UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Doctoral Thesis

Enhancement of vehicular ad hoc networks using machine learning-based prediction methods

by:

Leticia Lemus Cárdenas

Ph.D. Advisors:

Dr. Mónica Aguilar Igartua

Dr. Ahmad Mohamad Mezher

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Telematics

SISCOM

(Smart Services for Information Systems and Communication Networks)

Departament of Network Engineering

June 17, 2020

“No estudio por saber más, sino para ignorar menos.”

Sor Juana Inés de la Cruz

Abstract

Society is aware that the ecological predation of the planet is due to human activity. Therefore, it is necessary to carry out actions to help reverse this damage. Additionally, the trend of population growth in big cities and the uncontrolled volume of traffic cause serious problems such as traffic delays, traffic jams, increased CO_2 emissions, and traffic accidents. In this sense, so-called smart cities are motivated to create a greener and safer environment where both efficient mobility and public services seek to mitigate those problems. These initiatives are supported by smart technologies such as the Internet of things (IoT) and information and communication technologies (ICT) that provide the basis for creating and implementing many interesting smart city projects. In this context, modern vehicles today are equipped with a variety of sensors that enable them to detect and share information. This detected information can not only be useful for other vehicles, but also to collect relevant data related to traffic management such as traffic congestion, energy use, and so on. This data can help to generate smart mobility solutions and help to improve city services. In this context, vehicular ad hoc networks (VANETs) enable communication between vehicles (V2I) and also between vehicles and the city's fixed infrastructure (V2I). Additionally, VANET routing protocols can help to minimize the use of fixed infrastructure as they employ multi-hop V2V communication to reach the road side units (RSUs) of the city.

In this research framework, this thesis aims to contribute to the design of VANET routing protocols for urban environments. In the first part of the thesis, we have started our research work analyzing some important aspects of the hop-by-hop forwarding routing based on the evaluation of node metrics. We have analyzed different weighting strategies to compute a multimetric score used to arrange the candidate nodes (neighbouring nodes of the node currently holding the packet) to be chosen as the next node to forward the packet. As a result, we have proposed a weighted power mean function (W-PMF) to improve the selection of the best forwarding node. In this proposal, we have shown that the best way to combine several metrics is by implementing the geometric mean function. Additionally, we have improved the selection of forwarding nodes by accurately estimating their current position at the moment of forwarding messages, instead of using the position information received in the last beacon.

Nowadays, many applications and services are based on big data because valuable information can be extracted when the data is properly processed. Under this premise, it has been considered relevant to collect historical data about vehicular network conditions and

related them to the related performance of the VANET. The collected data provides us useful information to design new routing strategies based on predictions. With this goal in mind, we have evaluated different routing metrics (distance to destination, vehicles' density, available bandwidth) that describe the network conditions when the vehicle currently holding a packet needs to choose the best next hop to forward the packet towards a specific destination, following a hop-by-hop scheme. A large number of simulations under different representative scenarios have been conducted to collect the values of the routing metrics and the related binary output value (successfully delivered or not at destination). Then, we have carried out an offline dataset analysis where a statistical model for each metric has been obtained. This statistical model has been used to design a new forwarding algorithm for VANETs. With this algorithm, we have proposed a probability-based multimetric routing protocol (ProMRP), which selects the candidate nodes based on the highest probability to deliver the packet at destination. Also, an accurate estimation of the node's position has been included to obtain an enhanced version called EProMRP.

The last contribution is focused on the application of machine learning techniques to enhance routing decisions. To this end, a new dataset has been collected from five routing metrics (distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses). The collected dataset has been used to generate two different machine learning models: (i) a decision tree with a maximum tree depth of 20, and (ii) an artificial neural network with a dimension 1/3/1 (one input layer, 3 hidden layers and one output layer). Our proposals are called (i) multimetric predictive ML-based routing protocol (MPML), and (ii) multimetric predictive ANN-based routing protocol (MPANN), respectively. The goal is to generate machine learning-based forwarding models with the highest prediction accuracy. They have been assessed with both machine learning performance metrics and realistic VANET simulations. Different VANET scenarios with different network sizes and city maps, different orographies and road layouts, different vehicles' densities, have been considered to evaluate each proposal. We have also evaluated the level of flexibility of our proposals to adapt to other city scenarios, different from the city map used to train the model. For this purpose, MPANN (which has shown to be the best proposal) has been tested on two urban areas of different cities with different characteristics. With this latter evaluation, we have been able to assess the ability of our machine learning-based forwarding model to adapt to new network conditions.

Resumen

La Sociedad es consciente de que la depredación ecológica del planeta se debe a la actividad humana. Por lo tanto, es necesario llevar a cabo acciones para ayudar a revertir este daño. Sumado a esto, la tendencia de crecimiento de la población en las grandes ciudades y el volumen incontrolado de tráfico causan serios problemas, como demoras en el tráfico, atascos, aumento de las emisiones de CO_2 y accidentes de tráfico. En este sentido, las denominadas ciudades inteligentes están motivadas para crear un entorno más verde y seguro donde tanto la movilidad eficiente como los servicios públicos buscan mitigar esos problemas. Estas iniciativas están respaldadas por tecnologías inteligentes como la Internet de las cosas (IoT) y las tecnologías de la información y las comunicaciones (TIC) que proporcionan la base para crear e implementar numerosos interesantes proyectos para ciudades inteligentes. En este contexto, los modernos vehículos de hoy en día están equipados con una variedad de sensores que les permiten detectar y compartir información. Esta información detectada no solo puede ser útil para otros vehículos, sino también para recopilar datos relevantes relacionados con la gestión del tráfico, como nivel de congestión de tráfico, uso de energía, etc. para generar soluciones de movilidad inteligente y contribuir en mejorar los servicios de la ciudad. En este contexto, las redes de vehículos ad hoc (VANET) permiten la comunicación entre vehículos (V2I) y también entre vehículos y la infraestructura fija de la ciudad (V2I). Además, los protocolos de encaminamiento VANET minimizan el uso de infraestructura fija, ya que emplean comunicaciones V2V de múltiples saltos para llegar a las unidades de infraestructura de comunicaciones de la ciudad.

En este marco de investigación, esta tesis tiene como objetivo contribuir al diseño de protocolos de encaminamiento de redes vehiculares ad hoc (VANET) en entornos urbanos. Para este propósito, hemos iniciado nuestro trabajo de investigación analizando algunos aspectos importantes del encaminamiento salto-a-salto basado en la evaluación de métricas de encaminamiento de los nodos. Hemos analizado diferentes estrategias de ponderación de las métricas para calcular un valor multimétrica con el que ordenar los nodos candidatos (nodos vecinos del nodo que actualmente tiene el paquete) a ser elegidos como siguiente nodo encaminador del paquete hacia su destino. Como resultado, hemos propuesto una función de potencia con media ponderada (W-PMF) para mejorar la selección del mejor nodo de encaminamiento. En esta propuesta, demostramos que la mejor manera de combinar varias métricas es implementando la función de media

geométrica. Adicionalmente, hemos mejorado la selección de nodos encaminadores estimando su posición en el momento de reenvío de paquetes con mayor precisión, en vez de utilizar la posición recibida en el último mensaje *beacon*.

Actualmente, diversas aplicaciones y servicios se basan en grandes volúmenes de datos (*big data*) de la que se puede extraer información valiosa cuando se procesan adecuadamente los datos. Bajo esta premisa, se ha considerado relevante recopilar datos históricos sobre las condiciones de la red vehicular y relacionarlos con las prestaciones correspondientes de la VANET. Los datos recopilados nos proporcionan información útil para diseñar nuevas estrategias de encaminamiento basadas en predicciones. Con este objetivo en mente, hemos evaluado diversas métricas de encaminamiento (distancia al destino, densidad de vehículos, ancho de banda disponible) que describen las condiciones de la red cuando el vehículo que actualmente tiene el paquete necesita elegir el mejor siguiente nodo para encaminar el paquete hacia su destino, siguiendo un esquema salto-a-salto. Se ha realizado una gran cantidad de simulaciones bajo diferentes escenarios representativos para crear un conjunto de datos a partir de los valores de las métricas de encaminamiento y del valor binario del resultado (entrega con éxito o no en destino) asociado. Luego, hemos llevado a cabo un análisis del conjunto de datos generado y se obtuvo un modelo estadístico para cada métrica. Este modelo estadístico se ha utilizado para diseñar un nuevo algoritmo de encaminamiento de paquetes en VANETs. Con dicho algoritmo, hemos propuesto un protocolo de encaminamiento multimétrico basado en probabilidades (ProMRP), que selecciona los nodos candidatos en función de la mayor probabilidad que asegure la entrega del paquete en su destino. Además, se ha incluido una estimación precisa de la posición del nodo para así obtener una versión mejorada de nuestro protocolo denominada EProMRP.

La última contribución se centra en la aplicación de técnicas de aprendizaje automático (*machine learning*) para mejorar las decisiones de encaminamiento. Con este fin, se ha recopilado un nuevo conjunto de datos de cinco métricas (distancia al destino, trayectoria, densidad de vehículos, ancho de banda disponible y pérdidas a nivel MAC). El conjunto de datos recopilado se ha utilizado para generar dos modelos diferentes de aprendizaje automático: (i) un árbol de decisión con una profundidad máxima del árbol igual a 20, y (ii) una red neuronal artificial de dimensión 1/3/1 (una capa de entrada, 3 capas ocultas y una capa de salida). Nuestras propuestas se denominan (i) protocolo de encaminamiento multimétrico predictivo basado en *machine learning* (MPML), y (ii) protocolo de encaminamiento predictivo multimétrico basado en red neuronal artificial (MPANN), respectivamente. El objetivo es generar modelos de aprendizaje automático con la predicción exacta más alta. Las propuestas han sido evaluadas mediante métricas de rendimiento de aprendizaje automático y mediante simulaciones VANET realistas. Se han considerado diferentes escenarios VANET con diferentes tamaños de red, mapas de ciudades con diferente orografía y disposición de las calles, diversas densidades de vehículos, para evaluar cada propuesta. También hemos evaluado el nivel de flexibilidad de nuestra propuesta para adaptarse a otros escenarios de ciudades distintas al mapa urbano utilizado para entrenar los modelos. Para este propósito, MPANN (que ha mostrado ser la mejor propuesta) ha sido probado en dos áreas urbanas de distintas ciudades con características diferentes. Con esta última evaluación, hemos podido medir la capacidad de nuestro modelo para adaptarse a nuevas condiciones de red.

Resum

La Societat és conscient que la predació ecològica del planeta es deu a l'activitat humana. Per tant, cal dur a terme accions per ajudar a revertir aquest dany. A això s'hi afegeix la tendència del creixement de la població a les grans ciutats i el volum de trànsit descontrolat que causen greus problemes, com retards deguts al tràfic, embussos, augment de les emissions de CO_2 i accidents de tràfic. En aquest sentit, les denominades ciutats intel-ligents estan motivades a crear un entorn més ecològic i més segur on tant la mobilitat eficient com els serveis públics pretenen mitigar aquests problemes. Aquestes iniciatives compten amb el suport de tecnologies intel·ligents com la Internet de les coses (IoT) i les tecnologies de la informació i les comunicacions (TIC) que proporcionen les bases per crear i implementar nombrosos projectes interessants de ciutat intel·ligent. En aquest context, els moderns vehicles actuals disposen d'una varietat de sensors que permeten detectar i compartir informació. Aquesta informació detectada no només pot ser útil per a altres vehicles, sinó també per recopilar dades rellevants relacionades amb la gestió del tràfic, com ara el nivell de congestió del tràfic, l'ús d'energia, etc. per a generar solucions de mobilitat intel·ligent i millorar els serveis de la ciutat. En aquest context, les xarxes de vehicles ad hoc (VANET) permeten la comunicació entre vehicles (V2I) i també entre vehicles i la infraestructura fixa de la ciutat (V2I). A més, els protocols d'encaminament VANET minimitzen l'ús d'infraestructura fixa ja que utilitzen comunicacions V2V multi-salt per arribar a les unitats d'infraestructura de comunicacions de la ciutat.

En aquest marc de recerca, aquesta tesi vol contribuir al disseny de protocols d'encaminament de xarxa ad-hoc (VANET) en entorns urbans. Amb aquesta finalitat, hem iniciat el nostre treball de recerca analitzant alguns aspectes importants de l'encaminament salt-a-salt basat en l'avaluació de les mètriques dels nodes. Hem analitzat diferents estratègies de ponderació de les mètriques per a calcular un valor multimètrica segons el qual ordenar els nodes candidats (nodes veïns del node que actualment té el paquet) a ser triats com a següent node encaminador del paquet cap al seu destí. Com a resultat, hem proposat una funció de potència mitjana ponderada (W-PMF) per millorar la selecció del millor node d'encaminament. En aquesta proposta, vam demostrar que la millor manera de combinar diverses mètriques és mitjançant la implementació de la funció de mitja geomètrica. Addicionalment, hem millorat la selecció de l'estratègia de selecció de nodes de reenviament estimant la seva posició en el moment de reenviar el paquet, enlloc d'utilitzar la posició rebuda en el darrer missatge *beacon*.

Actualment, diverses aplicacions i serveis es basen en grans volumens de dades (*big data*, de la qual es pot extraure valuosa informació quan es processen les dades adequadament. Sota aquesta premissa, és rellevant recollir dades històriques sobre les condicions de la xarxa vehicular i relacionar-les amb les prestacions corresponents de la VANET. Les dades recopilades ens proporcionen informació útil per a dissenyar noves estratègies basades en prediccions. Amb aquest objectiu en ment, hem avaluat diverses mètriques d'encaminament (distància al destí, densitat de vehicles, ample de banda disponible) que descriuen les condicions de la xarxa quan el vehicle que actualment té el paquet necessita triar el millor següent node per a encaminar el paquet cap al seu destí, seguint un esquema salt-a-salt. S'han realitzat un gran nombre de simulacions sota diferents escenaris representatius per a crear un conjunt de dades a partir dels valors de les mètriques d'encaminament i del valor binari del resultat (entrega del paquet al destí amb èxit o no) associat. Després, hem fet un anàlisi del conjunt de dades fenerat i es va obtenir un model estadístic per a cada mètrica. Aquest model estadístic s'ha utilitzat per a dissenyar un nou algorisme d'encaminament de paquets a VANETs. Amb aquest algorisme, hem proposat un protocol d'encaminament multimètrica basat en probabilitats (ProMRP), que selecciona els nodes candidats en funció de la seva major probabilitat que asseguri el lliurament del paquet al seu destí. A més, s'ha inclós una estimació precisa de la posició del node per obtenir una versió millorada del nostre protocol anomenada EProMRP.

