: CPU=0.0%, Memory=18.6%, Reputation=1
Node FNV-14: CPU=0.0%, Memory=8.1%, Reputation=2
Node FNV-16: CPU=0.0%, Memory=7.5%, Reputation=6
Node FNV-17: CPU=0.0%, Memory=7.9%, Reputation=4
Node FNV-18: CPU=0.0%, Memory=24.8%, Reputation=3
Node FNV-19: CPU=0.0%, Memory=10.7%, Reputation=2
Node FNV-20: CPU=8.7%, Memory=9.2%, Reputation=3
Node FNV-24: CPU=0.1%, Memory=52.3%, Reputation=4
Node FNV-27: CPU=0.1%, Memory=2.2%, Reputation=6
Node FNV-28: CPU=1.0%, Memory=0.7%, Reputation=0
Level of participation: 100.00%
    
```

Figure 6.7 : Snapshot of the implementation of proposed the Stackelberg game-based incentive mechanism in Google Collab., attempt 3

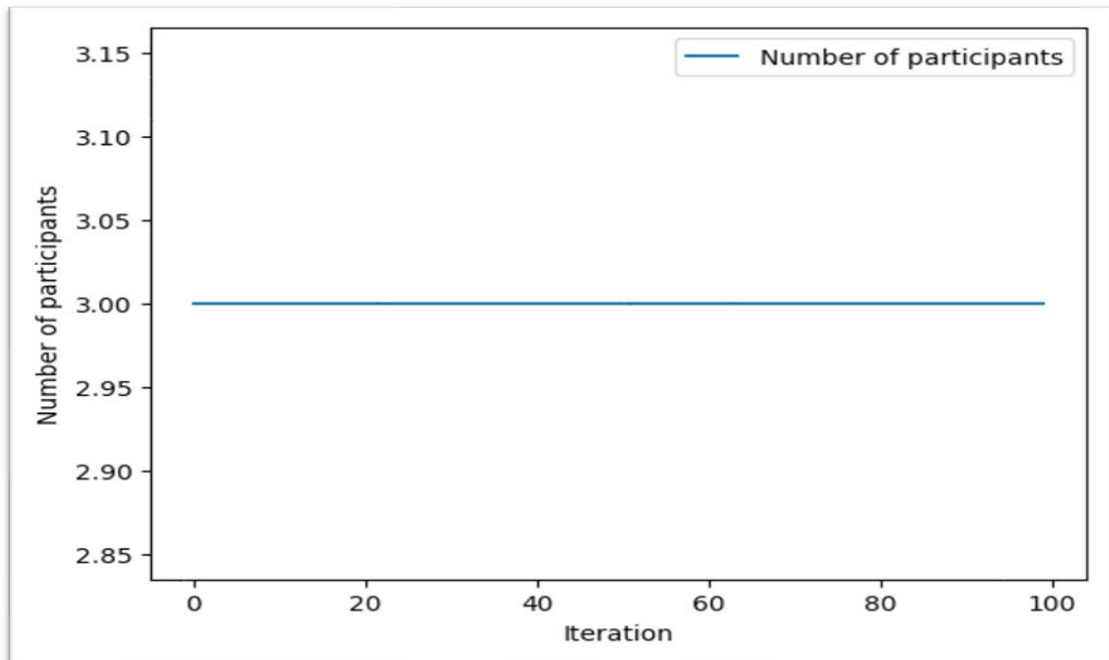


Figure 6.8 : Non-incentive scenario (attempt 1, FNvs =10, TNvs =8)

```

+ Code + Text

Iteration: 100
Node FNv-1: CPU utilization = 31.2%, Memory usage = 0.1%, Reputation value = 0, Decision = reject
Node FNv-2: CPU utilization = 30.6%, Memory usage = 19.5%, Reputation value = 6, Decision = accept
Node FNv-5: CPU utilization = 57.2%, Memory usage = 57.7%, Reputation value = 2, Decision = reject
Node FNv-6: CPU utilization = 0.0%, Memory usage = 8.8%, Reputation value = 1, Decision = reject
Node FNv-7: CPU utilization = 0.0%, Memory usage = 12.7%, Reputation value = 3, Decision = reject
Node FNv-8: CPU utilization = 47.5%, Memory usage = 50.4%, Reputation value = 5, Decision = accept
Node FNv-10: CPU utilization = 0.0%, Memory usage = 12.1%, Reputation value = 6, Decision = accept
Node FNv-12: CPU utilization = 1.0%, Memory usage = 7.3%, Reputation value = 0, Decision = reject
Node FNv-13: CPU utilization = 0.0%, Memory usage = 18.6%, Reputation value = 1, Decision = reject
Node FNv-14: CPU utilization = 0.0%, Memory usage = 8.1%, Reputation value = 2, Decision = reject
Level of participation: 37.50%

-----
Highest number of participants: 3
CPU Utilization, Memory Usage, and Reputation of Participating Nodes:
Node FNv-2: CPU = 30.6%, Memory = 19.5%, Reputation = 6
Node FNv-8: CPU = 47.5%, Memory = 50.4%, Reputation = 5
Node FNv-10: CPU = 0.0%, Memory = 12.1%, Reputation = 6
Level of participation: 37.50%
    
```

Figure 6.9 : Snapshot of the implementation of the non-incentive scenario in Google Collab., attempt 1

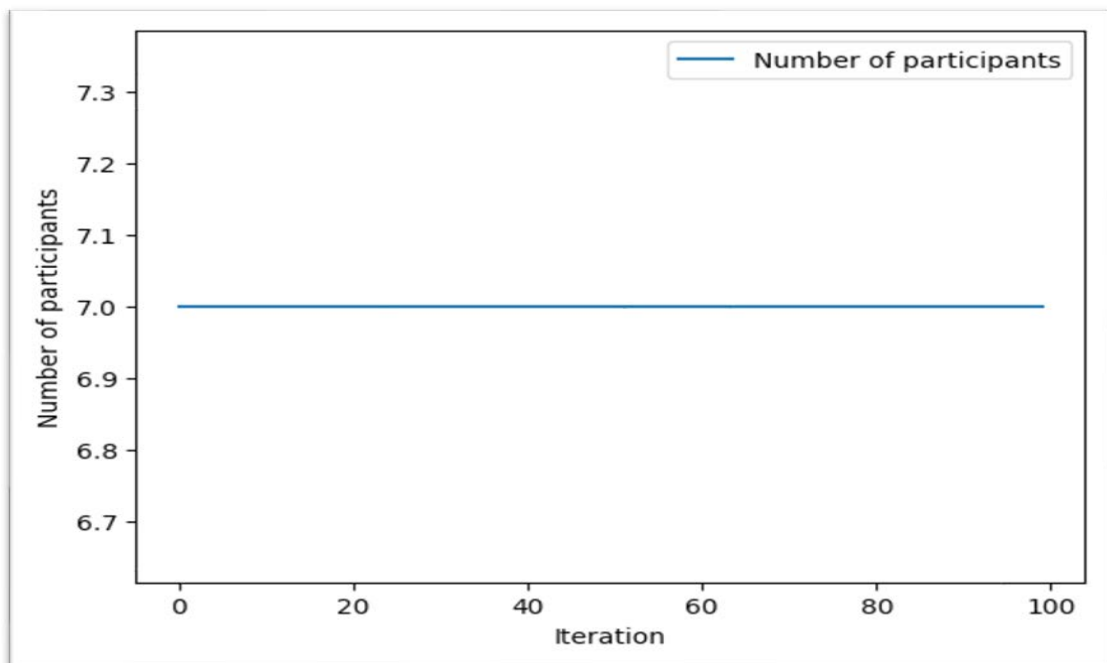


Figure 6.10 : Non-incentive scenario (attempt 2, FNvs =15, TNvs =13)

```

Non-incentive scenario, attempt2.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Iteration: 100
Node FNV-1: CPU utilization = 31.2%, Memory usage = 0.1%, Reputation value = 0, Decision = reject
Node FNV-2: CPU utilization = 30.6%, Memory usage = 19.5%, Reputation value = 1, Decision = reject
Node FNV-5: CPU utilization = 57.2%, Memory usage = 57.7%, Reputation value = 6, Decision = accept
Node FNV-6: CPU utilization = 0.0%, Memory usage = 8.8%, Reputation value = 3, Decision = reject
Node FNV-7: CPU utilization = 0.0%, Memory usage = 12.7%, Reputation value = 4, Decision = accept
Node FNV-8: CPU utilization = 47.5%, Memory usage = 50.4%, Reputation value = 5, Decision = accept
Node FNV-10: CPU utilization = 0.0%, Memory usage = 12.1%, Reputation value = 6, Decision = accept
Node FNV-12: CPU utilization = 1.0%, Memory usage = 7.3%, Reputation value = 0, Decision = reject
Node FNV-13: CPU utilization = 0.0%, Memory usage = 18.6%, Reputation value = 1, Decision = reject
Node FNV-14: CPU utilization = 0.0%, Memory usage = 8.1%, Reputation value = 2, Decision = reject
Node FNV-15: CPU utilization = 2.9%, Memory usage = 8.4%, Reputation value = 5, Decision = accept
Node FNV-16: CPU utilization = 0.0%, Memory usage = 7.5%, Reputation value = 6, Decision = accept
Node FNV-17: CPU utilization = 0.0%, Memory usage = 7.9%, Reputation value = 4, Decision = accept
Node FNV-18: CPU utilization = 0.0%, Memory usage = 24.8%, Reputation value = 3, Decision = reject
Node FNV-19: CPU utilization = 0.0%, Memory usage = 10.7%, Reputation value = 2, Decision = reject
Level of participation: 53.85%

-----
Highest number of participants: 7
CPU Utilization, Memory Usage, and Reputation of Participating Nodes:
Node FNV-5: CPU = 57.2%, Memory = 57.7%, Reputation = 6
Node FNV-7: CPU = 0.0%, Memory = 12.7%, Reputation = 4
Node FNV-8: CPU = 47.5%, Memory = 50.4%, Reputation = 5
Node FNV-10: CPU = 0.0%, Memory = 12.1%, Reputation = 6
Node FNV-15: CPU = 2.9%, Memory = 8.4%, Reputation = 5
Node FNV-16: CPU = 0.0%, Memory = 7.5%, Reputation = 6
Node FNV-17: CPU = 0.0%, Memory = 7.9%, Reputation = 4
Level of participation: 53.85%
    
```

Figure 6.11 : Snapshot of the implementation of the non-incentive mechanism in Google Collab., attempt 2

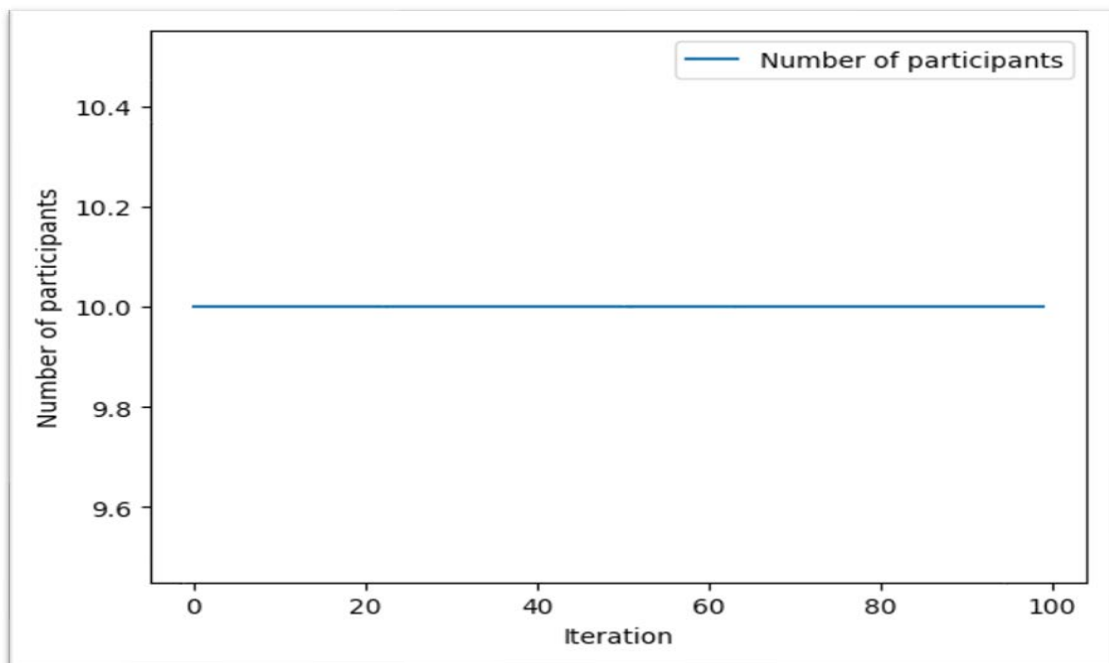


Figure 6.12 : Non-incentive scenario (attempt 3, FNvs =20, TNvs =17)

```

Iteration: 100
Node FNV-1: CPU utilization = 31.2%, Memory usage = 0.1%, Reputation value = 0, Decision = reject
Node FNV-2: CPU utilization = 30.6%, Memory usage = 19.5%, Reputation value = 1, Decision = reject
Node FNV-5: CPU utilization = 57.2%, Memory usage = 57.7%, Reputation value = 6, Decision = accept
Node FNV-6: CPU utilization = 0.0%, Memory usage = 8.8%, Reputation value = 3, Decision = reject
Node FNV-7: CPU utilization = 0.0%, Memory usage = 12.7%, Reputation value = 4, Decision = accept
Node FNV-8: CPU utilization = 47.5%, Memory usage = 50.4%, Reputation value = 5, Decision = accept
Node FNV-10: CPU utilization = 0.0%, Memory usage = 12.1%, Reputation value = 6, Decision = accept
Node FNV-12: CPU utilization = 1.0%, Memory usage = 7.3%, Reputation value = 0, Decision = reject
Node FNV-13: CPU utilization = 0.0%, Memory usage = 18.6%, Reputation value = 1, Decision = reject
Node FNV-14: CPU utilization = 0.0%, Memory usage = 8.1%, Reputation value = 2, Decision = reject
Node FNV-15: CPU utilization = 2.9%, Memory usage = 8.4%, Reputation value = 5, Decision = accept
Node FNV-16: CPU utilization = 0.0%, Memory usage = 7.5%, Reputation value = 6, Decision = accept
Node FNV-17: CPU utilization = 0.0%, Memory usage = 7.9%, Reputation value = 4, Decision = accept
Node FNV-18: CPU utilization = 0.0%, Memory usage = 24.8%, Reputation value = 3, Decision = reject
Node FNV-19: CPU utilization = 0.0%, Memory usage = 10.7%, Reputation value = 2, Decision = reject
Node FNV-20: CPU utilization = 8.7%, Memory usage = 9.2%, Reputation value = 3, Decision = reject
Node FNV-24: CPU utilization = 0.1%, Memory usage = 52.3%, Reputation value = 4, Decision = accept
Node FNV-26: CPU utilization = 0.1%, Memory usage = 45.0%, Reputation value = 5, Decision = accept
Node FNV-27: CPU utilization = 0.1%, Memory usage = 2.2%, Reputation value = 6, Decision = accept
Node FNV-28: CPU utilization = 1.0%, Memory usage = 0.7%, Reputation value = 0, Decision = reject
Level of participation: 58.82%
-----
Highest number of participants: 10
CPU Utilization, Memory Usage, and Reputation of Participating Nodes:
Node FNV-5: CPU = 57.2%, Memory = 57.7%, Reputation = 6
Node FNV-7: CPU = 0.0%, Memory = 12.7%, Reputation = 4
Node FNV-8: CPU = 47.5%, Memory = 50.4%, Reputation = 5
Node FNV-10: CPU = 0.0%, Memory = 12.1%, Reputation = 6
Node FNV-15: CPU = 2.9%, Memory = 8.4%, Reputation = 5
Node FNV-16: CPU = 0.0%, Memory = 7.5%, Reputation = 6
Node FNV-17: CPU = 0.0%, Memory = 7.9%, Reputation = 4
Node FNV-24: CPU = 0.1%, Memory = 52.3%, Reputation = 4
Node FNV-26: CPU = 0.1%, Memory = 45.0%, Reputation = 5
Node FNV-27: CPU = 0.1%, Memory = 2.2%, Reputation = 6
Level of participation: 58.82%

```

Figure 6.13 : Snapshot of the implementation of the non-incentive mechanism in Google Collab., attempt 3

6.6 Conclusion

In this chapter, a Stackelberg game-based incentive module is developed to encourage FNvs to participate in the task offloading process. The purpose of the incentive module is to increase the level of participation of FNvs in the task offloading process. To evaluate the working of the proposed incentive module in achieving the required number of participants during the task offloading process, we implemented two scenarios: one scenario utilised the proposed incentive mechanism and the other scenario did not use incentives to encourage FNvs to participate and their participation is based on their reputation value where the nodes that have a high reputation value in the network will always participate, while the other nodes with a very low reputation mostly decline to participate. After conducting various experiments in both scenarios, we concluded that the proposed Stackelberg game-based incentive

mechanism contributes to motivate FNvs to join the network, resulting in a 100% participation rate.