La darrera contribució es centra en l'aplicació de tècniques d'aprenentatge automàtic (*machine learning*) per a millorrar les decisions d'encaminament. Per a això, s'ha recopilat un nou conjunt de dades a partir de cinc mètriques (distància al destí, trajectòria, densitat de vehicles, ample de banda disponible i pèrdues a nivell MAC). El conjunt de dades recollit s'ha utilitzat per generar dos models diferents d'aprenentatge automàtic: (i) un arbre de decisió amb una profunditat màxima de l'arbre igual a 20, i (ii) una xarxa neuronal artificial amb una dimensió 1/3/1 (una capa d'entrada, 3 capes ocultes i una capa de sortida). Les nostres propostes s'anomenen (i) protocol d'encaminament multimètrica predictiu basat en *machine learning* (MPML), i (ii) protocol d'encaminament predictiu multimètics basat en xarxes neuronals artificials (MPANN), respectivament. L'objectiu és generar models d'aprenentatge automàtic amb la predicció exacta més elevada. Les propostes s'han avaluat mitjançant mètriques de rendiment d'aprenentatge automàtic i mitjançant simulacions VANET realistes. S'han considerat diferents escenaris VANET amb diferents mides de xarxa, mapes de ciutats amb orografia i disposició dels carrers diferent, diverses densitats de vehicles, per avaluar cada proposta. També hem avaluat el nivell de flexibilitat de la nostra proposta per a adaptar-se a altres escenaris de ciutats diferents del mapa urbà utilitzat per a entrenar els models. Per a aquest propòsit, MPANN

(que ha mostrat ser la millor proposta) s'ha provat en dos àrees urbanes de ciutats diferents. Amb aquesta darrera avaluació, hem pogut mesurar la capacitat del nostre model d'adaptar-se a noves condicions de xarxa.

Agradecimientos

Me es díficil mencionar tantas personas que de alguna manera han sido participes a lo largo de este proyecto y expresarles mi agradecimiento en unas cuantas líneas. Es imposible incluirlos a todos en una página, sin embargo se debe intentar.

A mi familia, porque siempre están presentes y apoyándome incondicionalmente en todo momento. Les agradezco cada mensaje de apoyo y hacerme sentir acompañada a pesar de la distancia.

A cada uno de los compañeros de laboratorio, que no sólo compartimos la hora del almuerzo y consejos para avanzar en el trabajo, también generamos una buena amistad. Aprendí mucho de ustedes e hicieron que la estancia fuera más agradable. Gracias: Christian, Carlos, Khaled, Ahmed, Víctor, Joseph y Byron. A Pablo, compañero del grupo de investigación, por su valiosa ayuda y colaboración en los trabajos de investigación.

A mis entrañables amigos de México, amigos que a pesar de la distancia estuvieron siempre pendientes y dándome ánimos para seguir adelante. Y a los que por casualidad de la vida, los encontré aquí en Barcelona. Gracias, me siento bendecida por tener tan buenos amigos. A Sonia y Rosario, por todo su apoyo y atenciones. Por tantas risas y deliciosas comidas juntas. Hicieron que siempre me sintiera como en casa y parte de su familia. Las llevo en mi corazón.

A Juan Pablo, por su motivación y apoyo para seguir adelante, aún cuando el camino se tornaba más complejo. Mi vida en Barcelona no hubiese sido lo mismo sin tu compañía.

A Ahmad, por aceptar ser mi codirector. Agradezco todo el apoyo brindado, consejos y por compartir tu valioso conocimiento. Gracias por escucharme y motivarme a creer que esto era posible.

Mi agradecimiento a Mónica Aguilar, directora de tesis, guía y amiga. Por todo su apoyo, paciencia y motivación para llegar a culminar este proyecto de investigación. Sobre todo por permitirme trabajar y aprender de su enorme experiencia. Para mi ha sido un placer trabajar y hacer equipo contigo. Gracias.

Mi agradecimiento a la Coordinación General Académica de la Universidad de Guadalajara, por la beca otorgada para estudios de doctorado. Así también al apoyo recibido y motivación por seguir preparándome que me brindó el Mtro. Gerardo durante su gestión como rector.

Además quiero agradecer a los proyectos INRISCO (TEC2014-54335-C4-1-R) y MAGOS (TEC2017-84197-C4-3-R) que financiaron este trabajo.

También quiero agradecer a los miembros del Tribunal de Tesis que aceptaron revisar el resultado del presente trabajo.

Contents

List of Figures

List of Tables

Acronyms

3GPP	Third generation partnership project
3MRP	Multimedia multimetric map-aware routing protocol
ABE	Available bandwidth
AI	Artificial intelligence
AIFS	Arbitration inter-frame space
AMGRP	AHP-based multimetric geographical routing protocol
ANN	Artificial neuronal network
API	Application programming interfaces
ASM	Attribute selection measures
AUC	Area under the ROC curve
AV	Autonomous vehicle
BM	Beacon message
BS	Base station
BW	Bandwidth
C-V2X	Cellular V2X
CBR	Constant bit rate
CCH	Central control channel
CI	Confidence Interval
CNN	Convolutional neural networks
CSF	Conic section function
CSMA/CA	Carrier sense multiple access with collision avoidance
D2D	Device-to-device
DFF	Deep feed-forward neural networks
DMRS	Demodulation reference signal
DSRC	Dedicated short-range communication
DSW	Dynamic self-configured weights
DT	Decision tree
EDAGF	Estimation and direction aware greedy forwarding
EDCA	Enhanced distributed channel access
EED	End-to-end delay

EProMRP	Enhanced probabilistic multimetric routing protocol
ELU	Exponential linear unit
FN	False negative
FNN	Feed-forward neural network
FP	False positive
GPS	Global position system
GPSR	Greedy perimeter stateless routing
GPU	Graphical processing units
GUI	Graphic user interface
HN	Header node
I-GPSR	Improvement-greedy perimeter stateless routing
ITS	Intelligent transportation system
LDA	Linear discriminant Analysis
LR	Logistic regression
LSTM	Long short term memory network
LTE	Long-term evolution
M-GEDIR	Multi-metric Geographic Routing
M-GPSR	Multi-metric geographic routing
MAC	Medium access control
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MedAE	Median absolute error
ML	Machine learning
ML-CC	Machine learning control congestion
MLP	Multi-layer perceptron
MM-GPSR	Maxduration-minangle GPSR
MPANN	Multimetric predictive artificial neural network
MPML	Multimetric predictive machine learning
MSE	Mean squared error
NED	Network description
OBU	On-board unit
OFDM	Orthogonal frequency division multiplexing
OH	Overhead
OS	Operating system
OSM	Open street maps
P-GPSR	Probabilistic GPSR
PA-GPSR	Path aware GPSR
PCA	Principal component analysis
PDF	Probability density function

PQL	Packet queue length
ProMRP	Probabilistic multimetric routing protocol
QoS	Quality of service
R-FM	Real time forwarding method
RAM	Random access memory
RBF	Radial basis function
ReLu	Rectified linear unit
REVsim	Realistic environment for VANET simulations
RF	Random forest
RL	Reinforcement learning
RNN	Recurrent neural network
ROC	Receiver operator characteristic
RST	Recent send table
RSU	Road side unit
SCH	service channel
SCRP	Stable connected dominating set-based routing protocol
SDN	Software-defined network
SINR	Signal to interference plus noise ratio
SUMO	Simulation of urban mobility
SVM	Support vector machine
SVR	Support vector regression
TN	True negative
TP	True positive
TraCI	Traffic control interface
V2I	Vehicle-to-infrastructure
V2V	Vehicle-to-vehicle
V2X	Vehicle-to-anything
VANET	Vehicular adhoc network
VD	Vehicles' density
VEINS	vehicular network simulator
W-PMF	Weighted power mean function
WAVE	Wireless access in vehicular environment
Wi-Fi	Wireless Fidelity

Chapter 1

Introduction

This chapter is devoted to present a brief introduction to this Ph.D. thesis, and it also points out the main goals of our research work. Besides, the chapter presents an overview of the contents covered throughout this dissertation.

1.1 Overview of the thesis. Motivation

Traffic and congestion caused by vehicles affect all urban centers, and their impact on city life will continue to increase up as cities grow. In effect, the United Nations (UN) estimates that more than 60% of the world population will live in cities by 2050 [10]. It means that a large portion of these cities might probably increase the use of particular vehicles, making traffic management a really complex task. Traffic congestion reduces the efficiency of transport infrastructure and increases travel time, fuel consumption, and environmental pollution. Therefore, there is a great need for effective traffic management and congestion solutions, not only to make mobility more efficient but also to reduce collateral damages such as pollution and health problems.

In this sense, mobility is a primary concern for cities striving to become smarter and more sustainable. Smart cities are assumed to collect big data from different computer devices and the Internet of things (IoT), to improve city services. In recent years, the implementation of technologies to make smart mobility have shown that they can help to reduce the cost of traffic and congestion in the environment, and moreover improve health and quality of life of citizens [11].

Combining collected traffic data from diverse physical hardware (e.g., sensors, antennas, small board computers) with prediction mechanisms help the intelligent transportation system (ITS) to improve traffic management and to minimize the environmental impact. Those mechanisms increase the benefits of transportation for all kinds of users, especially in urban areas. In fact, vehicular ad-hoc networks (VANETs) support ITS to improve drivers' safety and traffic efficiency on the road by exchanging traffic-related information between vehicles and also between vehicles and infrastructure [12, 13]. Consequently, routing protocols designed for VANETs should be flexible and able to adapt to the inherent dynamic network characteristics of this kind of networks.

In the last years, there is an interesting trend so that applications of many services are supported by information provided by large amount data. The applications of smart city data are still emerging. The amount of progress made so far and the potential for the future is immensely hopeful [14]. We believe that it is possible to take advantage of data analysis and apply the knowledge to improve the design of VANET routing protocols.

Considering the previous generic goals, the context of his thesis aims to **contribute to the design of VANET routing protocols to enable smart mechanisms to report traffic-related messages in urban environments**.

1.2 Objectives of the thesis

The main objective of this thesis is to contribute and find solutions to some open problems present in vehicular ad-hoc networks (VANETs) under urban environments. A vehicular multi-hop communication requires simple and efficient forwarding mechanisms that can work with limited and local topology information. Besides, it has to adapt rapidly to the fast changes in the network. With this in mind, **we aim to design a routing protocol able to adapt to the dynamic network conditions, making decisions based on prior knowledge of possible input events by detecting the current state of the network**. In this way, the forwarding of information is more efficient. To achieve this main objective, we have organized our tasks in three parts:

(i). We have studied and analyzed **different ways of averaging a set of parameters (routing metrics) used in multi-hop forwarding algorithms**. We have tested systematically the average power-mean function in the combination of metrics used in a previous proposal of VANET routing protocols named multimedia multimetric map-aware routing protocol (3MRP) [4] [15], which was previously developed by the directors of this thesis. Additionally, we have improved the estimation of the current distance of the forwarding candidate nodes to destination, based on the information received from the last beacon message, i.e. (x, y) and (v_x, v_y), for each neighbouring node. This more accurate position at the forwarding moment will outperform the adaptation of the routing protocols under fast changes of network topology.

(ii). Then, we have focused our research on the design of a probabilistic distribution model able to **estimate the probability of packet successfully delivered at destination, which will be used to improve the forwarding routing decisions**. For this purpose, we have done an offline data-analysis to build an accurate probabilistic model for each one of the considered routing metrics.

(iii). Finally, we have tackled the key issue of **flexible adaptation to enhance routing protocols for VANETs**. We have designed three **machine learning (ML) based forwarding algorithms that work using prior knowledge of possible**

events. For this purpose, we have built a representative dataset with the designed features gathered from many vehicular simulations. Hence, the supervised learning models designed have been trained and tested. Each trained model has been validated with performance evaluation metrics. An exhaustive performance using simulations shows the benefits of our proposals.

1.3 Summary of contributions

The main contributions of this thesis are described below:

- We have **improved the way to combine several routing metrics used to calculate a multimetric score**. This multimetric score is used to arrange the candidate nodes (i.e., those neighbours of the node currently holding the packet) to forward a packet. We have used the multimedia multimetric map-aware routing protocol (3MRP) [4] [15] previously developed by the directors of this thesis, as the starting point over which we have tested our proposals. Our goal was to improve the 3MRP's performance in urban environments. Firstly, a better way of combining routing metrics is proposed, suitable when routing forwarding decisions are based on averaging some node's metrics. With this strategy, we will see that the routing protocol can more efficiently evaluate the candidate nodes and choose the next hop forwarding node.

- In most geographic routing protocols for VANETs, the information of the position network nodes is updated at each reception of the signaling *hello messages*, also known as *beacons*. This information is used to select the route (in a hop-by-hop fashion) through which the message is forwarded. We have developed a mechanism to **accurately estimate the position of candidate nodes in the moment of sending a packet**. With this mechanism, the forwarding node can accurately estimate the current positions of its neighboring nodes to make the best routing decision in that moment.

- We have developed a **probabilistic distribution model used to take forwarding decisions that guarantee the packet delivery at destination with higher probability**, specially designed for urban environments. This probabilistic mechanism is based on the modeling of the probability density functions of three routing metrics. Such modeling was done based on a previous off-line analysis of a significant number of representative simulations performed in urban scenarios. Besides, by including an accurate estimation of the nodes' positions in the proposed mechanism, the forwarding routing decision is adapted to the current network conditions.

- Finally, we have proposed a **multimetric routing protocol based on machine learning (ML) predictions**. The designed routing mechanism includes a set of phases: (i) building a dataset composed by collected traffic values, (ii) data prepossessing, (iii) training the ML model, and (iv) validation of the trained model. The proposed mechanism makes predictions based on five different routing metrics (distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses) that describe the current conditions of the VANET nodes. Therefore, the node currently holding the packet can predict which of its neighboring nodes can successfully deliver the packet at destination with higher probability. This way, the node can make the best forwarding decision. An important point to highlight is the benefit obtained by using artificial neural networks in the design of our ML-based forwarding algorithm. The improvement in the efficiency of the selection of the forwarding nodes hop-by-hop is notable, especially in critical network conditions when the vehicles' density is very low or very high.

1.4 Outline of this thesis

This thesis is organized in eighth chapters. In this section, we briefly present the contents of each one of the chapters.

Chapter 2 describes some **important aspects of vehicular network implementations in urban environments**. We highlight the main features and technologies used in VANETs and some of their applications. Besides, we describe the **simulator tools and metrics** used to carry out the performance evaluation of our proposals.

The most representative **geographic routing protocols for VANETs** are described in Chapter 3. We have started our research work from the analysis of the **multimedia multimetric map-aware routing protocol (3MRP)** [4], previously designed by the directors of this thesis [15]. We have used 3MRP as a basis over which we have developed our contributions.

Chapter 4 proposes two contributions to enhance the forwarding decision algorithm included 3MRP [4] protocol. On the one hand, we have found which is the **best strategy to combine different routing metrics to compute a multimetric score used to arrange the neighbouring nodes** (i.e., vehicles within the transmission range of the vehicle currently holding the packet) that are candidates to be the next forwarding node. We have improved the selection of the best neighbouring vehicle to be the next-hop to forward the packet towards its destination. Also, we have proposed an **accurate estimation of the node positions at the current moment to forward a message**.