The next chapter details a target node vehicle (TNv) selection mechanism to help the overloaded source node vehicle (SNv) to find the most optimal TNv that can handle the task effectively. The purpose of the developed selection mechanism is to rank the list of TNvs that agree to participate in the task offloading process based on their workload variables (CPU utilization and memory usage). This list is prepared by the incentive module that was described in this chapter to encourage FNvs to serve as fog nodes to participate in the task offloading process.

Chapter 7

An intelligent framework for target node vehicle selection in the iVFC system

7.1 Introduction

Ensuring the successful execution of an offloaded task and maintaining the desired quality of service (QoS) requires a robust selection method to identify the most optimal service provider node that can handle the task effectively. Therefore, our second research objective is to develop an effective mechanism for target node vehicle (TNv) selection to help the overloaded node select the most optimal TNv to carry out the offloaded task successfully.

This chapter overviews the steps involved in developing a fog service provider selection (TNv selection) framework for task offloading in vehicular fog computing (VFC) using three methods to address research objective 3. The purpose of this framework is to assist the overloaded source node vehicle (SNv) to find the most optimal TNv that can efficiently handle the offloaded task. This framework comprises three phases, which are discussed in detail in this chapter. This chapter also includes the validation steps of the proposed framework to address the second part of research objective 4, which validates the solution to research objective 3.

This chapter is organised as follows: Section 7.2 explains the steps followed in developing the TNv selection module using three methods. Section 7.3 describes the experiments to evaluate and validate the selection module. Section 7.4 discusses the results and section 7.5 concludes this chapter and outlines the work to be covered in the next chapter.

7.2 The proposed framework of the iVFC-TNv selection module

When an SNv becomes overloaded and decides to offload its task to another fog node in the iVFC system for execution, it will send the offloading request to the nearest fog server node (FSN) including all the information related to task type and the resources required to execute the task. The corresponding FSN receives the offloading request from the overloaded SNv and processes the request in the incentive module to obtain a list of all the TNvs in the system that agreed to participate (Chapter 6 gives a detailed explanation as to how the request is processed in the incentive module). Then, the following steps are implemented by the corresponding FSN in the selection module through the iVFC system to select the most optimal TNv to handle the task:

Step 1: The FSN retrieves the list of all the TNvs from the incentive module and redirects the list to the selection module. This list contains information on each TNv including their current CPU utilization and memory usage which is continually updated by the prediction module.

Step 2: In the selection module, the ranking algorithm is applied to rank the TNvs according to their current CPU utilization and memory usage data from lowest to highest (the TNv that has less CPU utilization and has used less memory will have a higher ranking).

Step 3: The list is stored in the optimal TNvs' data file and the FSN chooses the most optimal TNv/TNvs from the list to handle the task (usually this is the top TNvs on the list). Figure 7.1 shows the proposed framework of the iVFC-TNv selection module.

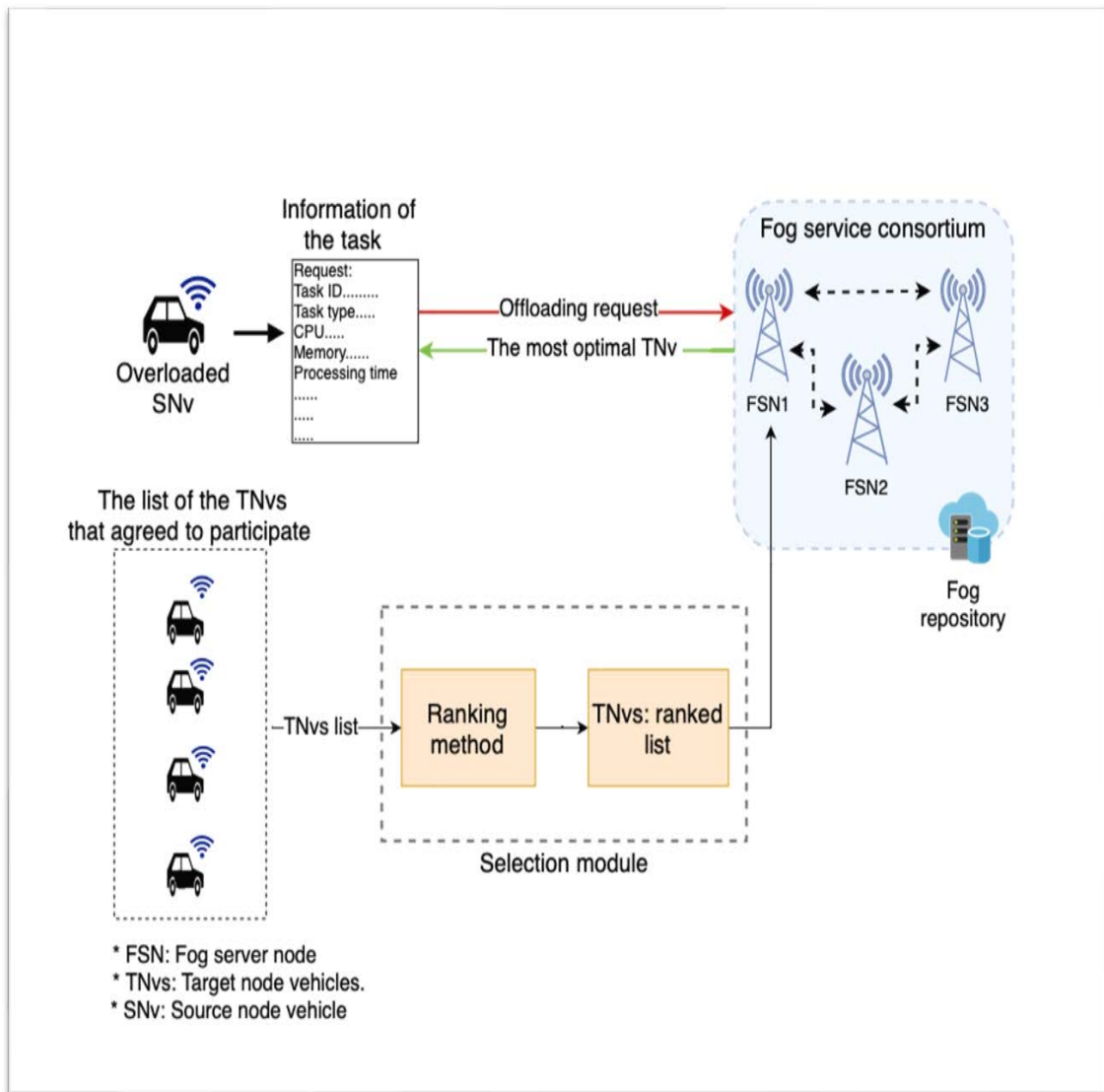


Figure 7.1 : Framework of the iVFC-TNv selection module

7.3 Evaluation of the proposed iVFC-TNv selection module

7.3.1 Dataset

The dataset used to develop the TNv selection module is the workload prediction data of the FNVs that were generated from the predictive analytic module discussed in Chapter 5. This dataset includes the predicted CPU utilization and the predicted memory usage of 100 FNVs generated for an interval of one hour at a frequency of

15 minutes, which generates 5 predicted values for each FNv. Of the five predicted values for each FNv, we choose one for the implementation of the TNv selection module.

As we utilize both ML methods and DL methods in the implementation of the selection module, which needs a large dataset to train and test the model, we generate a synthetic dataset from the dataset of 100 FNvs to increase it to 2000 rows using Python.

The Python code used to generate the synthetic data can be found at the following GitHub link: <https://github.com/alhamedy/Synthetic-data-generation>.

The dataset can be accessed at the following GitHub link: <https://github.com/alhamedy/DL-dataset>.

Furthermore, using Microsoft Excel, we sorted the dataset in ascending order, from lowest CPU utilization and memory usage to highest. The primary objective behind this ranking is to establish a reliable benchmark, serving as the ground truth for ML (Machine Learning) and DL (Deep Learning) ranking methods.

7.3.2 The selected implementation platform

We implement our experiments on Google Collaboratory, which is a cloud-based Jupyter notebook environment provided by Google. It allows users to write and execute Python code directly in their web browsers without the need for any local installations [94].

We chose Google Collaboratory to implement our experiments due to its ease of use and accessibility. Google Collaboratory provides free GPU or TPU (Tensor Processing Unit) runtime, which can help to implement machine learning tasks much faster [94]. It also provides an interactive and dynamic environment for executing code and documenting the experiment steps. The integration of Google

Collaboratory with Google Drive enables our notebooks and datasets to be stored and accessed, it simplifies data management and ensures that our work is backed up and accessible from anywhere [94].

Google Collaboratory provides a wide range of pre-installed libraries and packages that provide powerful tools and functions for machine learning and data analysis tasks, such as sklearn library, which facilitates the implementation of our ML and DL experiments [94].

7.3.3 The experimental set up and implementation

To evaluate and validate the proposed framework of the iVFC-TNv selection module, we implement two statistical methods, one machine learning (ML) method, and four deep learning (DL) methods. The following sections provide a detailed explanation of the implementation of the three methods used to develop the TNv selection module and analyse the evaluation metrics used to evaluate and validate each method.

Statistical methods: Multi-Criteria Decision-Making (MCDM) methods

Multi-criteria decision-making (MCDM) methods have been widely used in cloud and fog computing systems to solve the problem of service provider selection in the case of multi-criteria decision making [95]. In our proposed TNv selection module, two MCDM methods are used to rank the list of the TNvs that agreed to participate. The purpose of ranking the TNvs is to help the overloaded SNv to find the most optimal TNv that can handle the task effectively. Choosing the most optimal TNv from the available TNvs is based on multiple criteria (CPU utilization percentage and memory used) of the TNv, which is considered a MCDM problem. Therefore, MCDM methods are used in our proposed framework to select the most optimal TNv from the available alternatives to handle the offloaded task.

Of the many effective MCDM methods that have been used by researchers for fog service provider selection, we use two, namely the analytic hierarchy process (AHP) method to calculate the weights of the criteria of the TNvs and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method for ranking the different TNvs based on their criteria.

AHP is a popular multi-criteria decision-making method that helps to make complex decisions. It can be used to solve a wide range of problems and it provides a structured approach to decision making. It allows decision makers to make decisions based on multiple criteria and to balance conflicting objectives [96].

The TOPSIS method is based on the concept that the best alternative should have the shortest distance, namely Euclidean distance, from the ideal solution [97]. Since TOPSIS ranks alternatives according to how closely they resemble the worst and best solution, it makes it easier and more logical to compare alternatives. This is why we chose it over other MCDM approaches. TOPSIS is straightforward, easily comprehensible, widely favoured, and has a low level of computational complexity. As there is no restriction on the number of possibilities and criteria, it also offers more consistency [98].

To implement the TOPSIS method to find the best alternative to handle the offloaded task, we need to have the weights of all the criteria of the TNvs and then use these weights to rank the different TNvs based on the QoS parameters (CPU utilization and memory usage). This ranking is based on the relative closeness of a particular alternative to the ideal solution.

We assume that the weights of the criteria can be chosen according to user preferences, where each weight must be a decimal value between 0 and 1 and the sum of all the criteria weights must be 1. Therefore, we use AHP to calculate the weights of the TNvs criteria, namely CPU utilization and memory usage.

A) Analytic Hierarchy Process (AHP)

Table 7.1 shows our decision matrix with the two criteria that we use for the AHP to calculate the weights of the criteria, namely CPU utilization and memory usage of 100 nodes where CPU utilization and memory usage are obtained from the same time interval of the prediction values resulting from the solution to Objective 1. Each node has its own criteria values associated with it. The following steps are undertaken to calculate the weights of the

Table 7.1 : AHP decision matrix

Node-ID	CPU-Utilization	Memory-Usage
TN _v -1	31.2	0.1
TN _v -2	30.6	19.5
TN _v -3	73.5	91.7
TN _v -4	67.1	81.4
TN _v -5	57.2	57.7
TN _v -6	0.0	8.8
TN _v -7	0.0	12.7
TN _v -8	47.5	50.4
TN _v -9	0.0	63.5
TN _v -10	0.0	12.1
—	—	—
—	—	—
—	—	—
TN _v -100	—	—

criteria of the TN_vs, namely CPU utilization and memory usage, using AHP, based on [96]:

Step 1: Develop a hierarchical structure with a goal at the top level, the attributes/criteria at the second level and the alternatives (Node-IDs) at the third level, as shown in Figure 7.2.

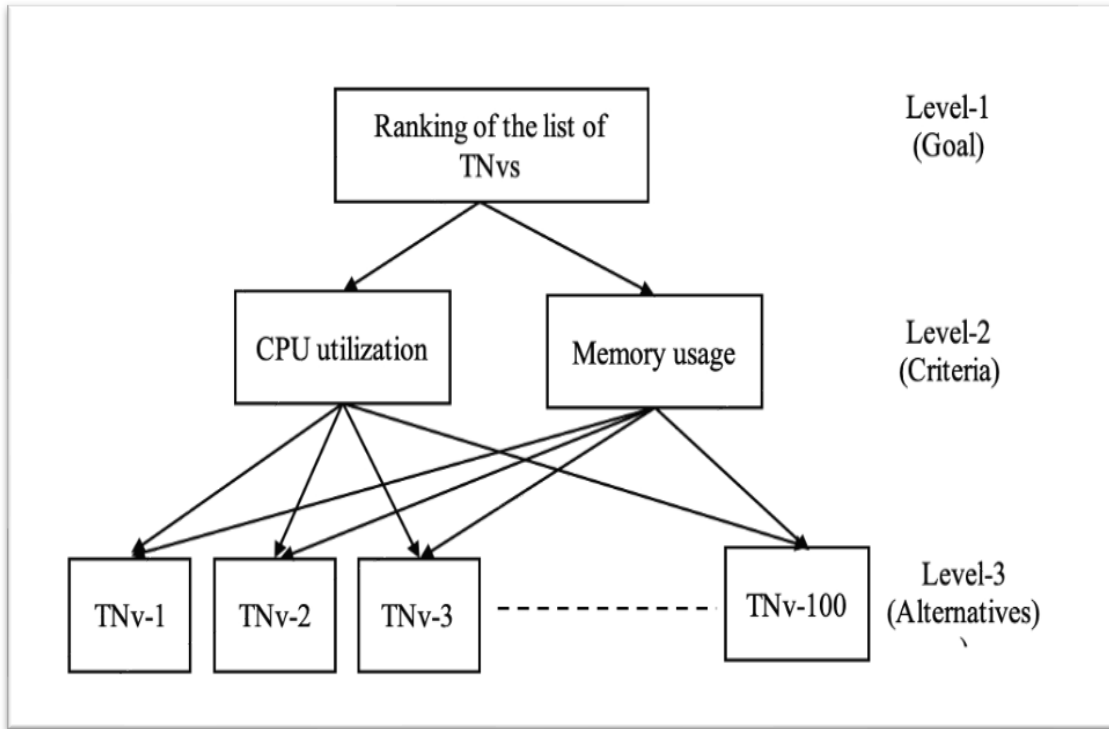


Figure 7.2 : Hierarchical structure for AHP method

Step 2: Create a pair-wise comparison matrix to determine the relative importance of the different attributes/criteria with respect to the goal.

To create this matrix, we need to determine the importance of each criterion, with respect to the other criteria, to achieve the goal, for example, how important is the CPU utilization criterion in ranking the TNvs. To determine the importance of each attribute, we used the following scale for relative importance: for AHP, we use a scale from 1-9 to represent the weighted percentage where 1 indicates equal importance when two activities contribute equally to the objective. Table 7.2 details all the weighted percentages of the scale.

Table 7.2 : Weighted percentage of the fundamental scale used for AHP method [96]

Intensity of importance on an absolute scale	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the objective
3	Moderate Importance	Experience and judgment slightly favour one activity over another
5	Essential or strong importance	Experience and judgment strongly favour one activity over another
7	Very strong importance	An activity is strongly favoured and its dominance is demonstrated in practice
9	Extreme importance	The evidence favouring one activity over another is of the highest possible order of affirmation
2,4,6,8	Intermediate values between the two adjacent judgments	When compromise is needed
1/2, 1/3, 1/4, 1/5, 1/7, 1/9 Values for inverse comparison		

The length of the pairwise matrix is equivalent to the number of criteria used in the decision-making process. As there are two criteria, namely CPU utilization and memory usage, we will have a 2x2 matrix. The values in the pairwise matrix depend on the decision maker or the node that has a task which needs to be offloaded to another node for execution.

Step 3: Identify the relative importance of each criterion in the decision-making process for ranking the TNvs list.

As we assumed that the weights of the criteria are assigned based on the preferences of the decision maker (i.e., the overloaded SNv), there will be different scenarios where the overloaded SNvs identify the relative importance of each criterion and with each scenario, hence the resulting weights of the

criteria using AHP will be different. We use one scenario in which the SNv with a computation task to be offloaded to another node might consider the “CPU utilization” and “memory usage” criteria as having almost equal to moderate importance when selecting the optimal TNv for task execution. This SNv considers that the importance of the “CPU utilization” compared to “memory usage” criteria falls between “equally important” and “moderately important”. Therefore, we assigned it a value of 2. “Memory usage” compared to “CPU utilization” = $1/2$, which is the value for the inverse comparison as shown in Table 7.2. The value 1 in Table 7.3 indicates a comparison between the same criteria, which is equal importance (i.e., CPU utilization compared to CPU utilization = 1 and memory usage compared to memory usage = 1).

Based on this scenario and the fundamental scale in Table 7.2, we have the following pair-wise matrix shown in Table 7.3:

Table 7.3 : The pair-wise matrix for AHP method

	CPU utilization	Memory usage
CPU utilization	1	2
Memory usage	$1/2$	1

Step 4: Convert the fractional values into decimal values and calculate the sum of each column, as shown in Table 7.4.

Table 7.4 : Sum of the pair-wise matrix for AHP method

	CPU utilization	Memory usage
CPU utilization	1	2
Memory usage	0.5	1
Sum	1.5	3

Then, the normalized pair-wise matrix is calculated by dividing each value in each column by the sum of the values of that column. This gives the following normalized pair-wise matrix shown in Table 7.5

Table 7.5 : The normalized pair-wise matrix for AHP method

	CPU utilization	Memory usage
CPU utilization	0.667	0.667
Memory usage	0.333	0.333

Step 5: Calculate the criteria weight by averaging all of the elements in each row, as shown in Table 7.6.

Table 7.6 : Criteria weights using AHP method

	CPU utilization	Memory usage	Criteria weight
CPU utilization	0.667	0.667	0.667
Memory usage	0.333	0.333	0.333

Step 6: Calculate the consistency to check whether the calculated criteria weights are correct and can be used for decision making or not.

To calculate the consistency, the first step is to take the same comparison matrix, which is not normalized, and multiply each value in the column with the criteria weight, as shown in Table 7.7.

Table 7.7 : Consistency calculations for AHP method, step 1

Criteria weight	0.667	0.333
	CPU utilization	Memory usage
CPU utilization	$1 * 0.667$	$2 * 0.667$
Memory usage	$1/2 * 0.333$	$1 * 0.333$

Then, the second step to calculate the consistency is to obtain the weighted sum matrix by to calculating the sum of each row, as shown in Table 7.8.

Table 7.8 : Consistency calculations for AHP method, step 2

	CPU utilization	Memory usage	Weighted sum value
CPU utilization	0.667	0.666	1.333
Memory usage	0.334	0.333	0.667

Then the third step is to calculate the ratio of each weighted sum value and criteria weight using the following formula:

$$Ratio = \frac{weighted\ sum\ value}{criteria\ weight} \quad (7.1)$$

As shown in Table 7.9.

The fourth step is to calculate λ_{max} average using the following formula [96]:

$$\lambda_{max} average = ratio / number\ of\ criteria \quad (7.2)$$

$$=(1.999 + 2.003)/2 = 2.001$$

Table 7.9 : Consistency calculations for AHP method, step 3

	CPU utilization	Memory usage	Weighted sum value	Criteria weight	Ratio
CPU utilization	0.667	0.666	1.333	0.667	1.999
Memory usage	0.334	0.333	0.667	0.333	2.003

The fifth step is to calculate the consistency index (C.I) using the following formula [96]:

$$Consistency\ Index(C.I) = (max - n)/(n - 1) \quad (7.3)$$

where n is the number of criteria.

$$C.I = 2.001-2/2-1= 0.001$$

Finally, the last step is to calculate the consistency ratio using the following formula [96]:

$$Consistency\ Ratio = Consistency\ Index(C.I)/RI \quad (7.4)$$

where RI is the random index. Random index is the consistency index of a randomly generated pair-wise matrix. The random index is calculated based on the number of criteria being compared [96].

Table 7.10 shows a pre-determined random index for up to 10 criteria.

Table 7.10 : The random index of a randomly generated pair-wise matrix [96]

n	1	2	3	4	5	6	7	8	9	10
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

In our dataset, the number of criteria =2 so n=2. Based on Table 7.10, for a 2-criteria matrix, the random index is 0, which means that the pairwise

comparison matrix is consistent and the CR is not applicable. Therefore, we do not need to calculate the CR for a 2-criteria matrix since the RI is zero and the matrix is consistent.

Based on the previous scenario to identify the weights of the TNv criterion using the AHP method, the criteria weights that we use for the implementation of the TOPSIS method to rank the list of TNvs are 0.667 for CPU utilization and 0.333 for memory usage.

The next section explains how these weights are used to rank the list of TNvs using the TOPSIS method.

B) Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method

We implement TOPSIS method through the following steps:

We assume that there are n fog nodes and each fog node has m criteria. This results in a m x n matrix called the evaluation matrix, let's say matrix p. Evaluation matrix p is normalized to form matrix X.

Step 1: Identify the criteria that will be used to evaluate the alternatives.

We evaluate the alternatives (TNvs) using two criteria (CPU utilization and memory usage), which are considered to be quantitative criteria.

Step 2: Normalize the decision matrix: Convert the raw data for each criterion into a dimensionless scale, such as a scale from 0 to 1, to eliminate the units of measurement and make the criteria comparable. To do so, we calculate the weighted normalized decision matrix using the following formula to obtain the performance value of each cell [97].

$$\underline{X} = \frac{X_{ij}}{\sqrt{\sum_{j=1}^n X_{ij}^2}}, \text{ where } \sum_{j=1}^n X_{ij}^2 = 1 \quad (7.5)$$

Step 3: Determine the weighted normalized decision matrix. Assign the criteria weights that were identified using AHP in section A to each criterion to reflect its relative importance. Then, multiply each normalized value by its corresponding weight. The weights of the criteria according to AHP are 0.667 for CPU utilization and 0.333 for memory usage.

Step 4: Calculate the ideal best and the ideal worst value (ideal alternative (AI) and the non-ideal alternative (AN)) for every QoS parameter.

The ideal best solution is the maximum value for a beneficial criterion and the minimum value for a non-beneficial criterion. The ideal worst solution is the minimum value for a beneficial criterion and the maximum value for a non-beneficial criterion. Beneficial criteria are the evaluation criteria that represent the desirable characteristics of the alternatives where the higher the value of the beneficial criteria, the better the performance of the alternatives. Non-beneficial criteria are the evaluation criteria that represent undesirable or negative characteristics of the alternatives. In this case, the lower the value of the non-beneficial criteria, the better the performance of the alternatives [98]. Our criteria (CPU utilization and memory usage) are considered non-beneficial criteria, hence a lower value is preferred. This means the lower the value of CPU utilization and memory usage, the better the performance of the TNv.

Step 5: Calculate the Euclidean distance from the ideal best and the ideal worst value to obtain the separation measures of each alternative from the ideal best using the following formulas [98]:

$$S_i^+ = \sqrt{\sum_{j=1}^n (V_{ij} - V_j^+)^2} \quad (7.6)$$

$$S_i^- = \sqrt{\sum_{j=1}^n (V_{ij} - V_j^-)^2} \quad (7.7)$$

Step 6: Calculate the relative closeness to the ideal best solution.

Calculate the relative closeness of each alternative to the ideal best solution by dividing the distance to the ideal worst solution by the sum of the distances to the ideal best and the ideal worst solutions. That is, calculate the performance score using the following formula[98]:

$$p^i = \frac{S_i^-}{S_i^+ + S_i^-} \quad (7.8)$$

Step 7: Rank the alternatives: we rank the TNvs based on their relative closeness to the ideal solution, where the TNv with the highest relative closeness is the best TNv. Based on the performance score from step 6, we rank our alternatives from 1-n to select the best one which is ranked 1 in the list.

Using the same weights of the criteria that we generated using AHP, which was explained in section 7.3.3, we implement TOPSIS using R (RStudio 2022.02.3+492) to rank the TNvs based on their criteria.

To implement TOPSIS, we define the criteria and their corresponding weights that will be used in the TOPSIS method, then, we use 'topsis' function to rank the TNvs based on the defined criteria.

The code of implementing the TOPSIS method using R can be accessed using the following link: <https://github.com/alhamedy/TOPSIS-R-code/tree/main>

Machine Learning (ML) method

Of the ML methods, we apply a regression XGBoost (Extreme Gradient Boosting) as a second approach to rank the list of the TNvs based on their CPU utilization and memory usage criteria. The purpose of implementing XGBoost method is to compare the resulting rankings from the XGBoost method with those obtained from the TOPSIS method.