Chapter 5 presents a **new multimetric routing protocol for VANETs based on the analysis, for each candidate node, of the probability of packet-delivery success at destination**. The goal is to select the best next-hop candidate within transmission range. We have explained the methodology used to model the probability of each routing metric based on a collected dataset. The proposal has been evaluated and compared with other protocols through network simulations. To this end, different vehicles' densities have been considered, from sparse to congested VANETs. This way, our data set is as generic as possible, including data from diverse network configurations.

Chapter 6 provides an **overview of basic machine learning knowledge**, focusing on the classification of classical learning and on the learning phases that a machine learning model follows. Specifically, this chapter is focused on decision tree and artificial neural network models. Besides, the most relevant performance metrics used to evaluate ML models are presented.

In Chapter 7, a **new routing protocol based on machine learning model predictions is proposed**. We have used five routing metrics (distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses) as the inputs to build a dataset, whereas the output is the binary certification of successfully arrival or not of the packet at destination. For this purpose, we have applied the learning process phases described in chapter 6 over these collected data. As a result, we have generated three different machine learning forwarding algorithms. Their performance has been evaluated through machine learning evaluation metrics. Also, they have been implemented over the 3MRP routing protocol and have been evaluated through extensive network simulations.

Finally, conclusions, publications generated from this thesis, and some future work guidelines are pointed out in Chapter 8.

Chapter 2

Vehicular networks in urban scenarios

In this chapter we describe the most important aspects that matter to the design of routing protocols for vehicular networks in urban scenarios. We highlight the main goals and applications intended for VANETs. Also, we describe the most important features and the technologies used in VANETs. Then, we follow with the description of the simulators that we have used in this thesis work to carry out the performance evaluation of our proposals.

2.1 Introduction

Vehicular ad hoc networks (VANETs) are nowadays a well-known type of telecommunications networks where nodes are vehicles. VANETs are considered as a promising ad hoc technology for real-life applications that combine both intelligent transport systems (ITS) and smart cities [12], specifically in urban scenarios. The main motivation of VANETs is to provide safety and security for citizens by sharing information about events such as accidents or traffic state. In the last decade, many research works with proposals to improve the VANET operation have been published. So we can state that today VANETs are considered a sufficiently mature technology [16]. In a VANET, vehicles are equipped with a set of wireless sensors and on-board units (OBUs) to allow wireless communications between vehicles and their neighbouring nodes [17]. These inter-vehicle communications are also known as vehicle-to-vehicle (V2V) communications. Besides, vehicles can communicate to other networks deployed in the city trough vehicle-to-infrastructure (V2I) communications. Vehicles can operate as packet senders, receivers as well as routers to forward messages to other vehicles or to road side units (RSUs). Vehicles can directly send packets to other nodes within their wireless transmission range; otherwise, they need relaying nodes to forward packets following a multi-hop path.

Recently, a global term called vehicle-to-anything (V2X) communications has been introduced by the third generation partnership project (3GPP) group [18]. V2X communications allow vehicles to interchange information with different elements of the intelligent transportation system (ITS), such as other vehicles, pedestrians, Internet gateways, and other transport infrastructures.

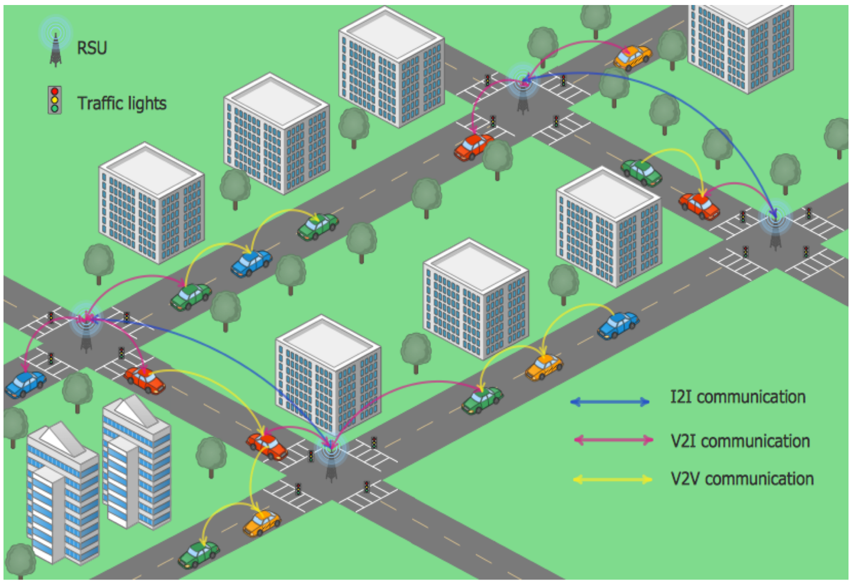


FIGURE 2.1: Typical VANET urban scenario.

In this dissertation we are interested to contribute in the design of vehicular communications in urban scenarios like the one depicted in Fig. 2.1. For this reason, the main contributions are based on real maps from European cities.

2.1.1 Mean features of vehicular networks

Vehicular ad hoc networks is a well-known type of mobile ad hoc network (MANET) where nodes are vehicles. Vehicles include several sensor devices and on-board units (OBUs) that allow them to communicate. In addition, there are other network components such as road side units (RSUs) spread along roads and city streets (see Fig. 2.1). VANETs can be characterized by these inherent features:

- Dynamic topology. Speed and direction of the vehicles might change frequently, which produce a very dynamic network topology.
- Unstable connectivity. This is a consequence of the frequent changes on the network topology. Connectivity between vehicles might change very quickly since vehicles can change their relative speed or change their route. it is more likely to happen in urban scenarios.
- Mobility patterns. Generally, vehicles follow patterns which describe their movements along the roads and streets. Besides, they respect traffic signals (traffics lights, speed limits, street direction, etc) that condition their behavior on the network.

- Communication architecture. The communication between the different VANET nodes can be classified according to the type of network devices involved and their scope.
  - *Vehicle-to-vehicle (V2V).* Communication to exchange data between vehicles and assist drivers about the own vehicle system.
  - *Vehicle-to-infrastructure (V2I)* Communication between vehicles and roadside fixed infrastructure.

2.1.2 Technologies employed in vehicular networks

Vehicular ad hoc networks can use any wireless networking technologies, which are based on the dedicated short-range communications (DSRC) global standard. Also, long-term evolution (LTE) [19] cellular technologies can be used for VANETs, given vehicles can communicate directly among them and also to the infrastructure. In the following we briefly describe the main aspects of the standards and technologies used in VANETs.

IEEE 802.11p standard for vehicular ad hoc networks (VANETs)

IEEE 802.11p is the amendment to the IEEE 802.11 standard that aims to provide wireless access in vehicular environments (WAVE) as vehicular communication system, see Fig. 2.2. It defines enhancements to the basis IEEE 802.11 required to support intelligent transportation systems (ITS) applications. The standard details how to perform the data exchange between high-speed vehicles and between vehicles and roadside units (RSU) as the infrastructure.

This kind of ad hoc communication does not require any infrastructure to operate and nodes themselves manage the network communications in a distributed fashion. The IEEE 802.11p standard describes the requirements to support ITS applications. More specifically, the WAVE protocol stack focuses on the MAC and network layers described in the set IEEE P1609.x standard suite [20].

To promote communications in vehicular networks, a dedicated spectrum band of 5.9 GHz (5.85–5.925 GHz) was allocated to these technologies. A bandwidth of 50MHz was assigned in Europe to handle vehicular communications, while in America a bandwidth of 75 MHz was reserved. To manage this reserved spectrum range, it is divided into small 10MHz channels, one dedicated central control channel (CCH) reserved for the system and the rest are devoted to service channels (SCHs) to non-safety applications.

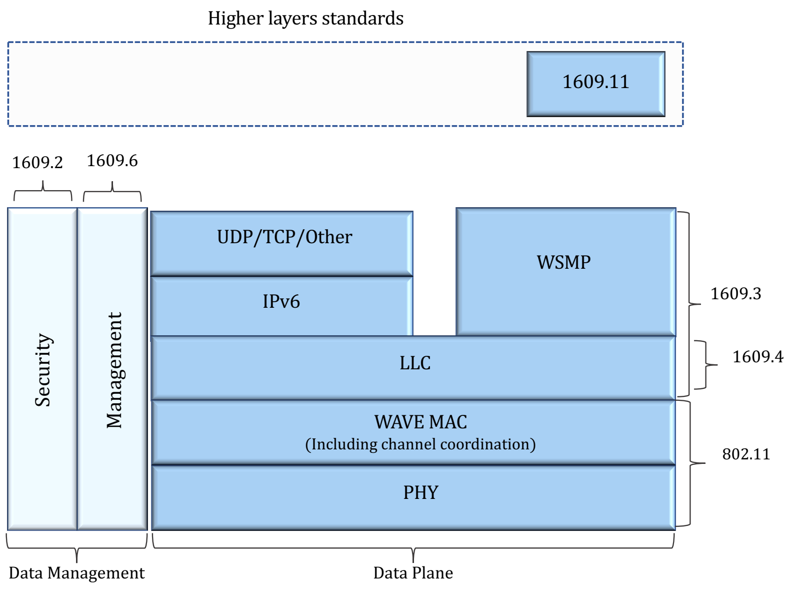


FIGURE 2.2: WAVE protocol stack.

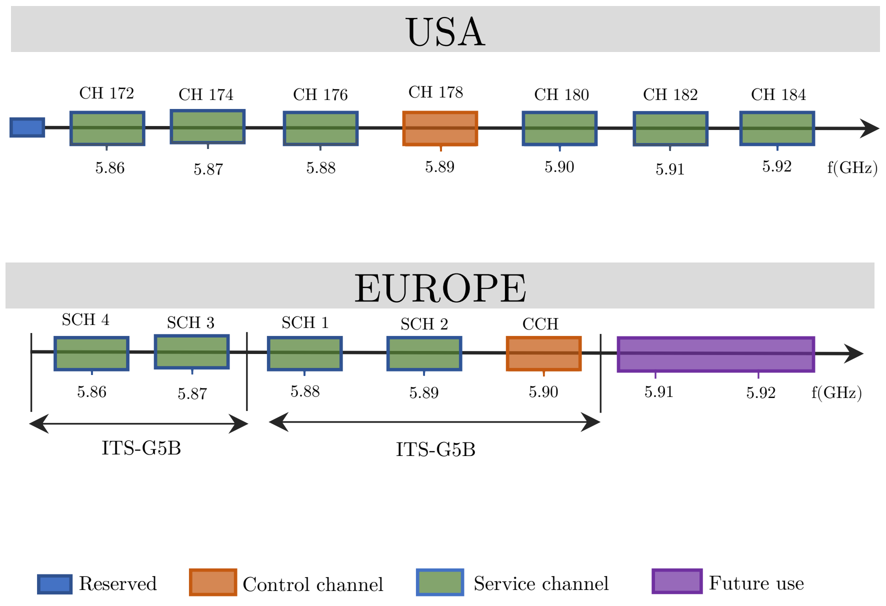


FIGURE 2.3: Spectrum allocation to vehicular communications in both operational regions: USA and Europe. (image adapted from [1]).

As a result, there are 7 different channels for IEEE WAVE operation and 5 for the case of the European standard ETSI ITS-G5. In Europe, 30 MHz (3 channels) are reserved for road safety in the ITS-G5A band, and 20 MHz are assigned for general purpose ITS services in the ITS-G5B band [1]. There are supported two operation modes, V2V and V2I.

The DSRC/C-ITS standard uses the orthogonal frequency division multiplexing (OFDM) scheme at the PHY layer. In order to reduce the 20 MHz channel spacing to 10MHz and adapt to the inter-carrier interference due to the high speed of the vehicles, the OFDM scheme operates with half clock. At the MAC layer, ITS-C5 employs enhanced distributed channel access (EDCA) with CSMA/CA and access categories for data traffic prioritization (see Fig. 2.3) [21].

Cellular vehicle-to-everything (C-V2X)

C-V2X is an alternative to IEEE 802.11p that also provides device-to-device (D2D) communication within legacy cellular-based LTE radio access networks. The 3GPP group published the term "cellular V2X" (C-V2X) based on LTE technology, specifying the requirements to make possible V2V communication. Basically, the enhancements are made in the LTE-broadcast and LTE-Direct operation modes. LTE-broadcast will facilitate V2I and V2N communications using the cellular infrastructure. The main benefits of enabling this kind of communications makes it possible a wide range of applications. Specially, C-V2X is able to increase the coverage area to a few miles to report warning messages. LTE-Direct enables V2V connectivity at distances up to hundreds of meters at very low latency (1 millisecond) with respect to IEEE 802.11p V2V communication [22].

- Main novelties included in C-V2X technology to enhance V2V communications:
  - A new D2D interface (PC5, or side link at the physical layer) was introduced and it has been enhanced for vehicular use cases, specifically addressing high speeds (up to 250Kph) and high densities.
  - Demodulation reference signal (DMRS) symbols have been added in V2V subframe to handle the high Doppler effect associated with relative speeds of up to 500kph and at high frequency. It allows a better tracking of the channel at high speed.
  - There are two high level deployment configurations, aiming to accomplish the V2V delay requirements. Configuration 1: scheduling and interference management of V2V traffic is supported based on distributed algorithms (Mode 4) implemented between the vehicles. Configuration 2: scheduling and interference management of V2V traffic is assisted by the LTE base station, or eNodeB (eNB) (Mode 3) via control signaling over an specific Uu interface (See Fig. 2.4).

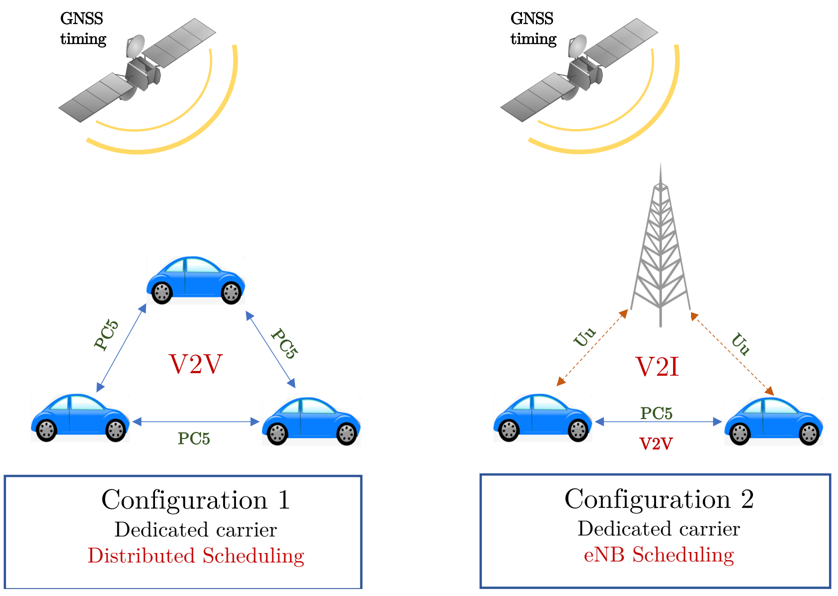


FIGURE 2.4: Deployment configurations in C-V2X communications (images adapted from [2]).