XGBoost method is a popular and efficient open-source implementation of the gradient boosted trees algorithm to accurately predict the rank variable based on multiple criteria (i.e., CPU utilization and memory usage in our proposed framework) [85].

We choose the XGBoost method to rank the TNVs for the following reasons: [85]:

- Given that the prediction of the rank variable of the TNVs relies on multiple criteria, specifically CPU utilization and memory usage, XGBoost is regarded as a fitting approach to predict the ranking of the TNVs.
- XGBoost is able to handle different types of data, which makes it a very flexible method to rank TNVs based on multiple criteria.
- It is highly accurate and reliable in ranking fog service providers based on their criteria.
- It is robust against overfitting as it uses regularization techniques to prevent overfitting that occurs when the model performs well on the training data but performs poorly when testing new data

These reasons make XGBoost a good choice for ranking TNVs based on their criteria.

Using Google Collaboratory, we conduct a number of experiments using a regression XGBoost method to predict the rank variable for the list of TNVs based on their CPU utilization and memory usage variables using the following libraries [85]:

- Pandas: which is a powerful data manipulation and analysis library in Python.
- The scikit-learn library (also known as sklearn) to split our dataset into training and testing subsets.
- XGBRegressor class from the XGBoost library, which is an optimized gradient boosting algorithm used for regression tasks.

- XGBoost to import the entire XGBoost module to be able to access all the functionality provided by the XGBoost library.

The experiments conducted using varying number of estimators (trees) while using learning rate =0.1 for all experiments. Then the performance of the model in each experiment is evaluated based on the Mean Absolute Error (MAE) metric on the training, testing and the top-30 testing datasets.

n_estimator represents the number of estimators or decision trees used in the gradient boosting ensemble in each experiment [85]. The experiment includes testing five different values for estimators: 100, 200, 300, 400, and 500.

Learning rate is a hyperparameter used to control the step size at each iteration during the gradient boosting process [85]. We use this parameter to prevent overfitting of the model. In all experiments we use a learning rate of 0.1.

The code of implementing the XGBoost method to rank the list of TNvs can be accessed using the following link: <https://github.com/alhamedy/ML-model-for-ranking-service-providers-/blob/main/XGBoost.ipynb>

Algorithm 7.1 shows the steps of the XGBoost method used to rank the list of TNvs.

Deep Learning (DL) Method

We implement deep neural networks (DNNs) as a third method to compare the result with the TOPSIS and XGBoost methods. The purpose of using DNNs is to predict the rank variable for the list of TNvs based on their workload variables (CPU utilization and memory usage).

We choose deep neural networks (DNN) to rank the TNvs based on their workload for the following reasons: [78]:

Algorithm 7.1 XGBoost method to rank TNvs

Require: The list of TNvs

Ensure: The ranked list of TNvs

- 1: Step 1: Load the data (the list of TNvs)
 - 2: Step 2: Split the data into training, testing, and top-30 sets
 - 3: Step 3: Define the XGBoost model hyperparameters (number of estimators and learning rate)
 - 4: Step 4: Train the model on the training data using the XGBoost algorithm:
 - 5: a. Initialize the model
 - 6: b. Iterate over the specified number of trees:
 - 7: i. Compute the gradient of the loss function
 - 8: ii. Construct a decision tree that minimizes the loss function using the gradient
 - 9: iii. Add the decision tree to the ensemble of trees in the model
 - 10: Step 5: Compute the final predictions for the ranking of the training data by summing the predictions of all the trees in the ensemble
 - 11: Step 6: Evaluate the model on the testing data by computing MAE
 - 12: Step 7: Evaluate the model on the top-30 testing data by computing MAE
-

- DNN can capture the relationships in the data used to rank the TNvs to determine whether it is able to deal with the different workload criteria of the TNvs.
- DNN is able to learn from large datasets which is important for ranking TNvs based on their workload because there may be a large number of nodes and a large amount of workload data.

In Google Collaboratory, we train a neural network model using TensorFlow and Keras by importing the following libraries:

- Numpy for numerical computations.
- Pandas for data manipulation and analysis.
- Tensorflow to build and train neural networks.
- Matplotlib.pyplot for data visualization.
- Various modules and functions from tensorflow.keras for building and configuring the neural network model.

The number of nodes (neurons) in each hidden layers determine using the `kernel_initializer='normal'`, which means that the weights of the hidden layers are randomly sampled from a Gaussian distribution.

We use the mean absolute error as the loss function and the Adam optimizer with a learning rate of 0.0001 to compile the model. The model's performance is evaluated on the training, validation, and testing datasets using the 'evaluate' method based on the mean absolute error (MAE) metric. We split the dataset into 70% for training the model, 15% for evaluating the model and 15% for testing using the DataFrame 'decision'.

We implement DNNs with one, three, four and five hidden layers. These neural networks consist of one input layer, one, three, four and five hidden layers and one output layer. To enable the model to capture non-linear relationships in datasets, we increase the number of hidden layers to increase the model's ability to learn complex patterns and relationships. In a DNN, each hidden layer learns and extracts features from the previous layer. A model with more hidden layers can learn hierarchical representations of input data, which helps to improve its performance. Also, increasing the hidden layers helps to capture the underfitting or overfitting of the model on the dataset.

To avoid overfitting of the model, we use the early stopping concept to prevent the model from continuing to train on the training set when its performance on a validation set starts to deteriorate, which prevents the model from over-optimizing the training data [99]. For this purpose, we use 'callbacks=[cp, logger]' where 'cp' callback is a model checkpoint object to save the best model during training based on the mean absolute error value and 'logger' callback is a CSVLogger object that saves the training history to a CSV file in the path we specified.

We employ a different number of epochs while using the same learning rate for all of the DNN experiments. Epochs are the number of times the model is trained on the training dataset and it has a significant impact on the performance of the model. At each epoch, the model slowly improves its performance as it optimizes its parameters based on the training data [99].

The purpose of training our selection model with different numbers of epochs is to find the best number of epochs to train the model to avoid underfitting or overfitting of the model. Underfitting of the model occurs when the epoch count is too low. In this case, the model may not have enough time to learn all the patterns in the data. On the other hand, overfitting of the model occurs when the epoch

count is too high. In this case, the model starts to memorize the training data instead of generalizing to new data [99].

The learning rate is a hyperparameter that determines the step size for updating the model's parameters during each iteration [99]. In our DNN experiments, using a small learning rate helps our model converge more accurately. This means that the training process of our model has reached a point of stability and the model has achieved a satisfactory level of optimization and has learned to generalize well on the training data.

Tables 7.11 to 7.14 show the implementation of DNNs with 1, 3, 4, and 5 hidden layers respectively, using different numbers of epochs. These tables also show the mean absolute error (MAE) for the training, validation and testing dataset and also the MAE for the top 30 nodes.

It is clear from the these tables that increasing the number of epochs improves the performance of the model in all experiments with a different number of hidden layers.

A) Artificial Neural Network (one hidden layer)

We implement an artificial neural network (ANN) with one hidden layer on different datasets (training, validation, testing, and top-30). We add the first hidden layer to the model, which is a dense layer with 256 nodes (neurons), using the ReLU activation function. This layer receives the input data and processes it through 256 nodes.

We use a different number of epochs and the same learning rate for all experiments. As the training progresses, we observe changes in the MAE values for the training, validation, testing and top-30 datasets, as shown in Table 7.11.

Initially, at epoch 100, the model shows relatively higher MAE values for all datasets. However, as the training progresses, particularly beyond epoch 1000,

the MAE values start to decrease, indicating an improvement in the model’s performance.

By epoch 3000, the model has achieved its best performance, as we can see that MAE is lower across all datasets. This suggests that the model has learned to make more accurate predictions. Therefore, we choose this experiment (using epoch 3000) as the best experiment in the implementation of ANN with one hidden layer.

The MAE values for the testing dataset and the top-30 subset provides additional insights. The decreasing trend of the MAE values indicates that the model’s performance improves as it learns to generalize better on unseen data and particularly on the top-30 instances.

The code of the ANN with one hidden layer can be accessed using the following link: https://github.com/alhamedy/DL-models-for-ranking-the-different-service-providers/blob/main/Neural_Network_1_layers.ipynb

Table 7.11 : Experiments of implementing artificial neural network to rank the list of TNvs

Epoch count	Learning rate	MAE training	MAE validation	MAE testing	MAE top-30
100	0.0001	58.850	62.887	58.958	11.417
1000	0.0001	15.939	15.311	15.796	5.383
1500	0.0001	15.326	14.096	14.742	3.207
2000	0.0001	13.764	13.594	14.332	1.557
3000	0.0001	11.958	11.954	12.511	1.489

B) Deep neural networks with three hidden layers

Based on the result of previous experiments of the ANN with one hidden layer, we observed relatively high MAE values for both the validation and testing

dataset. These high MAE values suggest underfitting of the model. To address this issue, we increase the number of DNN hidden layers to three hidden layers to add additional complexity and capacity to the model, allowing it to learn more complex patterns and representations from the input data.

In addition to the first hidden layer, which is a dense layer with 256 nodes (neurons), we add a second hidden layer with 128 nodes and a third hidden layer with 96 nodes.

We use a different number of epochs and the same learning rate for all experiments. As the training progresses, we observe changes in the MAE values for the training, validation, testing and top-30 datasets.

As shown in Table 7.12, at epoch 100, the model has relatively higher MAE values for all datasets, indicating its initial performance. However, as the training continues, particularly beyond epoch 500, the MAE values start to decrease, suggesting an improvement in the model's accuracy. The decreasing trend continues until epoch 2000, indicating that the model is progressively learning and making better predictions.

By epoch 2000, the model has achieved its best performance, as the MAE values are low across all datasets. This indicates that the model has effectively learned from the training data and is capable of making accurate predictions. Therefore, we choose this experiment (using epoch 2000) as the best experiment in the implementation of DNNs with three hidden layers.

The code of the DNNs with three hidden layers can be accessed using the following link: https://github.com/alhamedy/DL-models-for-ranking-the-different-service-providers/blob/main/Neural_Network_3_layers.ipynb

C) Deep neural networks with four hidden layers

Table 7.12 : Experiments of implementing deep neural networks with three hidden layers to rank the list of TNvs

Epoch count	Learning rate	MAE training	MAE validation	MAE testing	MAE top-30
100	0.0001	46.388	50.266	46.784	10.565
500	0.0001	13.680	13.922	14.029	1.513
600	0.0001	13.136	13.804	14.520	1.361
1000	0.0001	12.582	13.870	14.426	1.129
2000	0.0001	10.067	10.860	10.925	1.337

Based on the result from the previous experiments of the DNN with three hidden layers, MAE values continue to exhibit higher levels for both the validation and testing datasets. This indicates that the model’s performance is still not achieving the desired level of accuracy. Therefore, we increase the number of DNN hidden layers to four hidden layers.

In addition to the first hidden layer, which is a dense layer with 256 nodes (neurons) , the second hidden layer with 128 nodes and the third hidden layer with 96 nodes, we add one more layer with 72 nodes to allow it to capture the complex patterns and representations from the input data.

We use different numbers of epochs and the same learning rate for all experiments. As the training progresses, we observe changes in the MAE values for the training, validation, testing and top-30 datasets, as shown in Table 7.13.

As in the previous experiment (DNN with three layers), in this experiment, the model initially exhibits higher MAE values for all datasets at epoch 100, indicating its initial performance. However, as the training continues, specifically beyond epoch 500, the MAE values start to decrease, indicating an improvement in the model’s accuracy.

By epoch 2000, the model has achieved its best performance, as evidenced by the lowest MAE values across all datasets. Therefore, we choose this experiment (using epoch 2000) as the best experiment in the implementation of DNNs with four hidden layers.

The code of the DNNs with four hidden layers can be accessed using the following link: https://github.com/alhamedy/DL-models-for-ranking-the-different-service-providers/blob/main/Neural_Network_4_layers.ipynb

Table 7.13 : Experiments of implementing deep neural networks with four hidden layers to rank the list of TNvs

Epoch count	Learning rate	MAE training	MAE validation	MAE testing	MAE top-30
100	0.0001	38.492	43.003	40.700	6.604
500	0.0001	13.554	13.966	14.266	1.902
1000	0.0001	12.211	13.676	13.370	1.002
2000	0.0001	9.688	10.693	10.770	0.826

D) Deep neural networks with five hidden layers

To see if we can further improve the model’s performance, we increase the DNN hidden layers to five hidden layers by adding one more layer with 64 nodes.

We use different numbers of epochs and the same learning rate for all experiments. As the training progresses, we observe changes in the MAE values for the training, validation, testing and top-30 datasets, as shown in Table 7.14.

As in the previous two experiments (DNN with three layers and DNN with four layers), in this experiment, the model initially exhibits higher MAE values for all datasets at epoch 100, indicating its initial performance. However, as

the training continues, specifically beyond epoch 500, the MAE values start to decrease, indicating an improvement in the model's accuracy.

By epoch 2000, the model has achieved its best performance, as indicated by the lowest MAE values across all datasets. Therefore, we choose this experiment ((using epoch 2000) as the best experiment in the implementation of DNNs with five hidden layers.

The code of the DNNs with five hidden layers can be accessed using the following link: https://github.com/alhamedy/DL-models-for-ranking-the-different-service-providers/blob/main/Neural_Network_5_layers.ipynb

Table 7.14 : Experiments of implementing deep neural networks with five hidden layers to rank the list of TNvs

Epoch count	Learning rate	MAE training	MAE validation	MAE testing	MAE top-30
100	0.0001	38.836	43.082	40.214	7.748
500	0.0001	13.913	14.992	14.803	1.447
600	0.0001	14.039	13.946	14.293	1.290
1000	0.0001	11.163	12.015	12.371	1.057
2000	0.0001	7.990	8.507	8.605	0.859

7.3.4 Evaluation metrics

To evaluate and validate the efficiency of the chosen method in ranking the list of TNvs to choose the most optimal TNv to carry out the task, we compare the experiment results of the three methods: TOPSIS, XGBoost and DNNs, using MAE metric.

MAE is the absolute value of the difference between the predicted value and the actual value. MAE measures the accuracy of the continuous variables and gives an

indication of how large an error can be expected from the prediction on average.

We calculate MAE using the following formula [83]:

$$MAE = \frac{\sum_{i=1}^n (|y_i - x_i|)}{n} \quad (7.9)$$

where y_i is the predicted value (generated after applying ranking method), x_i is the actual value (actual ranking) and n is the total number of data points (total number of TNvs).

7.4 Results and discussion

7.4.1 Evaluation of the TOPSIS method

To evaluate the working of the TNv selection module using the TOPSIS method, we measure the accuracy of the TOPSIS based on MAE metric. Table 7.15 shows the resulting ranked list of the top-30 nodes using R according to the TOPSIS method compared to the actual ranking of the dataset.

Figure 7.3 shows a snapshot of the Excel calculations of MAE for the top-30 nodes resulting from TOPSIS ranking, showing that $MAE = 0.963$, which means that on average, the difference between the predicted ranking using the TOPSIS method and the actual ranking of the TNvs is 0.963. This indicates that the TOPSIS model is able to accurately predict the ranking of the TNvs based on their CPU utilization and memory usage criteria.

To evaluate the performance of the TOPSIS method compared to the other methods (XGBoost and DNNs) in ranking the list of the TNvs, we compare the MAE of the TOPSIS method with the performance of XGBoost and the DNN methods in section 7.4.4.

Table 7.15 : The top-30 ranked TNvs according to the TOPSIS method compared to the actual ranking

Actual Ranking				TOPSIS Ranking		
Rank	Node-ID	CPU utilization	Memory usage	Node-ID	CPU utilization	Memory usage
1	N30	0	0.18997	N30	0	0.18997
2	N91	0	0.36244	N91	0	0.36244
3	N39	0.00030311	1.0398	N39	0.00030311	1.0398
4	N29	0.0024722	1.5068	N34	0.047836	1.3042
5	N87	0	2.7112	N29	0.0024722	1.5068
6	N34	0.047836	1.3042	N32	0.68068	1.1225
7	N38	0.057987	2.2128	N84	0.17597	2.0505
8	N27	0.13162	2.2212	N800	0.489072653	1.956962869
9	N84	0.17597	2.0505	N38	0.057987	2.2128
10	N86	0.031147	2.8852	N98	1.0843	0.21381
11	N800	0.489072653	1.956962869	N27	0.13162	2.2212
12	N32	0.68068	1.1225	N28	1.0463	0.67013
13	N98	1.0843	0.21381	N99	1.2519	0.47938
14	N37	0.032575	4.8549	N97	1.2484	0.57431
15	N85	0.0000575	5.7733	N100	1.2824	0.59744
16	N28	1.0463	0.67013	N87	0	2.7112
17	N16	0	7.464	N86	0.031147	2.8852
18	N97	1.2484	0.57431	N31	0.71825	2.515
19	N99	1.2519	0.47938	N96	1.5179	0.9131
20	N100	1.2824	0.59744	N61	0.58683	3.5632
21	N31	0.71825	2.515	N33	0.72852	3.9053
22	N14	0	8.125	N37	0.032575	4.8549
23	N61	0.58683	3.5632	N92	2.1339	3.0525
24	N6	0	8.8201	N85	0.0000575	5.7733
25	N96	1.5179	0.9131	N94	2.8939	0.050839
26	N17	0.013336	7.9397	N89	0.57999	6.2918
27	N33	0.72852	3.9053	N88	1.3109	6.4288
28	N82	0.43754	7.2342	N81	0.8263	6.9557
29	N89	0.57999	6.2918	N48	1.994	5.8942
30	N62	0.00057464	9.1825	N82	0.43754	7.2342

Actual Ranking		TOPSIS Ranking		$y_i - x_i$		$ y_i - x_i $	
CPU utilization	Memory usage	CPU utilization	Memory usage	CPU utilization	Memory usage	CPU utilization	Memory usage
0	0.18997	0	0.18997	0	0	0	0
0	0.36244	0	0.36244	0	0	0	0
0.00030311	1.0398	0.00030311	1.0398	0	0	0	0
0.0024722	1.5068	0.047836	1.3042	-0.0453638	0.2026	0.0453638	0.2026
0	2.7112	0.0024722	1.5068	-0.0024722	1.2044	0.0024722	1.2044
0.047836	1.3042	0.68068	1.1225	-0.632844	0.1817	0.632844	0.1817
0.057987	2.2128	0.17597	2.0505	-0.117983	0.1623	0.117983	0.1623
0.13162	2.2212	0.489072653	1.956962869	-0.357452653	0.264237131	0.357452653	0.264237131
0.17597	2.0505	0.057987	2.2128	0.117983	-0.1623	0.117983	0.1623
0.031147	2.8852	1.0843	0.21381	-1.053153	2.67139	1.053153	2.67139
0.489072653	1.956962869	0.13162	2.2212	0.357452653	-0.264237131	0.357452653	0.264237131
0.68068	1.1225	1.0463	0.67013	-0.36562	0.45237	0.36562	0.45237
1.0843	0.21381	1.2519	0.47938	-0.1676	-0.26557	0.1676	0.26557
0.032575	4.8549	1.2484	0.57431	-1.215825	4.28059	1.215825	4.28059
0.0000575	5.7733	1.2824	0.59744	-1.2823425	5.17586	1.2823425	5.17586
1.0463	0.67013	0	2.7112	1.0463	-2.04107	1.0463	2.04107
0	7.464	0.031147	2.8852	-0.031147	4.5788	0.031147	4.5788
1.2484	0.57431	0.71825	2.515	0.53015	-1.94069	0.53015	1.94069
1.2519	0.47938	1.5179	0.9131	-0.266	-0.43372	0.266	0.43372
1.2824	0.59744	0.58683	3.5632	0.69557	-2.96576	0.69557	2.96576
0.71825	2.515	0.72852	3.9053	-0.01027	-1.3903	0.01027	1.3903
0	8.125	0.032575	4.8549	-0.032575	3.2701	0.032575	3.2701
0.58683	3.5632	2.1339	3.0525	-1.54707	0.5107	1.54707	0.5107
0	8.8201	0.0000575	5.7733	-0.0000575	3.0468	0.0000575	3.0468
1.5179	0.9131	2.8939	0.050839	-1.376	0.862261	1.376	0.862261
0.013336	7.9397	0.57999	6.2918	-0.566654	1.6479	0.566654	1.6479
0.72852	3.9053	1.3109	6.4288	-0.58238	-2.5235	0.58238	2.5235
0.43754	7.2342	0.8263	6.9557	-0.38876	0.2785	0.38876	0.2785
0.57999	6.2918	1.994	5.8942	-1.41401	0.3976	1.41401	0.3976
0.00057464	9.1825	0.43754	7.2342	-0.43696536	1.9483	0.43696536	1.9483
Sum $y_i - x_i$						14.64000067	43.12355526
MAE						0.488000022	1.437451842
						0.962725932	

Figure 7.3 : Snapshot of the accuracy calculations of TOPSIS method using Excel

7.4.2 Evaluation of the XGBoost method

Table 7.16 summarizes the results of the experiments that were conducted using the XGBoost method to predict the rank variable based on the CPU utilization and the memory usage criteria of the TNVs. The experiments were conducted by varying the number of estimators and learning rates, and the performance of the XGBoost model was evaluated using the MAE metric for the training, testing and the top-30 nodes data.

Table 7.16 : Experiments using XGBoost for ranking the list of the TNvs

Experiment	n_estimators	Learning rate	MAE training	MAE testing	MAE test-top-30
1	100	0.1	13.962	18.749	6.937
2	200	0.1	10.562	14.945	8.844
3	300	0.1	8.726	13.477	8.618
4	400	0.1	7.594	12.714	8.585
5	500	0.1	6.839	12.408	7.843

The results show that the first experiment with n estimators = 100 and learning rate = 0.1 achieved the best MAE (MAE= 6.937 for the ranking of the top-30 nodes). The MAE started to increase for the top-30 testing data when we increased the number of estimators (number of runs that XGBoost will try to learn). This indicates that the model may have overfitted the data, which leads us to conclude that increasing the number of estimators beyond a certain point might not necessarily improve XGBoost model performance.

Figures 7.4 and 7.5 depict the predicted ranking of the top-30 nodes using the XGBoost method with the optimal parameters (parameters of the best performing experiment) compared to the actual ranking. The figures show how the ranking results of the XGBoost model are near to the optimal ranking compared to the actual ranking. This indicates that the XGBoost model is able to accurately predict the ranking of the TNvs based on their CPU utilization and memory usage criteria.

In conclusion, the XGBoost method is an effective approach for predicting the ranking of the TNvs. The experiment using the XGBoost method with n_estimators = 100 and learning rate = 0.1 achieved the best performance in terms of the MAE metric, and the ranking result of the XGBoost model was very close to the optimal ranking. Therefore, we choose this experiment as the best experiment in the implementation of XGBoost.

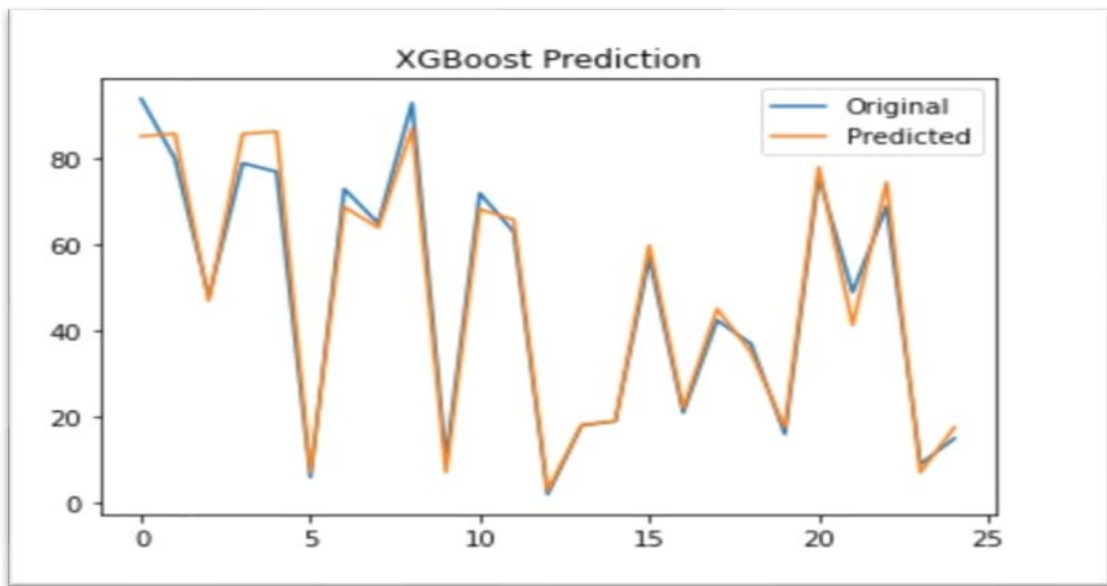


Figure 7.4 : XGBoost method ranking compared to the actual ranking of the top-30 nodes

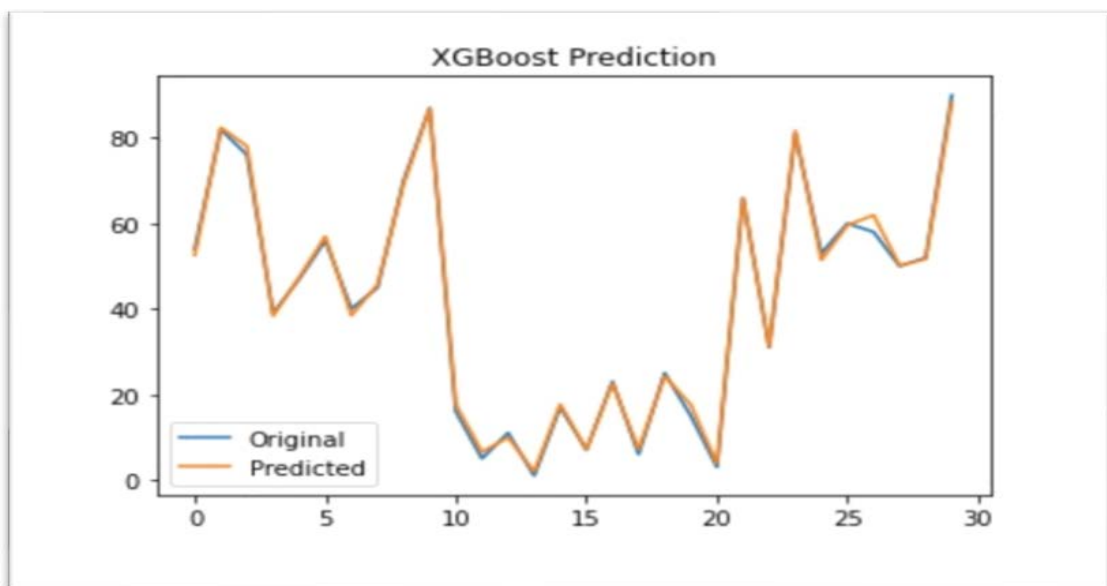


Figure 7.5 : XGBoost method ranking compared to the actual ranking of the whole dataset

7.4.3 Evaluation of the DNNs method

We evaluate the performance of the DNNs model using the MAE metric for the training, validation, testing, and the top-30 dataset of the TNvs.

Table 7.17 summarizes the results of the experiments using varying numbers of hidden layers (1, 3, 4 and 5 hidden layers) in the DNNs model. This table includes the best performing model in each DNNs experiment as discussed in section 7.3.3.

Table 7.17 : Summary of the experiments using DNNs with varying numbers of hidden layers

Hidden layers	Epoch count	Learning rate	MAE training	MAE validation	MAE testing	MAE-top-30
1	3000	0.0001	11.958	11.954	12.511	1.489
3	2000	0.0001	10.067	10.860	10.925	1.337
4	2000	0.0001	9.688	10.693	10.770	0.826
5	2000	0.0001	7.990	8.507	8.605	0.859

As shown in Table 7.17, increasing the number of hidden layers from 1 to 3, 4, and 5 has improved the model's performance. We can also see that the reduction in the epoch count indicates faster convergence when using more hidden layers.