2.2 Applications and services for vehicular networks

In this section the main VANET applications will be briefly described and summarized: (i) safety applications, (ii) traffic management applications and (iii) comfort driving and infotainment applications.

2.2.1 Safety applications

Avoiding risks of accidents is the main motivation to develop VANETs. In this sense, safety applications seek to avoid accidents to have safer driving by distributing information to drivers about hazards and obstacles. Sometimes, accidents cannot be avoided, but their impact can be minimized with on-time information. Basically, the idea is to warn other drivers around the road situation with emergency messages sent through wireless communication. This way, drivers can faster react and take proper maneuvers.

Safety applications require to manage information collected from sensor devices or from the interchanged messages between vehicles. Some examples of safety applications are:

- Cooperative and intersection collision warning: The goal is to warn drivers of a possible collision with the vehicle ahead of them, due to a sudden decrease in speed due to a stop, collision, or a sharp curve in the road.
- Emergency vehicle warning: In case there is an emergency vehicle, like a police car or an ambulance, the goal is to temporarily reserve a line for them on the road.

This is a kind of cooperative V2V communications and also V2I communications where RSUs cooperate to spread the warning messages.

- Line change warning. The goal is to warn the driver when a lane change occurs in their blind spot area. This kind of alert reduces the chance of accident in line-change scenarios.

Safety application are strictly delay sensitive. Thus, a fraction of time is very important in decision making. Therefore, warning messages are charged into the MAC layer to be delivered with minimum delay [23].

2.2.2 Traffic management applications

This kind of VANET applications focus on optimizing flows of vehicles in order to reduce trip times and to avoid traffic jam situations. The scope of traffic management applications is not only to optimize routes, but also to reduce gas emissions and fuel consumption. [24]. Most of the traffic management applications use the roadside infrastructure in order to collect and disseminate information. Examples of such infrastructure are intelligent traffic lights and automatic toll payment, among other many applications:

- Intelligent traffic lights. They can be adjusted in response to the traffic conditions at intersection and can even communicate the traffic state to neighbouring intersections. Neighbouring intersections can display this information on the e-sign boards and adjust their traffic lights according to the traffic state. Also, emergency vehicles could use a green-wave of traffic lights to drive quickly and safely at intersections.
- Automatic toll payment. Vehicles do not need to stop to pay toll fees as they communicate with the road infrastructure to be recognized and the fee is automatically charged to their account.

Also, it would be possible that some road infrastructure could be free to use, while others would need a user subscription [25]. This way, some roads in the cities could be allowed only for non-pollutant vehicles.

2.2.3 Comfort driving and infotainment applications

In this type of application, vehicles get Internet connection to access networks or servers where the information of interest is stored or to require a specific interest. Vehicles can access Internet through road side units (RSUs) or through other infrastructure (e.g., access points). There are considered as non-safety applications. Examples are traffic management and infotainment applications.

VANETs can be used for advertisements or announcements of location-based sales information. For example, gas stations can announce prices, or restaurants can highlight different offers to attract drivers.

Infotainment applications include sharing services among vehicles (V2V communication) or Internet access through the infrastructure (V2I communication). Examples of such services are sharing multimedia files and video-streaming [25].

2.3 VANET simulator platform used in this thesis

The simulation of VANETs is a useful tool to analyze and evaluate the design of proposals to improve intelligent transport systems before implementation. The level of realism and viability depends on the type of the realistic mobility models used and on the appropriate metrics to evaluate the benefits of the proposals [26]. In this sense, one of the initial objectives in this dissertation was to implement our proposals in realistic simulation scenarios. Thus, we selected a special set of simulation tools able to work together to attain a reliable analysis of the results. Each one of the tools that compose our simulation framework is described below.

2.3.1 Vehicles in network simulation (Veins)

Vehicles in network simulator (Veins) [27] is an open source vehicular network simulator framework used to simulate vehicular communications. This simulator includes a comprehensive suite of models, devices and protocol stack that allow researchers to carry out realistic vehicular network simulations. In our simulation framework, Veins works together with the traffic road simulator SUMO and with the event-based network simulator OMNeT++, described in the next sections.

2.3.2 OMNeT++ simulator

OMNeT++ [28] is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. This simulator works under an Eclipse-based IDE, a graphical runtime environment, and a host of other tools. It has another extensions for real-time simulation, network emulation, database integration, among other functions.

OMNeT++ has its own component architecture for models that communicate with message interchange. Components (modules) are programmed in C++, then assembled into larger components and models using a high-level language network description (NED). The NED language is used to define the structure of the model (modules and their interconnections), i.e., module declarations, compound module definitions and network definitions. Basically, OMNeT++ includes these components:

- Simulation kernel library (C++)
- The NED topology description language

- Simulation IDE based on the Eclipse platform simulation runtime GUI (*Qtenv*)
- Command-line interface for simulation execution (*Cmdenv*)
- Utilities (makefile creation tool, etc.)
- Documentation, sample simulations, etc.

OMNeT++ is an adaptable network simulator, it can run basically on all platforms where a C++ compiler is available. Furthermore, it counts on a designer community which constantly includes new capacities and components to simulate new technologies. OMNeT++ was designed to be as much general as possible, being adaptable to different systems. The OMNeT++ simulation kernel is standard C++, and it runs basically on all platforms where a modern C++ compiler is available. The simulation IDE requires Windows, Linux, or MacOS [29].

2.3.3 Simulation of Urban MObility (SUMO)

Simulation of urban mobility (SUMO) [9], is an open source software licensed under the GNU/Linux general public license (version 2) and allows high performance simulations of multi-modal traffic in city scale network. It is highly portable and offers a download platform that is constantly updated by its community of designers. SUMO allows to characterize how a given traffic of vehicles moves through a given road network. The simulation allows to address a large set of traffic management topics: traffic lights evaluation, route choice and re-routing, simulation vehicular communications, traffic forecast, among others.

One interesting characteristic of SUMO is its traffic control interface (TraCI) tool. TraCI allows users the interconnection of external control functions to SUMO (connection with others tools). TraCI provides flexible open application programming interfaces (APIs) to retrieve metrics (e.g., for combining them with network metrics into ITS-specific metrics) and to control the SUMO parameters, from the mobility model, road network or to manage traffic lights [30].

We use the SUMO simulator to generate urban scenarios from real maps and also to generate random routes of different vehicles' densities. These are the SUMO tools used to carry out our simulations:

- NETCONVERT. Network importer and generator. This tool reads road networks from different map formats and converts them into SUMO-format (network file).
- DUAROUTER. Computes fastest routes through the network, importing different types of demand description.
- POLYCONVERT. Imports points of interest and polygons from different formats and translates them into a description that may be visualized by the graphic user interface (SUMO-GUI).

2.3.4 OpenStreetMaps

Open Street Maps (OSM) [5] is a collaborative project to create free and editable maps. The maps are created using geographic information captured with mobile GPS devices, and diverse free sources. OSM is not only a set of free-ware maps. Actually, in OSM the data generated by the project includes many realistic features of the environment: road networks, streets, buildings, points of interest, etc. In addition to being free to use, OSM data is accurate and updated, since all the information added to this platform is continuously verified by the community of OSM collaborators. We use OSM maps to generate and create realistic urban scenarios used in our simulation framework OMNeT++/Veins/SUMO/OSM to assess the performance evaluation of our proposals.

2.4 Metrics used to carry out the performance evaluation of our proposals

We have performed extensive simulations with the aim of measuring the effectiveness of each one of our contributions to enhance VANETs. The simulations were performed to obtain results in terms of some quantitative metrics that are used to assess the performance of our proposals of routing protocols for VANETs. In the following we describe the main metrics that we have used to assess the benefits of our proposals.

2.4.1 Average packet losses

In wireless networks packet losses occur when one or more packets travelling across the network fail to reach a destination. Packet losses can occur due to several factors, such as radio interference, unstable channels, mobility of nodes or network congestion. In order to know if the strategy to select the path hop-by-hop to each packet that is transmitted in the network reach a destination, this value of losses is quantify in percentage. That is, those losses are measured as a percentage of packets lost with respect to packets sent. Average packet loss can be calculate:

$$\frac{Total\ number\ of\ packets\ sent - total\ number\ of\ packets\ received}{total\ number\ of\ packets\ sent} \tag{2.1}$$

2.4.2 Average end-to-end packet delay

Average end-to-end delay (in short EED) in a VANET can be described as the delay experienced by the successfully delivered packets that reached their destination. EED is a basic metric used to compare routing protocols. Since the delay primarily depends on how optimal were the used paths, it can denote how efficient the routing algorithm is.

$$Average\ EED = \frac{1}{N}\sum_{i=1}^{N}(r_{x,i} - t_{x,i}) \tag{2.2}$$

where N is the number of packets received successfully, $r_{x,i}$ is the time at which each packet i is received and $t_{x,i}$ is the time at which the packet was sent.

2.4.3 Overhead

The overhead (in short OH) is another numeric metric normally used to evaluate the routing performance. By the measurement of the routing overhead it is possible to describe the level of efficiency of a routing protocol. We measure the OH considering:

- Total number of bits transmitted to be able to deliver an amount of data bits. The relation between both numbers gives us an idea of the bit efficiency of the routing protocol and the network.
- To compute the control overhead, we include any bit transmitted that is not data (at the network layer).

The overhead can be defined as the ratio between additional routing packets and received packets at the intended destinations. It can be calculated as:

$$OH = \frac{Total\ number\ of\ overhead\ messages\ (bytes)}{Total\ transmitted\ packets\ (bytes)} \tag{2.3}$$

2.4.4 Computational cost and simulation time $t(n)$

An algorithm may be considered good if it achieves the target objectives. However, it may not be efficient in terms of the number of computational resources and computational time that it requires to be executed. Therefore, the algorithm must accomplish its designed objectives while keeping bounded the amount of resources consumed. The complexity of an algorithm allows researchers to assess the trade-off between benefits of the proposal and amount of resources needed to achieve that outcome.

The measurement of the computational cost is obtained calculating the amount of resources that are needed by the algorithm in the worst-case. In other words, we assess the highest amount of resources needed to run the algorithm. The complexity of an algorithm can also be measured as a function of the time that the algorithm requires to obtain the output for an input of a given size (n). This metric is known as computational cost of an algorithm.

To assess the efficiency of an algorithm, we measure the time it takes for the algorithm to execute the series of operations and statements when executing in the worst-case scenario.

That is, in the case when the size of the inputs is larger and therefore n increases. The worst case in terms of computational cost will be determined by the dominant term, and the coefficients are ignored. Let us see an example to clarify that definition. Suppose again the previous example:

$$3n^2 + 5n - 7 \; we \; can \; write \rightarrow t(n) = 3n \cdot n + 5 \cdot n - 7 \rightarrow t(n) = n \cdot n \tag{2.4}$$

The coefficient (3 in the example) that is applied to the most significant term in $t(n)$ is not considered to measure the cost. It does not affect how the worst-case time grows with n (input size) but only the units in which we measure the worst-case time. Thus, in this example $t(n)$ grows like $n \cdot n$ as n increases. Therefore, the higher time spent by the algorithm will be in the most significant term, in this case, it will be in the order of n^2.

Sometimes it is important to know how long does an algorithm takes to be executed by a computer. This information can be relevant to relate the time needed to execute an algorithm (e.g., included in the routing protocol) with the total simulation time. One way is to test the set of algorithm steps using a specific computer and measure the time spent. That is, we measure the time spent to carry out the operations in the algorithm for the specific CPU and available hardware (computer and compiler being used). An example of the cost in time for an CPU to execute the example of equation (2.4) is presented in Table 2.1, where a computer with an processor Intel Core i7 base clock frequency of 2.9 GHz was used.

$n = 10$	$n = 100000$	$n = 1000000000$
$tic; \leftarrow input$ $i = 3 \cdot n^2 + 5 \cdot n - 7;$ $t = toc; \leftarrow output$ $t(n) = 10.000182 \; seconds$	$tic; \leftarrow input$ $i = 3 \cdot n^2 + 5 \cdot n - 7;$ $t = toc; \leftarrow output$ $t(n) = 0.000185 \; seconds$	$tic; \leftarrow input$ $i = 3 \cdot n^2 + 5 \cdot n - 7;$ $t = toc; \leftarrow output$ $t(n) = 0.000241 \; seconds$

Table 2.1: An example of computational cost $t(n)$ when n increases. A computer with an CPU of 2.9 GHz Intel Core i7 was used.

In this last section, we have seen a way traditionally used to analyze the efficiency of the algorithms: the computational cost and simulation time $t(n)$. There are other methods for the analysis of algorithms that emerge from the field of computer science and an extensive available theory that deals with the subject.

It is important to measure this kind of metrics to evaluate the trade-off between the benefits (e.g., reducing losses and delay) of novel proposals and the cost incurred in terms of (i) overhead, (ii) computational cost and (iii) simulation time.

2.5 Conclusions

In this chapter we have presented the main characteristics of VANETs, emphasizing on the technologies used and the main applications. Without a doubt, vehicular networks are a key in the development of intelligent transport systems. Many applications are focused on safe driving, which are based in V2V and V2X communications. Besides, VANETs have the potential to drastically improve the driver's quality of life.

VANETs must cope with the inherent features of urban environment, where nodes might find infrastructure obstacles such as traffic lights, buildings or road junctions, which decrease the channel quality and worsen connectivity. All these characteristics cause dynamic changes in the network topology. Thus, the design of proper routing protocols for VANETs becomes a challenging goal. Routing on VANETs has to deal with mobility and scalability issues, while also a satisfactory network performance should be provided.

Each new design needs to be previously evaluated based on its performance and efficiency, considering metrics such as average packet losses, average packet delay, overhead, among others. Also, to asses the trade-off of the benefits to the costs, we need to compute also the overhead, the computational cost and the simulation time of our proposals.

To carry out the performance evaluation of our proposals, the use of simulation tools that are capable of simulating vehicle network simulators are used. For that, in this thesis a set of simulators were selected according to their capacity to generate vehicular behaviour (veins), vehicular traffic routes into real urban maps (SUMO) and to manage the simulation events and collect the results (OMNeT++) [28]. Besides, we also use real maps taken from OpenStreetMaps (OSM) [5].

In the next chapter, we present the main routing protocols designed for VANETs, describing their main contribution to forward messages in vehicular environments. Also, some of the features of these protocols are the starting point on which we base our novel contributions.

Chapter 3

Overview of routing protocols for VANETs

This chapter discusses a brief overview of the most representative geographic routing protocols for VANETs in the literature.

3.1 Introduction

The transmission of information through VANETs is a difficult task due to the inherent characteristics of this type of wireless networks. VANET nodes (i.e., vehicles) move at high speeds, which varies the network topology rapidly and, therefore, produces very dynamic link connectivity. Consequently, routing protocols designed for VANETs should be flexible and able to adapt to the dynamic network characteristic. In addition, they must fulfil the services' requirements. Specifically, safety applications are highly demanding and they require low packet delays and low packet losses.

During the last decade many works have been published about the design of routing protocols which try to cope with the VANET issues. These works show that the routing protocols that are more appropriate for vehicular networks are those which consider the node's position to take forwarding decisions [16]. These kind of routing protocols are known as position-based or geographic [31]. In the next section, some of the most representative proposals are briefly explained.

3.2 Geographic routing protocols for VANETs

In geographic routing protocols for VANETs, communication and message dissemination is made by nodes themselves using position information of neighbouring nodes. Vehicles are assumed to know their geographical position and the positions of their neighbours. Geographic routing protocols do not need to establish nor maintain any end-to-end route from source to destination, which is suitable to the dynamic environment of VANETs.

3.2.1 Greedy perimeter stateless routing (GPSR)

One of the first geographical routing protocols adapted for VANETs, and probably the most used to compare with, is the greedy perimeter stateless routing (GPSR) [3] protocol. This protocol deploys the shortest distance to destination as the only metric considered to forward packets. In GPSR the position of a packet's destination and the positions of the candidate next hops are enough to take forwarding decisions, without any other topological information. The GPSR algorithm consists in two methods to forward messages: a) greedy forwarding as the first operation mode; and b) perimeter greedy forwarding mode used when greedy forwarding can not be used.

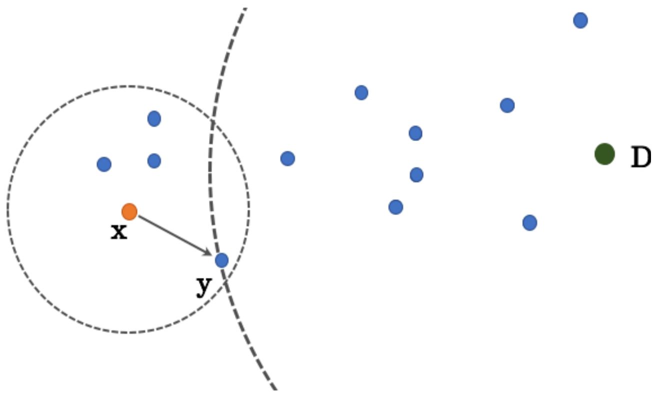


Figure 3.1: Greedy forwarding example, where y is x's closest neighbour towards D. Image from [3].

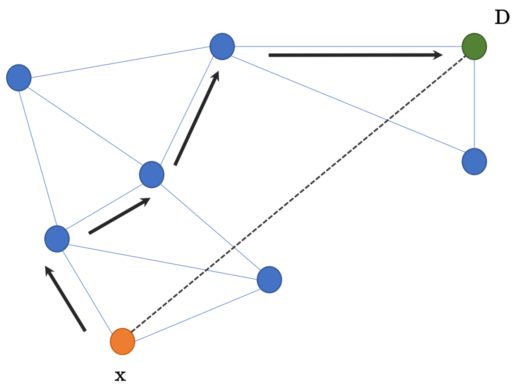


Figure 3.2: Perimeter forwarding example, where x is the node from which the packet is forwarded towards D according to perimeter mode. Image from [3].

- **Greedy forwarding**
  Each node periodically transmits beacons to the broadcast MAC address appending its own identifier and position. In this way, nodes will know the positions of their neighbours, i.e., nodes within their transmission range. The local optimal choice for the next hop is the neighbour geographically closest to the packet's destination. This forwarding algorithm is recursively repeated until the packet reaches its destination (see Fig 3.1).

- **Perimeter forwarding**
  The perimeter forwarding mode is activated when the greedy forwarding fails. In some cases there are sparse topologies in which the only route to destination requires a packet to move temporarily farther in geometric distance from destination. As a solution, the source node implements a hand-rule method to select the next-hop forwarding node, surrounding the empty area between source and destination nodes. In this way, it is more possible that the packet can reach the final destination (see Fig 3.2).

However, GPSR has some inconveniences: The best next forwarding node is the nearest node to the destination, but sometimes this neighbour node maybe into the edge of the communication range. Thus, due to the high speed of the VANET nodes, the position of the selected node may change before it received the packet. Also, large delays might happen due to the perimeter forwarding mode, making this protocol not suitable for urban scenarios where a lot of disruptions usually take place.

3.2.1.1 Improvements of GPSR

Some works were developed during the last decade with the aim to improve GPSR. Here we include some examples of proposals that improve the GPSR operation modes, specially to avoid the problematic perimeter mode. Specifically, new proposals consider more metrics to select the next-hop forwarding node.

- Improvement GPSR (I-GPSR). To improve the GPSR issues, [32] includes four metrics to select the next forwarding node in its neighbourhood: (i) distance to destination, (ii) vehicle's speed, (iii) vehicle's density and (iv) moving direction. I-GPSR gives more priority to those nodes with higher speeds and moving direction towards destination.

- Probabilistic GPSR (P-GPSR). In [33] P-GPSR propose a method to select relaying vehicles using additional information shared by the nodes in their hello messages. It calculates and weights three metrics to take the forwarding decision: (i) link quality estimation, (ii) probabilistic reception of the packet and (iii) moving direction.

- Greedy simplified perimeter routing with moving vector (GSPR-MV). Under greedy forwarding operation, nodes consider the neighbouring nodes' speeds and movement directions with respect to the source node. Every time that a node receives a beacon from a node within transmission range, it calculates the speed and movement direction of that neighbor node. The neighbor node with the less value of distance to destination and less angle in the horizontal direction with respect to destination, will be selected as next forwarding node. Under perimeter forwarding operation, to try to avoid redundant paths, nodes save a few status information (e.g., the identification of the last hop for that packet) to help to detect whether the packet has already been forwarded by the node before [34].

- Maxduration-minangle GPSR (MM-GPSR). This protocol aims to improve the two GPSR operation modes. Regarding greedy forwarding mode, they keep the same logic (select the neighbour with the shortest distance towards destination) and additionally they consider two new parameters: (i) communication area with respect to a source-to-destination area, named *allowed communication area* (Q), and (ii) the estimation time that a node will be in a specific communication area, named *accumulative duration time* (T). In this way, the best forwarding node will be the candidate with the highest T value and the Q area with the lowest angle value. That is, the selected node will be the one that remains longer in that area Q and is also more aligned in the direction to destination. Regarding perimeter forwarding mode, they define a parameter θ to classify neighbours. The sender node in turn calculates and compares the angles of those neighbours that are closer (than the current sender node itself) to destination. After that, the node selects the neighbour with the lowest angle θ as the next-hop forwarding node, which is the closest node to destination [35].

3.2.2 Multimedia multi-metric map-aware routing protocol (3MRP)

Multimedia multi-metric map-aware routing protocol (3MRP) [4] is a geographic routing protocol that uses a multi-metric score scheme to choose the best forwarding node, instead of using only the distance to destination like GPSR [3]. 3MRP works with local information that vehicles share with each other trough a beaconing process. In this way, forwarding decisions are taken hop-by-hop according with the current network conditions. Furthermore, this is a map-aware protocol since it includes the REVSim [36] tool used by vehicles to avoid sending packets to vehicles behind obstacles like buildings. This way the current forwarding node is building aware and it is able to discard those nodes behind vehicles to be selected as next forwarding nodes. In addition, 3MRP is able to forward small video warning messages in a multimedia format.

3MRP uses periodical interchange of hello messages (once per second) to get information from each neighbouring node. Each node shares its own hello messages with the other nodes into its transmission range. Afterwards, nodes include some new parameters in new fields of the beacon messages. Those additional fields are: node's speed, x and y axis values of the node's position, medium access control (MAC) layer losses in the link with that candidate node and nodes' density.

Using that data gathered from beacons interchanged with the neighbouring nodes, 3MRP computes five metrics in its forwarding node selection algorithm, which are depicted in the next section. In this way, when a node receives a hello message, it calculates not only the neighbour node's distance to destination, but also the other metrics in the list above. This multi-metric forwarding scheme allows 3MRP to take an optimal forwarding decision. Fundamentally, 3MRP considers the neighbour with the highest multi-metric score as the best next forwarding node.

3.2.2.1 Routing metrics used in 3MRP

The core of the 3MRP routing protocol is the design of five metrics:

- Distance from that candidate node to destination
- Trajectory of that candidate node
- Available bandwidth in the link with that candidate node
- Nodes' density of that candidate node

- MAC layer losses in the link with that candidate node

(i) The basic **distance** metric refers to the distance from each next forwarding candidate node to destination (Euclidean distance), where lower distances are preferred.

(ii) The **trajectory** is defined as the prediction of the future node's position, whose calculation allows the source node to know if the candidate node will be closer or going away from destination. For this case, nodes that move towards destination are preferable.

(iii) The **available bandwidth** considers the idle time of the wireless link formed between emitter and receiver, that is to say, between the current node holding the packet and each candidate next-hop node. The candidate node with more available bandwidth is better scored in that metric value. To compute the available bandwidth we have used the proposal presented in [37], as is explained in [4].

(iv) **Vehicles' density** is the number of nodes registered in the neighbour's list, so that nodes with a higher nodes' density value are better rated.

(v) **MAC layer losses** are computed locally at the node itself, and it is used as a kind of local feedback. A node with lower losses rate in the MAC layer, means that it is a better candidate to forward the message, so it will be better scored in that metric.

In accordance with the ad hoc principle of using only local information (i.e., infrastructureless operation), each sender node will compute a multi-metric score for each one of its neighbours. The node currently holding the packet will select the candidate with the highest multi-metric score as the next forwarding node to which the packet will be sent.

3.2.2.2 Forwarding decision in 3MRP

3MRP employs hop-by-hop forwarding decisions to forward packets towards their destination. As mentioned before, each sender node chooses from its list of neighbours the best forwarding node to send a packet. The neighbours that are not in line-of-sight with the sender node will be discarded from the list of neighbours using our REVsim [36] tool. This tool is used to detect presence of obstacles in the scenario by checking if two nodes in the same transmission range cannot establish communication due to an obstacle. Afterwards, each sender node computes a total multi-metric qualification assigned to each neighbour to determine the best next forwarding node.

Additionally, weights are assigned to each one of the five metrics to calculate our multi-metric score $\bar{u}_{Ngh}$ for each neighbour Ngh. For that, we have designed a dynamic self-configured weights (DSW) algorithm to update the weights dynamically throughout time, so that in case certain metric $u_{i,Ngh}$ is more overriding in the decision process its weight w_i increases.

$$\bar{u}_{Ngh} = \sum_{i=1}^{5} u_{i,Ngh} \cdot w_i \tag{3.1}$$

Finally, and by using Eq. (3.1), each sender node can compute a normalized multi-metric score for each candidate in its list of neighbours. The neighbour with the highest multi-metric value will be selected as the best forwarding node. In the next subsection, a brief summary of DSW can be found.

3.2.2.3 Dynamic self-configured weights (DSW) algorithm to update a multi-metric score

In this section we briefly summarize the DSW algorithm presented in [4]. Basically, DSW is used to compute multi-metric values using current network conditions, to score candidate forwarding nodes. With DSW, the current node holding a packet (the current sender node) dynamically updates a multi-metric score for each neighbour node, i.e. for each candidate node to forward the packet. The multi-metric score is updated depending on a set of current network metrics for that node. This way, the current sender node can take the best forwarding decision in that moment. When a sender node needs to forward a packet, it computes the multi-metric score value for all nodes included in its list of neighbours, using Eq. (3.1), where the weights $w_1, w_2, ..., w_5$ will be dynamically computed based on the DSW [4] algorithm to highlight the most decisive metrics. By doing this, the current sender node under analysis (Nan) will better select the best next forwarding node among the candidate nodes in its neighbours' list. Multi-metric scores for the five metrics mentioned in Section 3.2.2.1 will be assessed for each neighbour Ngh found in the neighbours' list: (i) Distance to destination $u_{dst,Ngh}$, (ii) trajectory $u_{trj,Ngh}$, (iii) nodes' density $u_{dns,Ngh}$, (iv) bandwidth $u_{abe,Ngh}$, and (v) MAC losses $u_{los,Ngh}$. More details about the equations of each one of the above mentioned metrics can be found in [4].

Let us R_m be the differential value for each m_{th} metric ($1 \leq m \leq 5$)) during the interval defined by the moments t_1 and t_2, being $t_2 > t_1$. The values R_m define a vector $\vec{R}$ expressed as follows:

$$\vec{R} = \begin{cases} R_1 = \dfrac{[u_1(t_2) - AV_1(t_2)] - [u_1(t_1) - AV_1(t_1)]}{2} \\ R_2 = \dfrac{[u_2(t_2) - AV_2(t_2)] - [u_2(t_1) - AV_2(t_1)]}{2} \\ \vdots \\ R_m = \dfrac{[u_m(t_2) - AV_m(t_2)] - [u_m(t_1) - AV_m(t_1)]}{2} \\ \vdots \\ R_5 = \dfrac{[u_5(t_2) - AV_5(t_2)] - [u_5(t_1) - AV_5(t_1)]}{2} \end{cases} \tag{3.2}$$

where $u_m(t_1)$, $u_m(t_2)$, $AV_m(t_1)$ and $AV_m(t_2)$ take values in the range [0, 1] for $m \in [1,5]$. Let us $u_m(t_1)$ and $u_m(t_2)$ be the current scores of each metric m in the moments t_1 and t_2, respectively. Let us $AV_m(t_1)$ and $AV_m(t_2)$ be the average score values of each metric m computed for all the neighbours of the current forwarding node. A negative value of R_m means that metric m is getting worst in the period $[t_1,\, t_2]$. In that case, R_m would be set to zero meaning the worst metric value. This way, R_m is defined in the range [0, 1].
Next, vector $\vec{R}$ is normalized by dividing each component R_m by the maximum value R_x found in vector $\vec{R}$, where x $\in [1,m]$. The normalized vector is named $\vec{S}$ and its components S_m take values in the range $0 \leq S_m \leq 1$:

$$\vec{S} = \begin{cases} S_1 = \dfrac{R_1}{R_x} \\ \vdots \\ S_x = \dfrac{R_x}{R_x} = 1 \\ \vdots \\ S_5 = \dfrac{R_5}{R_x} \end{cases} \tag{3.3}$$

Now vector $\vec{S}$ is normalized so that the sum of weights of all the metrics is equal to one. To do that, a variable ξ have been defined:

$$\xi = \frac{1}{\sum_{i=1}^{5} S_i} \tag{3.4}$$

Finally, the normalized vector of weights W is:

$$\vec{W} = \begin{cases} W_1 = S_1 \cdot \xi \\ \vdots \\ W_x = S_x \cdot \xi = \xi \\ \vdots \\ W_5 = S_5 \cdot \xi \end{cases} \tag{3.5}$$

To sum up, Eq. (3.5) will be used to compute the dynamic weights (W_1, W_2, W_3, W_4, W_5) instead of using equal weights (w_1, w_2, w_3, w_4, w_5) with $w_i = 1/5$, $1 \leq i \leq 5$. Notice that in the rare case that all the metrics for a specific node were getting worst, $W_m = 0$ for $0 \leq m \leq 1$, meaning that no preferences are given to any metric. In that case, all weights would be set to the equal weight condition, i.e. $w_i = 1/5$, $1 \leq i \leq 5$.