Additionally, we can see a consistent improvement in MAE for the training, validation, and testing dataset as the number of hidden layers increases, specifically for the top 30 predictions.

Table 7.17 shows that DNNs with four layers achieved the best performance in terms of MAE, with an MAE value of 0.826 for the top-30 nodes dataset. This indicates that the DNNs model with four hidden layers is better at predicting the CPU utilization and memory usage percentage for the TNvs than the other models with a different number of hidden layers.

Furthermore, the results of the experiments also show that the DNN model with

four hidden layers required 2000 epochs for training, and the learning rate was set to 0.0001. This indicates that the DNNs model required a moderate number of epochs to achieve optimal performance, and a relatively small learning rate was sufficient for the model to learn the patterns in the data.

Figures 7.6, 7.7 and 7.8 depict the ranking of the training dataset, testing dataset and the top-30 nodes predicted by the DNNs model with four hidden layers compared to the actual ranking, respectively. The figures illustrate that the ranking result of DNNs with four hidden layers is very close to the optimal ranking. This indicates that the DNNs model is able to accurately predict the ranking of the FNvs based on their criteria of CPU utilization and memory usage percentage.

In conclusion, the results of the experiments shown in Table 7.17 indicate that the DNNs model with four hidden layers is the most effective in predicting the CPU utilization and memory usage percentage of the TNvs. The DNNs model achieved the best performance in terms of the MAE metric, and the ranking result of the DNNs model was very close to the optimal ranking. These findings suggest that the DNNs model with four hidden layers is a suitable approach for ranking the TNvs based on the criteria of CPU utilization and memory usage percentage.

7.4.4 Comparison of the three methods used to develop the iVFC-TNv selection module

After conducting experiments using two MCDM methods (AHP and TOPSIS), one ML method (XGBoost) and four DL methods (DNNs), and based on the results shown in Table 7.18 and in Figures 7.9 and 7.10, it is clear that the DNNs with four layers outperformed the statistical method (TOPSIS) and the ML method (XGBoost) in terms of the MAE metric. MAE is an important evaluation metric that measures the average difference between the predicted and actual values. Thus, a lower MAE value indicates better performance.

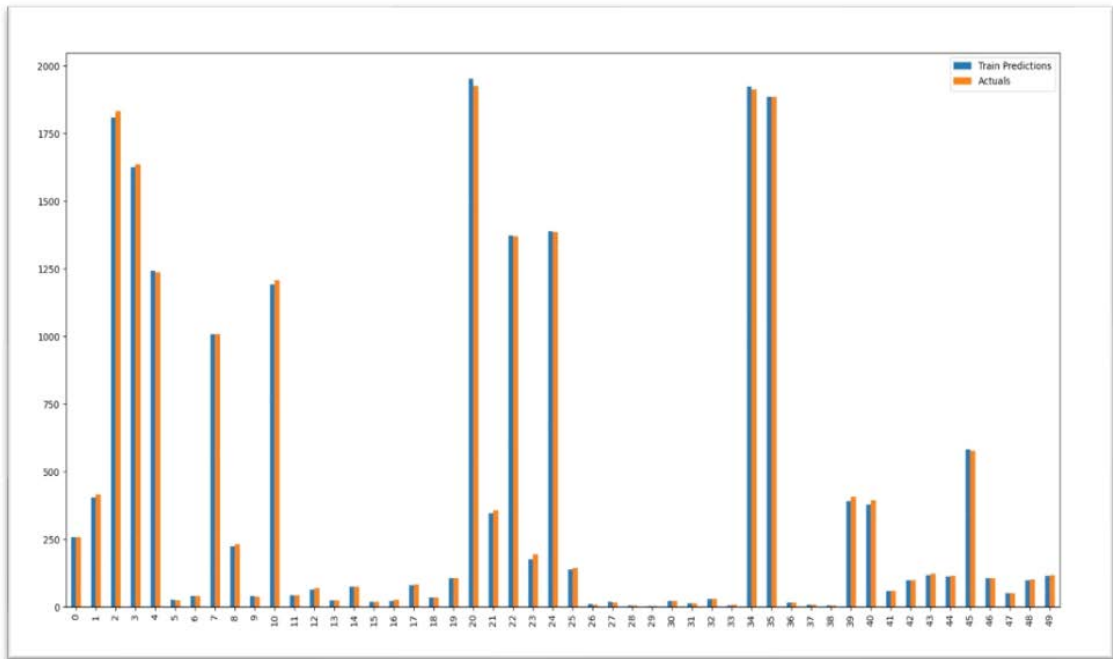


Figure 7.6 : Prediction results for the training dataset using DNNs with four layers

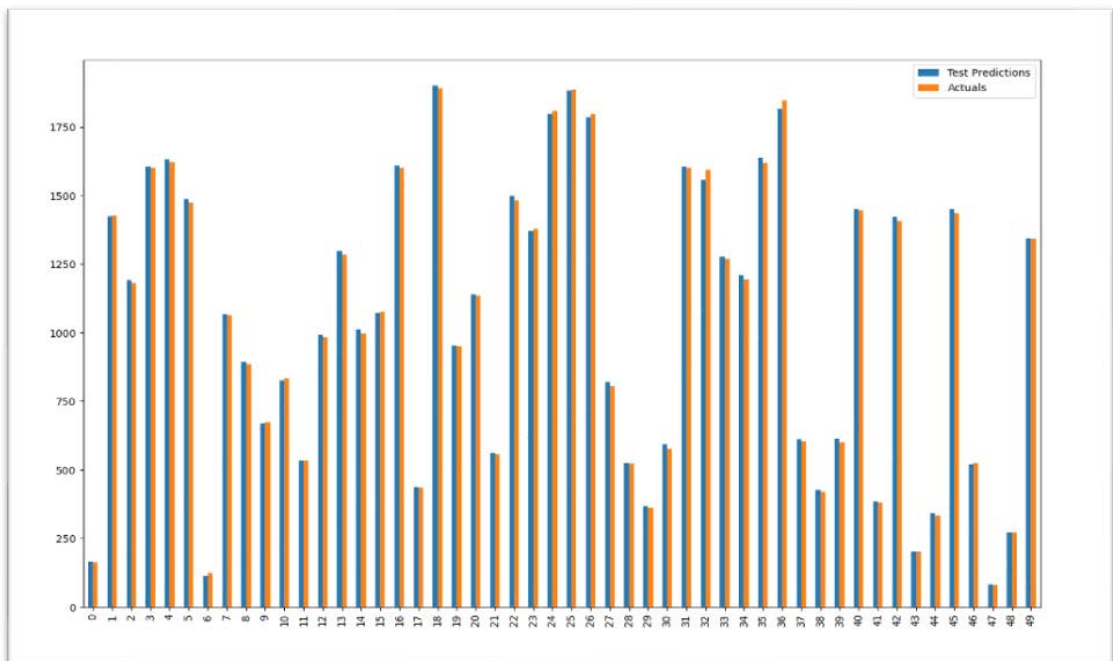


Figure 7.7 : Prediction results for the testing dataset using DNNs with four layers

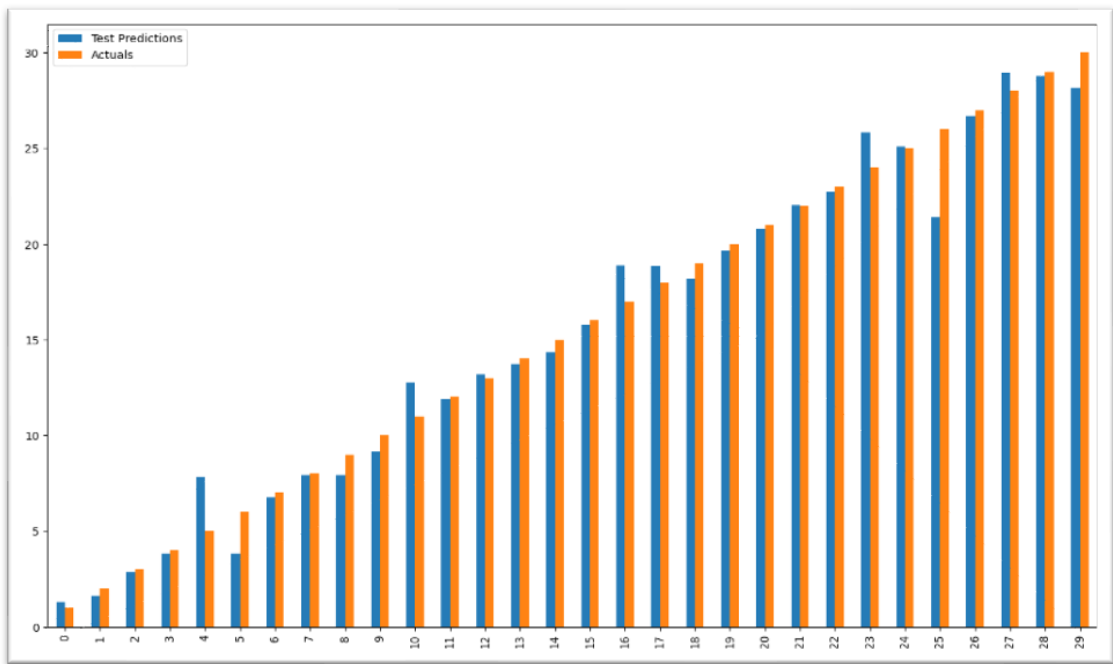


Figure 7.8 : Prediction results for the top-30 testing dataset using DNNs with four layers

This means that the DNNs with four layers performed better at predicting the ranking of the TNvs list based on the CPU utilization and memory usage percentage criteria.

Although the performance of the TOPSIS method was close to the performance of the DNN method in terms of the MAE metric, it is important to note that TOPSIS required significantly less run time than the DNNs method. This indicates that TOPSIS may be more efficient for applications that require faster responses. However, DNNs perform much faster if they can be trained by optimizing the hyperparameters and using more powerful computing resources, such as GPUs.

In comparison to the ML method (XGBoost), the DNNs with four layers significantly outperformed it in terms of the MAE metric, however it takes a longer time to run. In this sense, DNNs are better at capturing complex patterns in data

than XGBoost. However, it should be noted that the XGBoost method has its own strengths such as higher accuracy, which may be required for certain types of data or applications.

Overall, the results indicate that the DNN with four layers was the most effective method for ranking the different TNvs based on the criteria of CPU utilization and memory usage percentage, based on the MAE metric. The TOPSIS approach, on the other hand, may be more efficient for specific applications that require a faster response time. When choosing the best method for a given application, it is crucial to take into account both the strengths and weaknesses of the various approaches.

Table 7.18 : MAE and run time for the three methods for TNv selection

	Statistical method (TOPSIS)	DL method (DNN with four layers)	ML method (XGBoost with learning rate= 0.1 and n_estimators = 100)
MAE	0.963	0.826	6.937
Run time	3.26 secs	8 mins 22 secs	0.3 s

7.5 Conclusion

This chapter describes the development of the iVFC-TNv selection module using three methods, namely statistical methods (AHP and TOPSIS), an ML method (XGBoost) and DL methods (DNNs). These methods are used to rank the list of the TNvs that agreed to participate in the task offloading process to help the overloaded FNV select the most optimal TNv from the list that can carry out the task effectively.

The performance of the TNv selection module was evaluated based on the MAE metric, which gives an optimal result of 0.963 using the TOPSIS method and 0.826

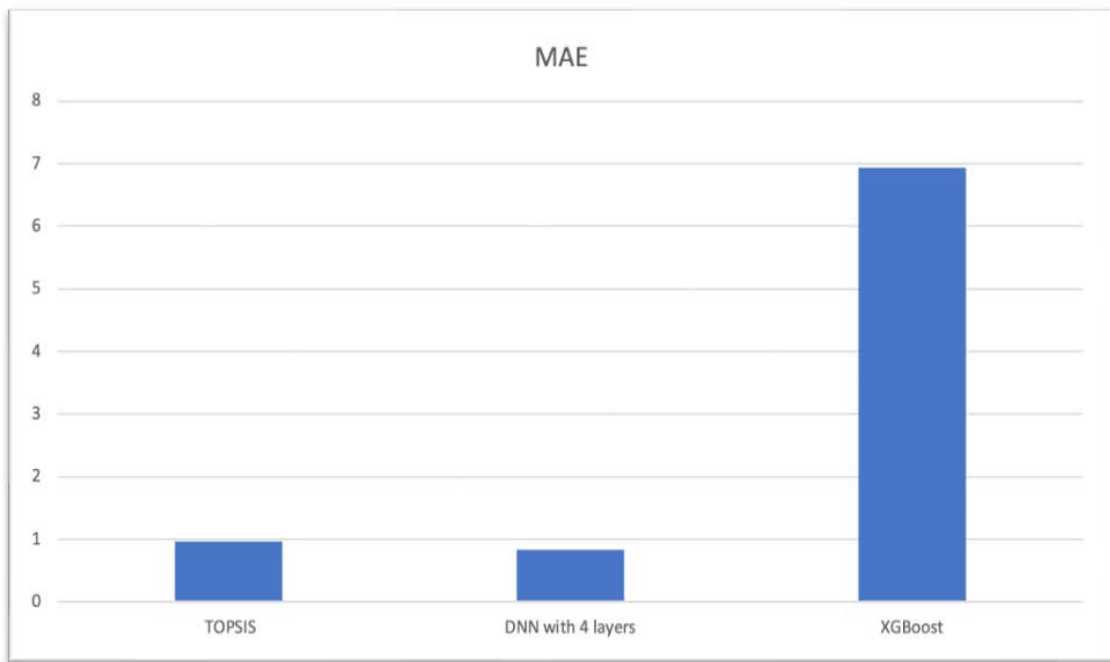


Figure 7.9 : Comparing the different ranking methods based on MAE

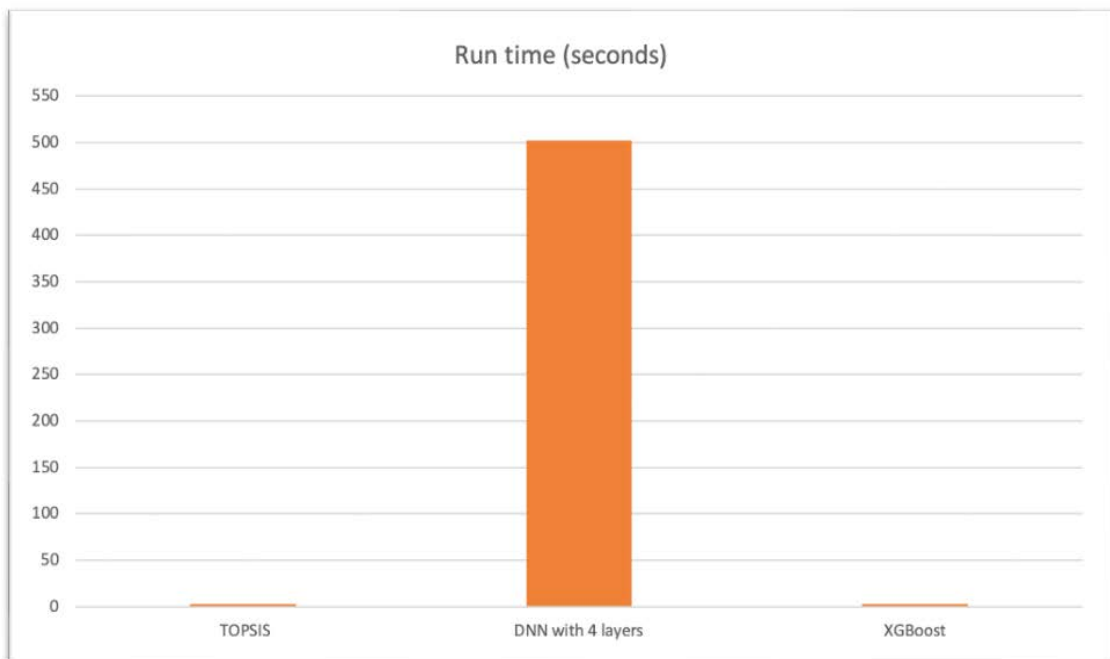


Figure 7.10 : Comparing the different ranking methods based on run time in seconds

using the DNNs method with four hidden layers, with a slight difference in the execution time where the TOPSIS method outperforms DNNs in terms of the execution time. The experiments results indicate that the proposed selection module works well in selecting the optimal TNv that can handle the task. The experiment results show that our proposed framework can effectively guarantee the proper selection of the optimal TNv.