3.2.2.4 Some simulation results for 3MRP+DSW

Simulation results are presented to test DSW algorithm operation together with the 3MRP routing protocol. The performance evaluation is done in terms of average percentage of packet losses and average end-to-end packet delay. Two version of multi-metric routing protocol 3MRP were tested: (i) a first version (named 3MRP) uses equal weights to compute the multi-score value, and (ii) a second one (named 3MRP+DSW) using the dynamic self-configured weights scheme described in the previous section. Besides, 3MRP is compared with the basic GPSR [3] an with another routing protocol that improves GPSR named VIRTUS [38]. Recall that GPSR was described in section 3.2.1. VIRTUS is a routing protocol for VANETs also specially designed to transmit video streaming, as 3MRP. It uses distance to destination and nodes' density as metrics to take forwarding decisions. It also considers a time window to take forwarding decisions to forward packets.

The tests were implemented in an urban scenario from a Barcelona city area (1700 m x 580 m) with the presence of buildings. Two vehicles' densities are considered, $50 vehicles/km^2$ (sparse urban scenario) and $100 vehicles/km^2$ (dense urban scenario). The average vehicles' speed is 20 km/h, whereas their maximum speed is 50 km/h. Then, one fixed destination, an access point (AP), through which vehicles connect to the communication network to report traffic information, specifically a video-reporting message about a traffic accident.

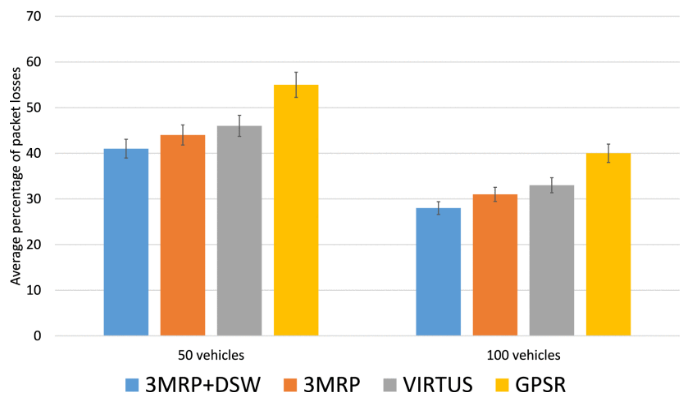


FIGURE 3.3: Average percentage of packet losses. (taken from ([4]).

Fig. 3.3 shows the average percentage of packet losses of 3MRP considering equals (3MRP) and dynamic (3MRP-DSW) weights. We can notice that a notable reduction of losses are achieved considering all five metrics and the scheme with self-configured weights (3MRP+DSW). The reason is that with 3MRP+DSW (blue columns in Fig. 3.3) a better forwarding decision can be made compared to 3MRP (orange columns) with fixed weights to derive the multi-metric score. Also, we can notice that both 3MRP approaches outperform VIRTUS and GPSR.

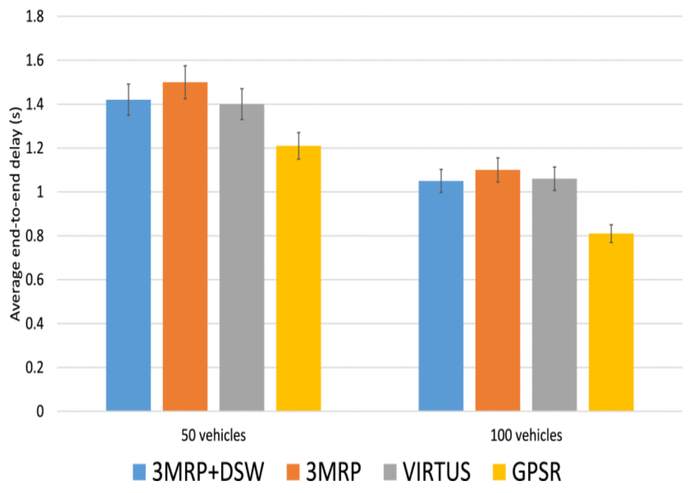


FIGURE 3.4: Average end-to-end packet delay (taken from ([4]).

Fig. 3.4 shows the average end-to-end packet delay. 3MRP represents a slight increment with respect to GPSR. This is more notable in sparse density of nodes. As previously described, GPSR takes forwarding decision based only on shortest distance to destination, it obtains the lowest packet delay in both scenarios. This is because with GPSR, a lower number of packets arrived at destination and much of the lost packets traveled through a considerable number of hops before being dropped. Nevertheless, 3MRP+DSW is able to reduce the packet losses with a lower increment in delay than 3MRP.

The notable improvement of 3MRP+DSW resides in the selection of the next forwarding node, which is done being aware of the special characteristics for VANETs (dynamic nodes' topology). 3MRP+DSW uses a dynamic weights scheme to derive the multi-metric scores for each candidate to be the next forwarding node. This scheme classifies nodes in a better way, giving each metric its importance depending on the current environment conditions.

3.3 Conclusions

In this chapter the main geographic routing protocols for VANETs have been reviewed and briefly summarized. As it has been pointed out, on of the first geographic routing protocols designed for VANETs was GPSR [3], which considers the minimum distance to destination as the metric to chose forwarding nodes to send information in a hop-by-hop bases. GPSR presents low average packets delays, but it is not efficient under the presence of obstacles when it activates the perimeter forwarding mode. Of course, some interesting improvements to the GPSR were published in the literature. They mainly focus on solving the failures of the two modes of the GPSR operation (greedy and perimeter forwarding) achieving improvements in terms of packet losses.

In this sense, 3RMP [4] argues that in the routing forwarding decision for VANETs more metrics should be considered, so that the routing protocol can adapt to the inherently dynamic VANET behaviour. 3MRP+DSW proposes five metrics (distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses) and a dynamic metric weighs scheme to classify candidate nodes. Therefore, each current sending node can classify its neighbours according to a multi-metric score and select the best candidate node in that moment to forward the current message to destination. 3MRP+DSW enhances GPSR with a significantly lower percentage of average packet losses.

In addition, the 3MRP+DSW routing protocol uses dynamic weights to compute a multi-metric score for each metric. Thus, it can be more adaptable to the inherent dynamic VANET characteristics. For this reason, we decided to use 3MRP+DSW as the basis to develop our proposals, which are presented in the next chapters.

Chapter 4

Contributions to improve 3MRP

4.1 Results of this chapter

The results from the proposals presented in this chapter have been published in:

- Ahmad Mohamad Mezher, **Leticia Lemus Cárdenas**, Julián Cárdenas-Barrera, Eduardo Castillo Guerra, Julian Meng, Mónica Aguilar Igartua, ”Improved Selection of the Best Forwarding Candidate in 3MRP for VANETs”, *IEEE Symposium on Computers and Communications (ISCC 2019)*, Barcelona, Spain. pp.1-6 2019. [39].

- **Leticia Lemus Cárdenas**, Ahmad Mohamad Mezher, Nely Patricia López Márquez, Pablo Barbecho Bautista, Julian Cárdenas-Barrera, Mónica Aguilar Igartua, ”3MRP+: An improved multi-metric geographical routing protocol for VANETs”, *15th ACM PE-WASUN*, Montreal, Canada.pp. 33–39, 2018. [40].

4.2 Introduction

In recent years, serious efforts from car manufactures, governments, as well as the research community, are being made with a clear objective of creating a standardized platform for vehicular communications. The special requirements and unique characteristics of VANETs (e.g., special mobility patterns, short life links, rapid topology changes) pose challenges for the research community. The goal is to have efficient routing protocols specially designed for VANETs.

Routing protocols implemented for VANETs can be classified in two categories: (i) topology-based routing and (ii) geographic-based routing. (i) Topology-based routing protocols establish and maintain end-to-end routes, which does not seem suitable because of frequent link failures caused by vehicle’s mobility that makes it difficult generate reliable and stable routes. (ii) Geographic-based, also called position-based, routing protocols have shown a better performance in VANETs than topology-based, since they take hop-by-hop forwarding decisions [31]. Thus, geographic-based protocols are better able to adapt to frequent topology changes. One of the first proposed geographic routing protocols was greedy perimeter stateless routing (GPSR) [3], a position-based protocol that considers the distance to destination as the single metric to take forwarding decisions [16]. In section 3.2.1 we summarized the GPSR operation. Other approaches that improve GPSR have been developed in the last years, as it was commented in section 3.2.1.1. In next section we highlight those approaches more related to ours that also use a multi-metric scheme.

As we pointed out in section 4.2, in this thesis we have contributed in the improvement of VANETs by means of the design of routing protocols specially designed for urban scenarios. The network topology is inherently dynamic in VANETs due to the high mobility of nodes. For this reason, it is necessary to design proper algorithms to assist routing protocols to forward packets following a hop-by-hop scheme from source to destination. We focus our work on urban scenarios (see Fig. 4.1) where the presence of buildings make communications more challenging. Also, our approaches consider multiple metrics to take the best forwarding decisions in the current moment.

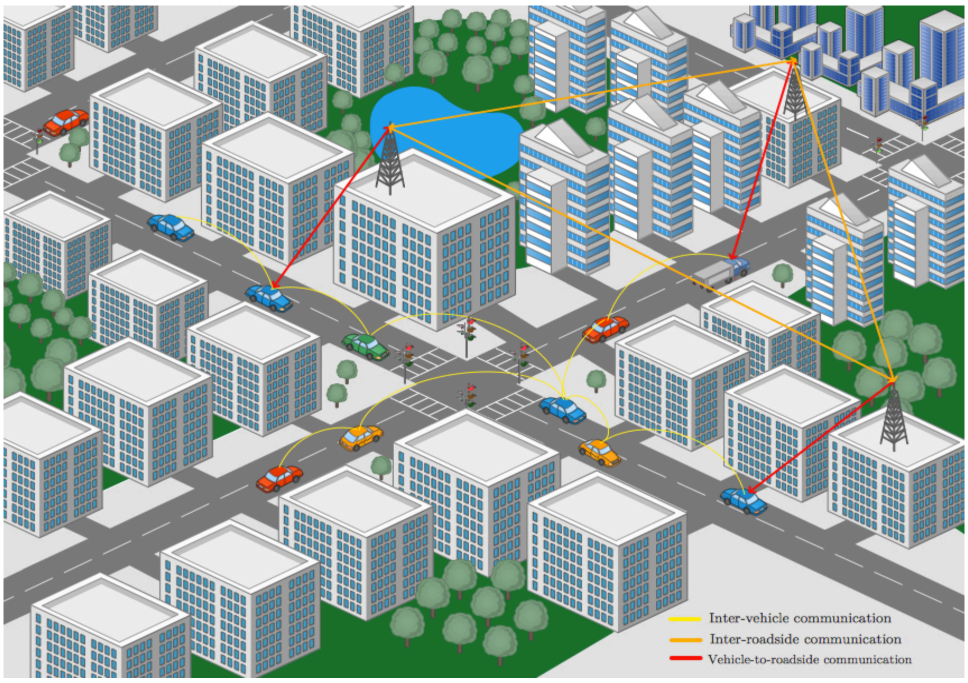

Inter-vehicle communication
Inter-roadside communication
Vehicle-to-roadside communication

FIGURE 4.1: Vehicular ad hoc network scheme in a urban scenario.

In this chapter, a methodology to select the best next forwarding node is proposed and analyzed. An extensive analysis is made to determine the best way to combine the different metrics designed in 3MRP by using the **unweighted power mean function** (see section 4.4). 3MRP considers several quality of service (QoS) metrics to select the best next forwarding vehicle for each packet in each hop towards its destination. These metrics are properly weighted to obtain a multi-metric score (computed arithmetically in [4]) for each vehicle in the transmission range so that the current forwarding node can take the best next hop forwarding decision.

In the same sense, in the second part of this chapter, **our approach accurately estimates the current position of the vehicles' neighbours in the moment of sending a packet** (see section 4.5). Nodes periodically interchange hello messages, which contain several metrics about each sender node, e.g. its location, trajectory, percentage of packet losses. Those metrics will be used by the routing algorithm to take packet forwarding decisions. So far, the forwarding algorithm uses the sender node's location taken from the last hello message received from that sender node. Conversely, a better estimation of the current position of the nodes will be computed taking into account the speed projection on the x-axis and the y-axis, named v_x and v_y, respectively. By default, the forwarding algorithm uses the nodes' distance (to destination) of a node received in the last beacon received from that node. Nonetheless, we claim that this distance can be quite obsolete in the moment of using it to take the current forwarding decision. Our approach updates the position of the node by accurately estimating its current position using the speed of the node. The reduction of error between the last received position of a node and its estimated current position depends on the speed of the aforementioned node. As the vehicle's speed increases, the new estimation of the current neighbours' positions will be much accurate than simply using the location received in the last beacon. Obviously, for static nodes there will be no differences between the conventional method and our method to asses the current nodes' distances.

The rest of the chapter is organized as follows. First, a list of related works is depicted in section 4.3 as a summary of the state of the art. Then, our first contribution of this chapter is presented in section 4.4, starting by the weighted power mean function description in section 4.4.2 and the simulation results in section 4.4.3. In the second contribution, an introduction of the proposed improved multi-metric routing protocol is presented in section 4.5, as well the motivation in section 4.5.1 and the analytically description of the estimation current node's position strategy in section 4.5.2. Then, simulation results and discussion are shown in subsection 4.5.3. Finally, conclusions are pointed out in section 4.6.

4.3 Related works

Position-based routing in VANETs refers to the process of selecting the best vehicle or vehicles in the neighbouring of the current forwarding node. This process is repeated in a hop-by-hop scheme till the packet reaches the intended destination. Most of the first routing protocols in the literature selected the best candidate node just using the distance to destination, thus following the shortest path. In the last years, alternative routing protocols that use several metrics to take better forwarding decisions have been published [41].

In a VANET, when a source node needs to send a packet to a destination (vehicle or RSU), it reviews its neighbours' table to select a forwarding node among the candidates. This is where the specific forwarding algorithm plays its node selection. The performance of the routing protocol depends on a good choice of the forwarding vehicle.

There are some proposals of routing protocols that use several metrics to select the best forwarding node, which improve the vanilla GPSR. One of them is EDAGF [42], in which each node maintains two tables, one for the neighbours' nodes and another for the RSUs. Both routing tables are organized and updated via beacon messages. First, the node checks its neighbours' table to see if the destination is there; if not it calculates position and distances to RSUs and neighbours to find a candidate (RSU or node) near to destination. It means that, when destination is registered in the neighbours' table the message is forwarded. However, we claim that the real destination position in the present could be different from the one updated in the last beacon received, because the position information in the table corresponds to a moment in the past.

M-GEDIR [43] selects next hop vehicles from dynamic forwarding regions and considers several parameters of the urban environment, including: (i) received signal strength, (ii) future position of vehicles, (iii) critical area of vehicles at the border of the transmission range, (iv) node's speed, (v) distance to destination and (vi) moving direction. All these metrics are calculated using information carried in the last beacon received.