Next chapter will conclude this thesis through summarizing the main points presented throughout the thesis and outline specific areas for future investigation and development.

Chapter 8

Conclusion and future work

8.1 Introduction

This chapter concludes this thesis by summarizing the research findings and providing recommendations for future work. This thesis developed an intelligent method for proactive-based task offloading in vehicular fog computing (VFC). While previous researchers have developed several reactive-based solutions to address task offloading issues in VFC where the offloading decision occurs when the fog node vehicle (FNv) consumes all of its computation resources and has no resources to process its tasks, this thesis differs as it provides a proactive-based method for the task offloading process where the workload of the FNv is predicted to provide a future overloading decision. The lack of a solution to provide proactive-based task offloading in VFC has been proved in chapter 2 where the results of a systematic literature review and the extensive analysis of prior research are presented. Based on an analysis of the selected literature, research gaps were identified, leading to the development of a novel solution called iVFC, a proactive-based methodology for task offloading in VFC to address the identified research gaps.

This chapter is organized as follows. Section 8.2 discusses the research gaps that have been addressed in this thesis. Section 8.3 outlines the contributions of this thesis to the existing literature and 8.4 concludes our thesis and provides recommendations for future work.

8.2 Problems addressed in this thesis

This thesis primarily focuses on addressing significant gaps in the existing literature related to task offloading in VFC. Based on the systematic literature review reported in chapter 2, the following research gaps were identified and addressed in this thesis:

1. All of the task offloading target node vehicle (TNv) selection from the (host) source node vehicle (SNv) is done on-the-fly, so there are little or no predictive mechanisms at play for offloading the tasks to a TNv. No method has been proposed to provide a proactive-based task offloading where the overloading FNv will have prior knowledge of the overloading status based on its workload.
2. There is no intelligent selection mechanism to help the overloaded SNv to select an optimal TNv to efficiently handle the offloaded task.
3. There is no existing method to provide an incentive mechanism to encourage fog nodes to participate in the task offloading process, taking into account the number of the participants in the VFC environment. Also, in most of the proposed incentive mechanisms, the nature of incentive or penalty and its implementation is unclear.

The next section summarizes the contributions of this thesis to the existing literature.

8.3 Contributions to the existing literature

Based on the research gaps identified in the literature, the main contribution of this thesis to the existing literature is the development of an intelligent framework called iVFC for task offloading in VFC. The following sub-sections outline the contributions of the proposed iVFC framework to the existing literature.

8.3.1 Systematic literature review (SLR)

In this thesis, we carried out an extensive state-of-the-art systematic literature review (SLR) in the area of task offloading in VFC. This SLR is documented in chapter 2 of this thesis. To conduct this SLR, we identified the search terms related to task offloading in VFC. These terms were input into four prestigious databases to retrieve relevant papers and the results of the search process were evaluated based on the inclusion and exclusion criteria to select the relevant studies. The selected studies were critically reviewed to identify the research gaps in the literature. Based on the review outcomes, we found that no framework is proposed in the literature to provide a proactive-based task offloading method in VFC. This SLR was published in the Future Generation Computer System journal, which is ranked in the top quartile of journals (JCR Q1). The contents of this SLR are available at the following link: <https://doi.org/10.1016/j.future.2022.03.019>

8.3.2 Development of a novel framework called iVFC for task offloading in VFC

In this thesis, an intelligent framework called iVFC was developed which offers an intelligent task offloading process in VFC system. This intelligent process comprises three modules: the predictive analytic module to enable the SNvs to priori know the future overloading condition, an incentive module to encourage FNvs to participate in the offloading process and a TNv selection module to help the overloaded SNv to select the most optimal TNv to efficiently handle the offloaded task. To the best of our knowledge, no such framework has been proposed in the literature. These three modules that comprise the intelligent iVFC framework are as follows:

- a) **A proactive-based framework for task offloading in VFC using machine learning (ML) time-series prediction methods**

The first module of the proposed iVFC framework provides a proactive-based task offloading method in VFC to assist FNvs make future offloading decisions. To do this, ML time-series prediction methods were used to predict the future workload of FNvs based on their current workload (i.e., their CPU utilization and memory usage). Using our proposed proactive-based framework, a FNv will have prior knowledge of when it will become overloaded and when it needs to offload its task in the future, which helps to reduce processing delay as the proactive decision-making process helps to minimize the time spent in making offloading decisions during runtime.

b) **A Stackelberg game-based mechanism for incentivising the vehicular fog nodes to participate in the iVFC**

The second module of the proposed iVFC framework motivates the FNvs on the network to participate and share their idle resources to process the offloaded task by offering them rewards as a reputation value for resource sharing, and in the case they decline the option to participate, the FNvs will receive a penalty as a negative or zero reputation. To do this, the Stackelberg game theory was used to provide incentives to FNvs to increase the participation level of FNvs to guarantee the availability of sufficient nodes to process the offloaded task. Our proposed incentive mechanism helps in increasing the participation level of fog nodes within the VFC environment.

c) **An intelligent framework for target node (TNv) selection in the iVFC system**

The third module of the proposed iVFC framework helps the overloaded SNv select the most optimal TNv that can efficiently handle the offloaded task. To do this, statistical methods, ML method and DL methods were used to rank the different FNvs based on their available workload (CPU utilization

and memory usage). Based on the ranking results, a list of all FNvs that have sufficient computation resources, ordered from highest to lowest resources, is prepared. This list is then used by the fog server node (FSN) to select the most optimal TNv/TNvs from the list to process the task. Our proposed TNv selection framework helps the overloaded SNv to intelligently select the most optimal TNv/TNvs to process its task.

8.3.3 Evaluation and validation of the proposed framework

To evaluate the performance of our proposed solution, we developed a software prototype. The software prototype comprises three modules, the predictive analytic module, the incentive module, and the TNv selection module. The performance of the approaches proposed in this thesis was evaluated and tested based on several evaluation metrics. Chapters 5, 6 and 7 demonstrate the development and the evaluation of the three modules, corresponding to research objectives 1, 2 and 3.

8.4 Conclusion and future work

Our proposed research framework is designed to intelligently manage the task offloading process within a VFC environment, however, there remain several issues that can be explored in the future. The following areas will be addressed in our future research:

a) **Developing a registration mechanism to register the vehicular fog nodes**

Our proposed framework assumes that each vehicle has to register with the iVFC system to become an iVFC client and act as a fog node vehicle. Our framework does not include a registration mechanism to register vehicles to the system and store their information in the fog repository. Details on the vehicle's registration process are out of this research scope. Future research

can involve a registration mechanism to register any vehicle that requests to become part of the iVFC system.

b) Using the reputation value as another parameter for optimal TNv selection

Our proposed TNv selection module ranks the FNvs in the network based on their current workload parameters (CPU utilization and memory usage). Future research can include multiple relevant parameters to evaluate FNvs and select the optimal TNv to process the task. One of the parameters is the reputation of the FNv. Selecting the optimal TNv based on its reputation will contribute to guarantee the trustworthiness and reliability of the chosen TNv.

c) Using the reputation value of the TNv as a feedback mechanism to decide whether to form a future offloading request with it or not

As the SNv depends on the TNv to complete its offloaded tasks, it needs to ensure that it forms a service level agreement (SLA) only with nodes that have a record of committing to the offloaded tasks. This distinguishes a malicious TNv from a reliable or trusted one to reduce offloading failures. One way of doing this is to determine the social reputation of the TNv after the task offloading process, which represents the extent of the commitment to the formed SLA within the prescribed deadline. This assessment will help other SNvs select the appropriate TNv. The objective of this requirement is to compute this reputation value.

d) Predicting the future reputation of FNvs based on their participation

Currently, we compute reputation based on the offered reward for resource sharing. In future work, another mechanism can be developed to predict the future reputation of FNvs. Predicting the future reputation of FNvs will help

those nodes to make a decision on sharing their resources in the future, which will reduce the delay resulting from the negotiation process on resource sharing.

e) **Implementing our proposed iVFC system in a real VFC environment**

In this research, we proposed the iVFC framework and developed a software prototype to evaluate the proposed framework. Future work can build a real VFC system that uses our proposed framework.

Bibliography

- [1] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho, “Urban planning and building smart cities based on the internet of things using big data analytics,” *Computer Networks*, vol. 101, pp. 63–80, 2016, industrial Technologies and Applications for the Internet of Things.
- [2] R. Stair and G. Reynolds, *Principles of information systems*. Cengage Learning, 2020.
- [3] D. Bandyopadhyay and J. Sen, “Internet of things: Applications and challenges in technology and standardization,” *Wireless personal communications*, vol. 58, pp. 49–69, 2011.
- [4] J. Zhao, M. Kong, Q. Li, and X. Sun, “Contract-based computing resource management via deep reinforcement learning in vehicular fog computing,” *IEEE Access*, vol. 8, pp. 3319–3329, 2020.
- [5] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, “Vehicular fog computing: A viewpoint of vehicles as the infrastructures,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.
- [6] C. Tang, C. Zhu, X. Wei, H. Peng, and Y. Wang, “Integration of uav and fog-enabled vehicle: Application in post-disaster relief,” in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, 2019, pp. 548–555.
- [7] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, “Reliable task offloading for vehicular fog computing under information asymmetry and information

- uncertainty,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8322–8335, 2019.
- [8] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, “Adaptive learning-based task offloading for vehicular edge computing systems,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3061–3074, 2019.
- [9] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, “Vehicular edge computing and networking: A survey,” *Mobile networks and applications*, vol. 26, pp. 1145–1168, 2021.
- [10] Y. Xiao and C. Zhu, “Vehicular fog computing: Vision and challenges,” in *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*. IEEE, 2017, pp. 6–9.
- [11] X. Kui, Y. Sun, S. Zhang, and Y. Li, “Characterizing the capability of vehicular fog computing in large-scale urban environment,” *Mobile Networks and Applications*, vol. 23, pp. 1050–1067, 2018.
- [12] Z. Ning, J. Huang, and X. Wang, “Vehicular fog computing: Enabling real-time traffic management for smart cities,” *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, 2019.
- [13] A. M. A. Hamdi, F. K. Hussain, and O. K. Hussain, “Task offloading in vehicular fog computing: State-of-the-art and open issues,” *Future Generation Computer Systems*, vol. 133, pp. 201–212, 2022.
- [14] F. Lin, Y. Zhou, G. Pau, and M. Collotta, “Optimization-oriented resource allocation management for vehicular fog computing,” *IEEE Access*, vol. 6, pp. 69 294–69 303, 2018.
- [15] C. Zhu, G. Pastor, Y. Xiao, and A. Ylajaaski, “Vehicular fog computing for

- video crowdsourcing: Applications, feasibility, and challenges,” *IEEE Communications Magazine*, vol. 56, no. 10, pp. 58–63, 2018.
- [16] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, “Resource management approaches in fog computing: a comprehensive review,” *Journal of Grid Computing*, vol. 18, no. 1, pp. 1–42, 2020.
- [17] M. Kong, J. Zhao, X. Sun, and Y. Nie, “Secure and efficient computing resource management in blockchain-based vehicular fog computing,” *China Communications*, vol. 18, no. 4, pp. 115–125, 2021.
- [18] M. Aazam, S. Zeadally, and K. A. Harras, “Offloading in fog computing for iot: Review, enabling technologies, and research opportunities,” *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.
- [19] X. Gao, X. Huang, S. Bian, Z. Shao, and Y. Yang, “Pora: Predictive offloading and resource allocation in dynamic fog computing systems,” *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 72–87, 2020.
- [20] H. Liao, Z. Zhou, X. Zhao, B. Ai, and S. Mumtaz, “Task offloading for vehicular fog computing under information uncertainty: A matching-learning approach,” in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 2001–2006.
- [21] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, “Delay-sensitive task offloading in the 802.11 p-based vehicular fog computing systems,” *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 773–785, 2019.
- [22] Y. Wu, J. Wu, G. Zhou, and L. Chen, “A direction-based vehicular network model in vehicular fog computing,” in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of*

People and Smart City Innovation

(*SmartWorld/SCALCOM/UIC/ATC/CBDCCom/IOP/SCI*). IEEE, 2018, pp. 585–589.

- [23] S.-s. Lee and S. Lee, “Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information,” *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 450–10 464, 2020.
- [24] X. Gao, X. Huang, S. Bian, Z. Shao, and Y. Yang, “Pora: Predictive offloading and resource allocation in dynamic fog computing systems,” *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 72–87, 2019.
- [25] T. Liu, J. Li, F. Shu, and Z. Han, “Optimal task allocation in vehicular fog networks requiring urlc: An energy-aware perspective,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1879–1890, 2019.
- [26] T. Ye, X. Lin, J. Wu, G. Li, and J. Li, “Toward dynamic computation offloading for data processing in vehicular fog based f-ran,” in *2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2019, pp. 196–201.
- [27] C. Zhu, Y.-H. Chiang, A. Mehrabi, Y. Xiao, A. Ylä-Jääski, and Y. Ji, “Chameleon: Latency and resolution aware task offloading for visual-based assisted driving,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 9038–9048, 2019.
- [28] Q. Wu, H. Ge, H. Liu, Q. Fan, Z. Li, and Z. Wang, “A task offloading scheme in vehicular fog and cloud computing system,” *IEEE Access*, vol. 8, pp. 1173–1184, 2019.
- [29] S. Iqbal, A. W. Malik, A. U. Rahman, and R. M. Noor, “Blockchain-based

- reputation management for task offloading in micro-level vehicular fog network,” *IEEE Access*, vol. 8, pp. 52 968–52 980, 2020.
- [30] S. Keele *et al.*, “Guidelines for performing systematic literature reviews in software engineering,” 2007.
- [31] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, “Folo: Latency and quality optimized task allocation in vehicular fog computing,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4150–4161, 2018.
- [32] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, “Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8322–8335, 2019.
- [33] J. Zhao, M. Kong, Q. Li, and X. Sun, “Contract-based computing resource management via deep reinforcement learning in vehicular fog computing,” *IEEE Access*, vol. 8, pp. 3319–3329, 2019.
- [34] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, “Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3113–3125, 2019.
- [35] J. Xie, Y. Jia, Z. Chen, Z. Nan, and L. Liang, “Efficient task completion for parallel offloading in vehicular fog computing,” *China Communications*, vol. 16, no. 11, pp. 42–55, 2019.
- [36] S. Zhou, Y. Sun, Z. Jiang, and Z. Niu, “Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks,” *IEEE Communications Magazine*, vol. 57, no. 5, pp. 49–55, 2019.

- [37] Z. Wang, Z. Zhong, and M. Ni, “Application-aware offloading policy using smdp in vehicular fog computing systems,” in *2018 IEEE international conference on communications workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.
- [38] A. U. Rahman, A. W. Malik, V. Sati, A. Chopra, and S. D. Ravana, “Context-aware opportunistic computing in vehicle-to-vehicle networks,” *Vehicular Communications*, vol. 24, p. 100236, 2020.
- [39] Y.-D. Lin, J.-C. Hu, B. Kar, and L.-H. Yen, “Cost minimization with offloading to vehicles in two-tier federated edge and vehicular-fog systems,” in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–6.
- [40] M. Ran and X. Bai, “Vehicle cooperative network model based on hypergraph in vehicular fog computing,” *Sensors*, vol. 20, no. 8, p. 2269, 2020.
- [41] H. Li, X. Li, and W. Wang, “Joint optimization of computation cost and delay for task offloading in vehicular fog networks,” *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, p. e3818, 2020.
- [42] C. Xu, Y. Wang, Z. Zhou, B. Gu, V. Frascolla, and S. Mumtaz, “A low-latency and massive-connectivity vehicular fog computing framework for 5g,” in *2018 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2018, pp. 1–6.
- [43] X. Huang, D. Ye, R. Yu, and L. Shu, “Securing parked vehicle assisted fog computing with blockchain and optimal smart contract design,” *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 426–441, 2020.
- [44] Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, “Computation task assignment in vehicular fog computing: A learning approach via neighbor

- advice,” in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2019, pp. 1–5.
- [45] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. Rodrigues, X. Kong, J. Huang, and R. Y. Kwok, “Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1060–1072, 2019.
- [46] C. Tang, C. Zhu, X. Wei, H. Peng, and Y. Wang, “Integration of uav and fog-enabled vehicle: application in post-disaster relief,” in *2019 IEEE 25th international conference on parallel and distributed systems (ICPADS)*. IEEE, 2019, pp. 548–555.
- [47] X. Wu, S. Zhao, R. Zhang, and L. Yang, “Mobility prediction-based joint task assignment and resource allocation in vehicular fog computing,” in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.
- [48] S. Mu, Z. Zhong, and M. Ni, “Multi-destination computation offloading in vehicular networks,” in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2018, pp. 446–451.
- [49] T. Ye, X. Lin, J. Wu, G. Li, and J. Li, “Processing capability and qoe driven optimized computation offloading scheme in vehicular fog based f-ran,” *World Wide Web*, vol. 23, pp. 2547–2565, 2020.
- [50] T. Halabi and M. Zulkernine, “Reliability-driven task assignment in vehicular crowdsourcing: A matching game,” in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2019, pp. 78–85.

- [51] W. Chen, Z. Su, Q. Xu, T. H. Luan, and R. Li, “Vfc-based cooperative uav computation task offloading for post-disaster rescue,” in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 228–236.
- [52] Y. Wu, J. Wu, L. Chen, G. Zhou, and J. Yan, “Fog computing model and efficient algorithms for directional vehicle mobility in vehicular network,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 2599–2614, 2020.
- [53] C. Tang, X. Wei, C. Zhu, Y. Wang, and W. Jia, “Mobile vehicles as fog nodes for latency optimization in smart cities,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9364–9375, 2020.
- [54] C. Liu, K. Liu, H. Ren, Y. Zhou, L. Feng, S. Guo, and V. Lee, “Enabling safety-critical and computation-intensive iov applications via vehicular fog computing,” in *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*. IEEE, 2019, pp. 378–383.
- [55] B. Yang, M. Sun, X. Hong, and X. Guo, “A deadline-aware offloading scheme for vehicular fog computing at signalized intersection,” in *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2020, pp. 1–6.
- [56] C. Liu, K. Liu, S. Guo, R. Xie, V. C. Lee, and S. H. Son, “Adaptive offloading for time-critical tasks in heterogeneous internet of vehicles,” *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7999–8011, 2020.
- [57] I. Sorkhoh, D. Ebrahimi, C. Assi, S. Sharafeddine, and M. Khabbaz, “An infrastructure-assisted workload scheduling for computational resources

- exploitation in the fog-enabled vehicular network,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5021–5032, 2020.
- [58] H. Liao, Y. Mu, Z. Zhou, M. Sun, Z. Wang, and C. Pan, “Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4051–4063, 2020.
- [59] S. A. Kazmi, T. N. Dang, I. Yaqoob, A. Manzoor, R. Hussain, A. Khan, C. S. Hong, and K. Salah, “A novel contract theory-based incentive mechanism for cooperative task-offloading in electrical vehicular networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8380–8395, 2021.
- [60] O. Nazih, N. Benamar, and A. Addaim, “An incentive mechanism for computing resource allocation in vehicular fog computing environment,” in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*. IEEE, 2020, pp. 1–5.
- [61] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, “Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 198–14 211, 2020.
- [62] A. Lakhan, M. Ahmad, M. Bilal, A. Jolfaei, and R. M. Mehmood, “Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4212–4223, 2021.
- [63] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, “Priority-aware task offloading

- in vehicular fog computing based on deep reinforcement learning,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 067–16 081, 2020.
- [64] C. Liu, K. Liu, X. Xu, H. Ren, F. Jin, and S. Guo, “Real-time task offloading for data and computation intensive services in vehicular fog computing environments,” in *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 2020, pp. 360–366.
- [65] S. Vemireddy and R. R. Rout, “Fuzzy reinforcement learning for energy efficient task offloading in vehicular fog computing,” *Computer Networks*, vol. 199, p. 108463, 2021.
- [66] Z. Liu, P. Dai, H. Xing, Z. Yu, and W. Zhang, “A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 7, pp. 4388–4401, 2021.
- [67] A. Chopra, A. U. Rahman, A. W. Malik, and S. D. Ravana, “Adaptive-learning-based vehicle-to-vehicle opportunistic resource-sharing framework,” *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 497–12 504, 2021.
- [68] M. Ibrar, A. Akbar, S. R. U. Jan, M. A. Jan, L. Wang, H. Song, and N. Shah, “Artnet: Ai-based resource allocation and task offloading in a reconfigurable internet of vehicular networks,” *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 67–77, 2020.
- [69] S. A. Kazmi, T. M. Ho, T. T. Nguyen, M. Fahim, A. Khan, M. J. Piran, and G. Baye, “Computing on wheels: A deep reinforcement learning-based approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 22 535–22 548, 2022.

- [70] I. Sarkar and S. Kumar, “Delay-aware intelligent task offloading strategy in vehicular fog computing,” in *2022 International Conference on Connected Systems & Intelligence (CSI)*. IEEE, 2022, pp. 1–6.
- [71] Z. Wei, B. Li, R. Zhang, X. Cheng, and L. Yang, “Dynamic many-to-many task offloading in vehicular fog computing: A multi-agent drl approach,” in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 6301–6306.
- [72] Z. Gao, L. Yang, and Y. Dai, “Fast adaptive task offloading and resource allocation via multiagent reinforcement learning in heterogeneous vehicular fog computing,” *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6818–6835, 2022.
- [73] J. Shi, J. Du, J. Wang, and J. Yuan, “Federated deep reinforcement learning-based task allocation in vehicular fog computing,” in *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*. IEEE, 2022, pp. 1–6.
- [74] Y. Li, B. Yang, H. Wu, Q. Han, C. Chen, and X. Guan, “Joint offloading decision and resource allocation for vehicular fog-edge computing networks: A contract-stackelberg approach,” *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 969–15 982, 2022.
- [75] K. Wang, Y. Tan, Z. Shao, S. Ci, and Y. Yang, “Learning-based task offloading for delay-sensitive applications in dynamic fog networks,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 399–11 403, 2019.
- [76] F. Arena, G. Pau, and A. Severino, “An overview on the current status and future perspectives of smart cars,” *Infrastructures*, vol. 5, no. 7, p. 53, 2020.
- [77] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, “A survey of machine learning

- for big data processing,” *EURASIP Journal on Advances in Signal Processing*, vol. 2016, pp. 1–16, 2016.
- [78] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, “State-of-the-art in artificial neural network applications: A survey,” *Heliyon*, vol. 4, no. 11, 2018.
- [79] R. Bělohlávek and G. J. Klir, *Concepts and fuzzy logic*. MIT press, 2011.
- [80] K.-S. Chan and J. D. Cryer, *Time series analysis with applications in R*. Springer, 2008.
- [81] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.
- [82] S. Noreen and N. Saxena, “A review on game-theoretic incentive mechanisms for mobile data offloading in heterogeneous networks,” *IETE Technical Review*, vol. 34, no. sup1, pp. 15–26, 2017.
- [83] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, V. A. Kamaev *et al.*, “A survey of forecast error measures,” *World applied sciences journal*, vol. 24, no. 24, pp. 171–176, 2013.
- [84] Z. ur Rehman, O. K. Hussain, and F. K. Hussain, “IaaS cloud selection using mcdm methods,” in *2012 IEEE Ninth international conference on e-business engineering*. IEEE, 2012, pp. 246–251.
- [85] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

- [86] “Gwa-t-12 bitbrains,” <http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains>, 2015, accessed: 2023-01-30.
- [87] S. Shen, V. Van Beek, and A. Iosup, “Statistical characterization of business-critical workloads hosted in cloud datacenters,” in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2015, pp. 465–474.
- [88] J. Chilberto, S. Zaal, G. Aroraa, E. Price, J. Chilberto, S. Zaal, G. Aroraa, and E. Price, “Exploring the azure portal,” https://doi.org/10.1007/978-1-4842-5437-0_2, 2015.
- [89] S. Tarannum and S. Jabin, “A comparative study on fuzzy logic and intuitionistic fuzzy logic,” in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. IEEE, 2018, pp. 1086–1090.
- [90] E. H. Mamdani, “Application of fuzzy algorithms for control of simple dynamic plant,” in *Proceedings of the institution of electrical engineers*, vol. 121, no. 12. IET, 1974, pp. 1585–1588.
- [91] O. Caelen, “A bayesian interpretation of the confusion matrix,” *Annals of Mathematics and Artificial Intelligence*, vol. 81, no. 3-4, pp. 429–450, 2017.
- [92] A. Abdel-Hafez, Y. Xu, and A. Jøsang, “A normal-distribution based reputation model,” in *Trust, Privacy, and Security in Digital Business: 11th International Conference, TrustBus 2014, Munich, Germany, September 2-3, 2014. Proceedings 11*. Springer, 2014, pp. 144–155.
- [93] “Welcome to Colab!, howpublished = <https://colab.research.google.com>, note = Accessed: 2023-07-18.”

- [94] E. Bisong and E. Bisong, “Google colaboratory,” *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, pp. 59–64, 2019.
- [95] A. M. Mostafa, “An mcdm approach for cloud computing service selection based on best-only method,” *IEEE Access*, vol. 9, pp. 155 072–155 086, 2021.
- [96] T. L. Saaty, “How to make a decision: the analytic hierarchy process,” *European journal of operational research*, vol. 48, no. 1, pp. 9–26, 1990.
- [97] J. Sidhu and S. Singh, “Improved topsis method based trust evaluation framework for determining trustworthiness of cloud service providers,” *Journal of Grid Computing*, vol. 15, pp. 81–105, 2017.
- [98] S. Chakraborty, “Topsis and modified topsis: A comparative analysis,” *Decision Analytics Journal*, vol. 2, p. 100021, 2022.
- [99] U. Michelucci, *Applied Deep Learning with TensorFlow 2: Learn to Implement Advanced Deep Learning Techniques with Python*. Springer, 2022.