Authors in [44] propose a real time forwarding method (R-FM) to decrease the beacon frequency and thus reduce the overhead in the network. It considers several metrics to select the optimal next node among the neighbouring nodes: (i) distance to destination, (ii) moving direction and (iii) link quality. However, the calculation of metrics is made considering also information from the past. If the nodes' speed is high, the current node's position could have notably changed since the last update. Then, errors about the nodes' real positions would be noticeable at the moment when a packet is required to be sent.

In [45], authors estimate the future position of the neighbours' nodes to describe their trajectory. Based on this information the forwarding decision is taken. In [4], one of the metrics used to choose the best forwarding node is the nodes' trajectory metric as well. However, for the distance metric they have also used the nodes' positions reported in their last beacon messages instead of the current position of the nodes.

In section 3.2.1.1 we also pointed out other representative approaches that improve the basic GPSR. However, all the routing protocols commented in this section include information about the nodes' positions which is only updated when a new beacon message is received. Conversely, it would be much accurate to use an estimation of the current position of each candidate node to take the forwarding decision. We tackle this goal in section 4.5.2.

Next, we highlight some representative works regarding the way in which the several considered metrics are averaged to obtain the multi-metric score for each neighbouring node. That multi-metric score is used to select the best next-hop vehicle to forward a packet towards its destination.

In [43] and in [44], authors implement just the addition of the weighted metrics. In [46], an AHP-based multi-metric geographical routing protocol (AMGRP) has been proposed. In AMGRP, authors employ an analytical hierarchical process mechanism to combine multiple decision criteria into a single weighing function. AMGRP considers several metrics: (i) mobility metric, (2) link lifetime, (3) nodes' density and (iv) node's status.

Finally, [4] uses an arithmetic global score function that combines five weighted metrics to compute the multi-metric score used to select the best next forwarding node. The considered metrics are: (i) available bandwidth on the link formed between the current sending node and each neighbour, (ii) nodes' density, (iii) node's trajectory, (iv) distance to destination and (v) percentage of MAC losses. Also, the same logic way is follow, its select the best forwarding node from the neighbours tables that are updated every time that a beacon is received.

In our proposal 3MRP+ [40] of a multi-metric routing protocol for VANETs, we estimate the current position of the neighbouring nodes in the moment of sending a packet. Thanks to that, we have achieved a better selection of the best next forwarding node since the estimated positions of the nodes are very close to their real positions. We have analysed results of extensive simulations, which show that the geometric mean attains the lowest average percentage of packet losses.

4.4 Improved selection of the best forwarding candidate in 3MRP for VANETs

The results of this section have been published in [39]. In this section, a strategy to improve the selection of the best forwarding candidate included in 3MRP [4] is presented. We will analyse different functions to combine three network metrics (bandwidth, distance to destination and nodes' density) to attain a multi-metric score for each candidate forwarding node in the neighbourhood of the current node holding the packet. In this context, nodes select the next forwarding node following different weighting schemes, so that we analyse which is the best way to weight the several considered metrics.

The multi-metric score is computed based on the weighted power mean function (W-PMF) [47]. The W-PMF or weighted generalized mean is just a way of expressing most of the common means (like the arithmetic mean) in one formula, see Eq. (4.1).

4.4.1 Motivation to analyse the W-PMF in our routing protocol

As we pointed out in section 4.1, in this thesis we have contributed in the improvement of VANETs by means of the proposal of routing protocols specially designed for urban scenarios. In the literature, e.g. [4, 48], different metrics have been proposed and included in a routing protocol for VANETs seeking to improve performance in terms of average percentage of packet losses and average end-to-end delay. Nevertheless, all of these works have used the classical arithmetic form to combine these metrics (see Eq. (3.1)) to classify and choose the best next forwarding node from a list of neighbours of the current packet-holding vehicle. Here, we have tested different ways to combine metrics using the W-PMF [47] in combination with the DSW algorithm using different values of p for the power mean function (see Eq. (4.1)). That is, for each p value we have applied the DSW algorithm [4] to obtain the corresponding weights for each one of the metrics. Let us see the detailed explanations in the following sections.

4.4.2 Weighted power mean function (W-PMF) to combine several metrics in routing protocols for VANETs

If p is a non-zero real number, and x_1,...,x_n are positive real numbers, we can define the weighted power mean for a sequence of positive weights with sum $\sum w_i = 1$, as:

$$A_p(x_1, ..., x_n) = \left(\sum_{i=1}^{n} w_i \cdot x_i^p \right)^{1/p} \tag{4.1}$$

Recall that the equally weighted mean is obtained when setting all $w_i = 1/n$. For the W-PMF, we can identify some special cases for different p values:

- The minimum value $(A_{-\infty})$.
- The harmonic mean (A_{-1}).
- The geometric mean $(A_{p \to 0})$.
- The quadratic mean (A_2).
- The cubic mean (A_3).
- The maximum value $(A_{+\infty})$.

Let $\mathbb{R}$ represent the set of real numbers, and let $\mathbb{O}$ denote an interval in $\mathbb{R}$. Then, $A_p(x_1, ..., x_n)$ is a power mean function of n variables defined on $\mathbb{O}$, with the following properties expressed for a general $\mathcal{M}$ function.

Property 1

$\mathcal{M}$ is a bounded function as follows:

$$min\{x_1, ..., x_n\} \leqslant \mathcal{M}(x_1, ..., x_n) \leqslant max\{x_1, ..., x_n\} \tag{4.2}$$

where $min\{x_1, ..., x_n\}$ and $max\{x_1, ..., x_n\}$ are equivalent to $A_{-\infty}\{x_1, ..., x_n\}$ and to $A_{+\infty}\{x_1, ..., x_n\}$, respectively, for all x_j in $\mathbb{O}$.

Property 2

The second property of $\mathcal{M}$ is that $A_p(a, ..., a) = a$ for all a in $\mathbb{O}$.

Property 3

The third property deals with inequalities that relate the harmonic mean (A_{-1}), the geometric mean $(A_{p \to 0})$ and the arithmetic mean (A_{+1}), as follows:

$$A_{-1}(x_1, ..., x_n) \leqslant A_{p\to 0}(x_1, ..., x_n) \leqslant A_{+1}(x_1, ..., x_n) \tag{4.3}$$

To illustrate the importance of determining which form of combining the considered metrics can lead to a better performance in VANETs, we show a numerical example in Fig. 4.2. Here we can see a source node S that needs to send packets to an access point (AP). We can see that source S has four candidates nodes (C_1, C_2, C_3, and C_4) within its transmission range. One of them will be selected to forward the packet following a hop-by-hop mode towards its destination.

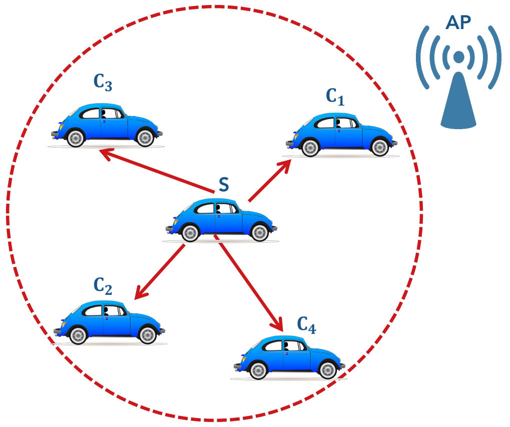


FIGURE 4.2: An example of a source node S that needs to send a packet to an access point (AP). S has four candidates nodes (C_1, C_2, C_3, and C_4) within its transmission range.

For the sake of simplicity, in this numerical example we use just three network metrics: (i) available bandwidth (BW), (ii) distance to destination (Dis2Des), and (iii) nodes' density. Each candidate node computes its metrics and normalizes them with Eq. (3.3). Results are shown in Table 4.1. Table 4.1 shows the multi-metric score using the power mean function for different values of p. We can deduce from Table 4.1 that depending on the p value, a source node would choose a different next forwarding node to send the packet. The best next forwarding node is the neighbour with the highest multi-metric value, highlighted in blue in Table 4.1. For instance, if we used the arithmetic mean $A_{p=1}$, the source node S would choose node C_1; however, by using the geometric mean $A_{p\to 0}$ or the harmonic mean $A_{p=-1}$, the source node S would choose node C_3 as the next forwarding node. If we used the quadratic mean $A_{p=2}$ or the cubic mean $A_{p=3}$, the source node S would choose node C_1. Also, using the maximum $A_{p\to+\infty}$ option, candidate node C_2 would be chosen. Finally, using the minimum $A_{p\to-\infty}$ option, candidate node C_4 would be selected. We assume in this numerical example that we have equal weights of 1/3 each.

In the next section, simulation results using the W-PMF algorithm included in the 3MRP routing protocol to average several metrics to take forwarding decisions, are shown.

TABLE 4.1: Numerical example. Metrics' values for four candidate nodes (C_1, C_2, C_3 and C_4) and their corresponding multi-metric score values.

Metric	C_1	C_2	C_3	C_4
BW	0.9	0.9	0.7	0.55
Dis2Des	0.2	0.5	0.7	0.55
Density	0.9	0.4	0.5	0.55
Multi-metric score	C_1	C_2	C_3	C_4
Arithmetic mean $A_{p=1}$	0.66	0.6	0.63	0.55
Geometric mean $A_{p\to 0}$	0.54	0.56	0.62	0.55
Harmonic mean $A_{p=-1}$	0.41	0.53	0.61	0.55
Quadratic mean $A_{p=2}$	0.74	0.63	0.64	0.55
Cubic mean $A_{p=3}$	0.78	0.67	0.64	0.55
Maximum $A_{p\to+\infty}$	0.9	0.9	0.7	0.55
Minimum $A_{p\to-\infty}$	0.2	0.4	0.5	0.55

4.4.3 Simulation results and discussion. W-PMF analysis

Simulation results are presented in this section. As stated earlier, all the analyses realized in this dissertation are focused on urban environments. Accordingly, the scenario description and main configured settings are described in section 4.4.3.1. After that, in section 4.4.3.2 the results of the simulations are presented in two figures that show the percentage of packet losses and the end-to-end average packet delay.

4.4.3.1 Simulation scenario to analyse the W-PMF

To evaluate the performance of VANETs for different p values (see Eq. (4.1)), we have used the Vehicles in Network Simulator (VEINS) [27] over the multilayer open source OMNET++ simulator [49]. The routes of the vehicles are generated through the Simulation of Urban Mobility (SUMO) [9] using real maps downloaded from OpenStreetMaps [5]. In this case, the zone of Berlin city shown in Fig. 4.7 is the simulation map.

The simulation settings of the scenario are shown in Table 4.2. All the figures show confidence intervals (CI) of 95% obtained from five simulations per point, each simulation with an independent mobility scenario and seed.

Table 4.2: Simulation settings of the VANET scenario

Map zone	**Berlin city**
Area	(2.5×2.5) km^2
Density of vehicles	100 vehicles/km^2
Number of nodes	625 vehicles
Transmission range	340 m
Average vehicle rate	20 km/h
Mobility generator	SUMO 25.0 [9]
MAC specification	IEEE 802.11p
Nominal bandwidth	6 Mbps
Source bit rate	CBR (10 packets/s)
Beacon interval	1 s
Simulation time	100 s
Routing protocol	3MRP [4]

The performance of the routing protocol 3MRP for different typical values of p is analyzed. The simulation area is 6.25 km^2. The network vehicles are randomly positioned and the density of vehicles is 100 vehicles/km^2. The average speed of the vehicles is 20 km/h while the maximum speed is 50 km/h. Recall that the multi-metric score used in the forwarding scheme of the 3MRP uses the dynamic weights algorithm (DSW), see section 3.2.2.3. We assume that a vehicle that acts as a source node transmits a warning text message (e.g., to report a traffic accident) periodically to a fixed destination located at a roadside unit (RSU).

4.4.3.2 Simulation results

In the following we will present the results of our performance evaluation and we will discuss the results. We show the average percentage of packet losses (see Fig. 4.3) as well as the average end-to-end packet delay (see Fig. 4.4), for different values of p in the weighted power mean function (see Eq. 4.1).

The average percentage of packet losses for different values of p in the weighted power mean function, is depicted in Fig. 4.3. We observe that the lowest average percentage of packet losses equals 28.5% when p tends to zero, which is the case of the geometric mean.

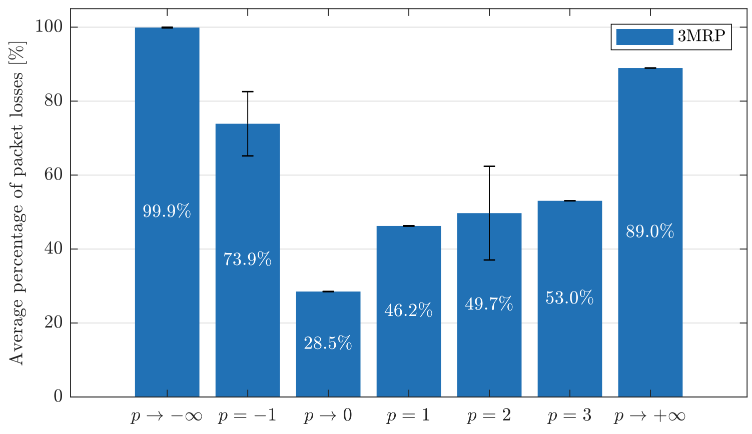


FIGURE 4.3: Average percentage of packet losses with 3MRP for different p values in the weighted power mean function used to weight the metrics.

The worst two results of packet losses is obtained when p tends to $\pm\infty$. The reason is the following. In the first case when p tends to $+\infty$, we choose only the highest score value of the five metrics, which is equivalent to take into account only one metric (i.e., the one which has the highest score value). In the second case when p tends to $-\infty$, we choose only the lowest score value of the five metrics, which means that again we are also taking into consideration only one metric, but in this case, the one which has the lowest score value, which should provide even worst results than the one obtained when p tends to $+\infty$, a fact that is shown in Fig. 4.3. Recall that the higher the score of a node, the better the node is considered to be selected as the next forwarding node.

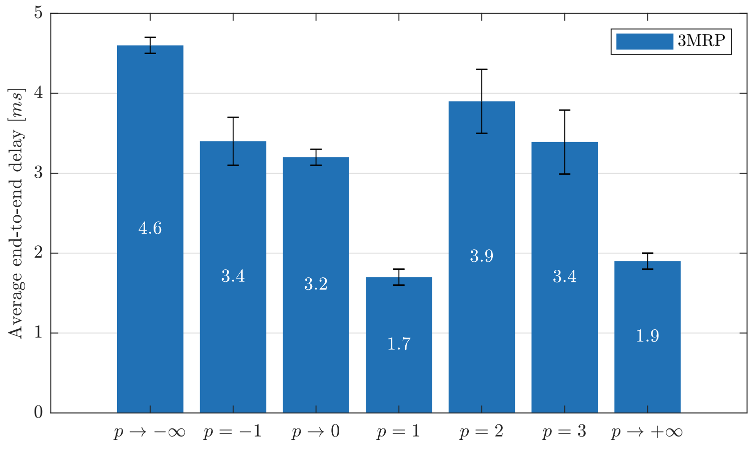


FIGURE 4.4: Average end-to-end packet delay with 3MRP for different p values in the weighted power mean function used to weight the metrics.

Fig. 4.4 shows the average end-to-end packet delay for different p values, specifically the special cases of the W-PMF. Recall that the delay is calculated using only those packets that successfully arrived at destination. We can notice that there is not a significant difference between the results obtained for the different p values, since the minimum delay is 1.7 ms when $p = 1$ (arithmetic mean) and the maximum is 4.6 ms when $p \to -\infty$

(minimum metric), which is an acceptable range of values for the average end-to-end packet delay in most of the VANET applications. Thus, by seeing Fig. 4.3 and Fig. 4.4, we can deduce that for $p \to 0$ (geometric mean), we obtain the best performance in terms of packet losses and average end-to-end packet delay compared to the other tested p values.

4.5 An improved multi-metric geographical routing protocol for VANETs (3MRP+)

The results of this section have been published in [40]. In this chapter, we propose and analyze a method to select the best forwarding node by accurately estimating the current position of the vehicles' neighbours in the moment of sending a packet. Nodes periodically interchange beacons (hello messages) which contain several metrics about that sender node, e.g. location, trajectory, packet losses. Those metrics will be used by the routing algorithm to take packet forwarding decisions. So far, the forwarding algorithm uses the sender node's location received in the last hello message received from that sender node. Conversely a better estimation of the current position of the nodes will be computed taking into account the speed projection on the x-axis and the y-axis, named v_x and v_y, respectively. By default, a forwarding algorithm would use the distance received in the last beacon from a node, although we claim that this distance can be quite obsolete in the moment of using it to take the current forwarding decision for a packet. Our approach updates the position of the node by estimating its current position using the speed of the node. The error reduction between the last received position of a node and its estimated current position depends on the speed of the aforementioned node. As the speed of the vehicle increases, the new estimation of the current neighbours' positions will be much more accurate than simply using the location received in the last beacon. Obviously, for static nodes, there will be no difference between the conventional method and our method.

4.5.1 Motivation to develop our 3MRP+ routing protocol

In a position-based routing protocol, after retrieving the location identification of the packet's destination, the source node searches a next forwarding node. That next-hop node should fulfil some requirements established by the routing metrics considered in the forwarding algorithm, such as shortest path to destination, a minimum density of neighbours, a minimum available bandwidth, moving trajectory towards destination or future position be closer to destination, among other metrics.

In the literature there are several proposals of VANET routing protocols with respect to the implementation of diverse methods and strategies [4] [44] [50] that have shown a good performance. The beaconing period is an important factor that affects the performance of the routing protocol in terms of overhead. Another determinant aspect, is the moment when the next forwarding node is selected. Most of the proposals use the information exchanged by the nodes through the beaconing process, allowing the algorithm to calculate or estimate the positions of the neighbouring nodes, the bandwidth link, the time duration link, the speed node, the moving direction and future position, among other metrics. These combined metrics will be used by the routing protocol to make a good node selection to forward packets.

It is important to highlight that in most of the existent routing protocols, the information used to take the packet forwarding decision corresponds to a time in the past. That information usually corresponds to the last hello message received from each neighbouring node. In the next section, we present our proposal to accurately estimate the current node's position to be used in the packet forwarding moment.

4.5.2 Accurate estimation of the current node's position

In this section, our methodology to estimate the current node's position is presented. We aim to improve the performance of 3MRP [4] by correcting the positions of the neighbouring nodes' of the current packet-holding node. Consequently, this node will be able to take an optimal selection of the next forwarding node. So far, the neighbours' positions were taken from their last hello messages received by the current forwarding node. Instead of using the distance reported in the last beacon message, we will estimate the current position to be used in the moment when a packet needs to be forwarded. In that moment, a next hop node must be selected and it is important to use realistic updated distances of the neighbouring nodes. We call our new approach as 3MRP+ [40]. Simulation results show the benefits of our proposal 3MRP+, since better forwarding candidates are selected.

For a fair comparison with 3MRP, in this section we use in our proposal 3MRP+ an arithmetic weighting function to compute the multi-metric score, since 3MRP uses an arithmetic weighting function.

Let us consider the moment $t_s = t_1 + \Delta t$ (see Fig. 4.5) when a source node S needs to send a message, where t_1 is the moment when the last beacon was received, t_s is the current sending moment and Δt is the incurred time between both moments. The S node will search the packet's destination in its neighbours' list. If the destination is not in the neighbourhood, S looks for a proper neighbouring node to forward the message. Considering a beacon period $T_b = t_2 - t_1$ (set to 1 second in our simulations), we have that Δt fulfils $0 \leq \Delta t \leq T_b$, as it is shown in Fig. 4.5.

The position information of each candidate node is taken from the last beacon received from that node at moment t_1. That position will not be updated until the reception of the next beacon at time t_2. Hence, the current node S holding the packet uses the previous known position information of each candidate node in time t_1 to select the next forwarding node to which it will send the packet at moment $t_s = t_1 + \Delta t$. Nevertheless, we claim that it is possible to accurately estimate the current position of candidate nodes at t_s. This way, the forwarding algorithm will be able to select a better candidate node for that packet in that moment.

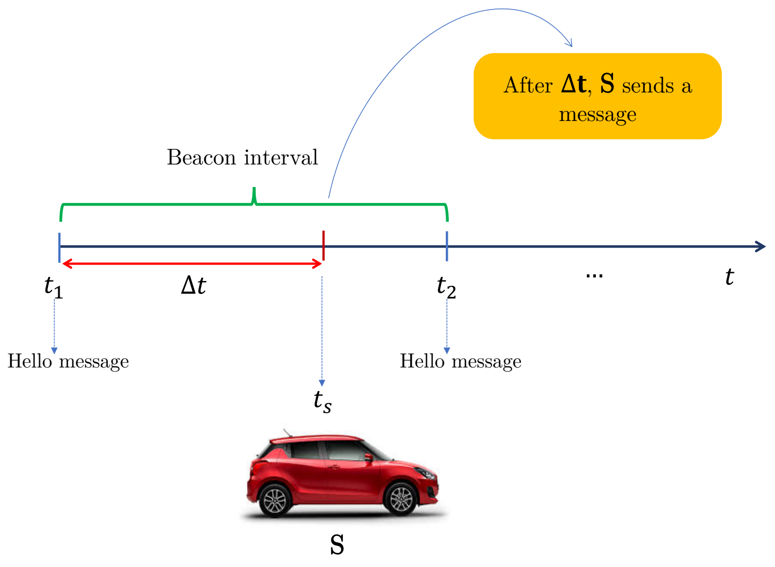


FIGURE 4.5: Time line representation of the beacon interval $(t_2 - t_1)$ and a sending message event at moment $t_s = t_1 + \Delta t$.

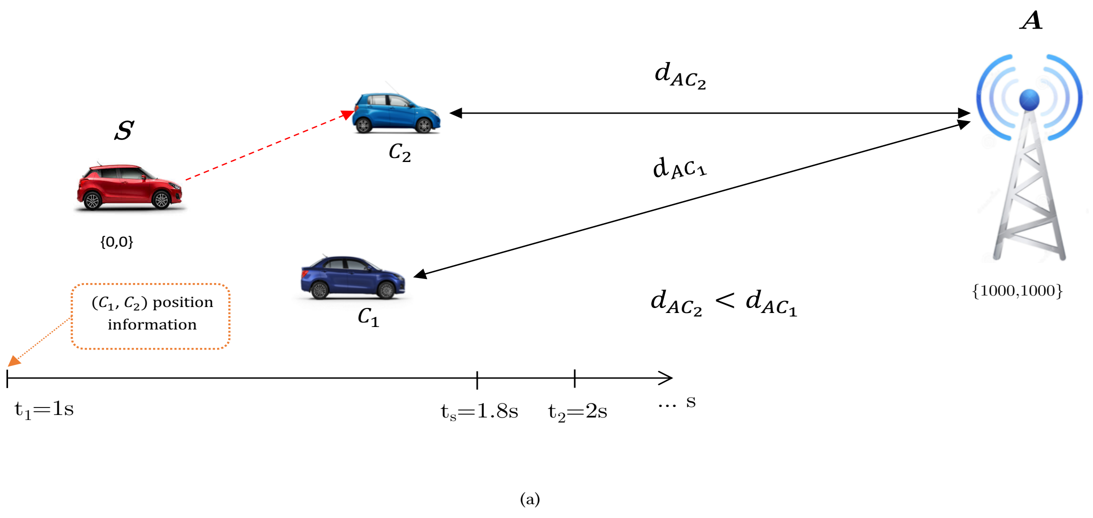
For the sake of simplicity we will consider Euclidean distance, although other distances (e.g. distance travelled over the roads) could also be used. It is well-known that in the Euclidean space $\mathbb{R}^2$, the distance d between two points (x_1, y_1) and (x_2, y_2) is given by:

$$d = \left(\sum_{i=1}^{2} |(x_i - y_i)|^2 \right)^{\frac{1}{2}} \tag{4.4}$$

Let us give a simple numerical example to explain the proposed algorithm to estimate the node's position at moment t_s, see Fig. 4.6. Suppose the sender node S is at position $(0, 0)$ and needs to send a packet at the moment $t_s = t_1 + \Delta t = t_1 + 0.8s$ to an access point A located at position $(1000, 1000)$. Let us consider that node S has two candidate nodes C_1 and C_2 in its neighbourhood to forward the packet. We assume that the two last beacon messages from both candidate nodes C_1 and C_2 were received at S right before moment $t_1 = 1s$ (not simultaneously so they do not collide). 3MRP [4] would compute distances $d_{(A-C_1)}$ and $d_{(A-C_2)}$ between the access point A and the two candidates C_1 and C_2, respectively. To obtain those distances, we use equation (4.4) with the nodes' last positions (x_1, y_1) and (x_2, y_2) reported in their beacon messages, as it is shown in Table 4.3. Using equation (4.4) we obtain $d_{(A-C_2)} = 692,9646$ and $d_{(A-C_1)} = 707,1068$. Since $d_{(A-C_2)} < d_{(A-C_1)}$, 3MRP chooses C_2 as the next forwarding node to send the packet at moment $t_s = 1.8s$ (see Fig. 4.6 a).

TABLE 4.3: Parameters of two candidate nodes reported in the last beacon messages.

Candidate node	**x(m)**	**y(m)**	v_x**(m/s)**	v_y**(m/s)**
C_1	x_1=500	y_1=500	20	20
C_2	x_2=510	y_2=510	2	20



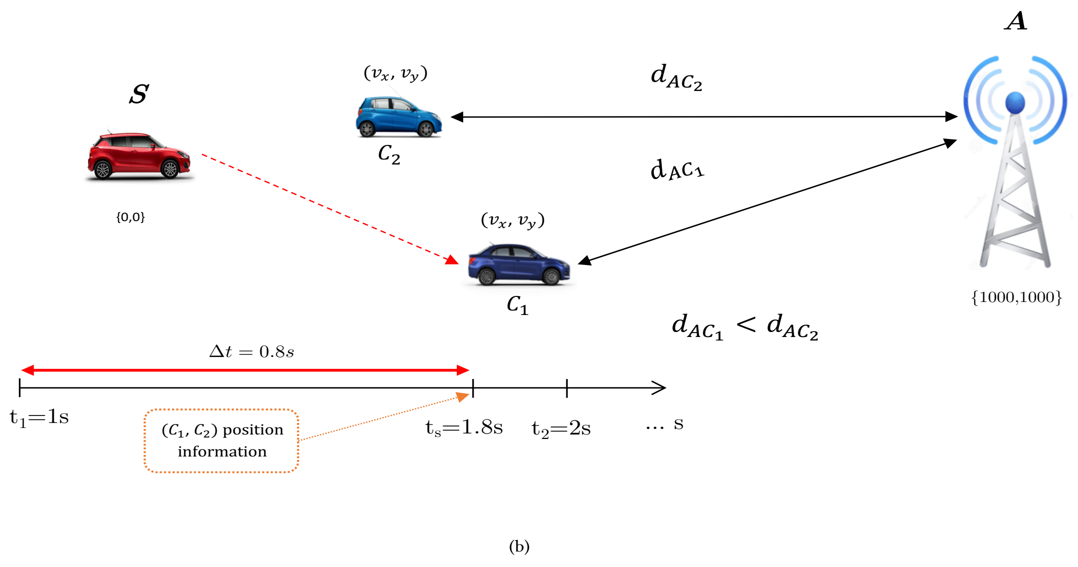


FIGURE 4.6: Selection of a candidate node according to (a) 3MRP or (b) 3MRP+.

However, if we take into account the (v_x, v_y) speed of each one of the candidates to estimate their current positions at the moment when the packet is intended to be sent, i.e. $t_s = 1.8s$, we will be able to take a better forwarding decision. Distances $d_{(A-C_1)}$ and $d_{(A-C_2)}$ will be thus computed using the estimated current positions of C_1 and C_2 instead of the reported ones via beacon messages (see Table 4.3), which are past positions not updated. Contrary to 3MRP [4], 3MRP+ [40] is able to update the nodes' locations at $t_s = 1.8s$. Thus, 3MRP+ estimates the current positions for nodes C_1 and C_2 using equation (4.5) and it detects that $d_{(A-C_1)} < d_{(A-C_2)}$. Therefore, C_1 would be selected by 3MRP+ as the best next forwarding node instead of C_2, which would have been selected by 3MRP.

To perform the 3MRP+ [40] operation, we need to add two more fields in the beacon messages to allocate the previous node's position (x_1, y_1) of the forwarding candidate (see Table 2) which sends the beacon (see Fig. 4.6 b).

TABLE 4.4: Format of the new fields in the hello message.

v_x	v_y	x_1	y_1

For high mobility scenarios, an estimation on the current nodes' position will improve the performance of any routing protocol in which its decision to choose the best forwarding node depends completely [3] or partially [51] on the distance to destination. When a node needs to forward a packet, it must estimate the current position of the forwarding candidates according to this equation:

$$(x_{est}, y_{est})(\Delta t) = (v_x, v_y) \cdot \Delta t + (x_1, x_2), 0 < \Delta t < T_b \tag{4.5}$$

where (x_{est}, y_{est}) corresponds to the estimation of the current position of a candidate node used by the node currently holding the packet. The node's speed is represented by (v_x, v_y), and Δt refers to the interval time elapsed since the moment when the last beacon from that candidate node was received.

To select the next forwarding node for the packet, our algorithm arranges the neighbouring vehicles of the current holding node in a list. This arrangement is done according to the distances to destination of each candidate node. Those distances are computed with the estimated current positions of each candidate node. After that, if the destination node is not in the neighbouring list of the current holding node, the best candidate node (i.e., the closest node to destination) in the top list position is selected to forward the packet.

Algorithm 1 describes the sorting process of the neighbours list. First of all, a variable i is initialized with value equal to one, and it will be incremented until it reaches a value equal to the size of the neighbours list. Two pointers are pointing to the two first locations of the neighbours list (lines 2 and 3). Then, for each list location the current node position is calculated (line 6). After that, the distance to destination with its new updated position is calculated. Then, that distance is compared to the other nodes' distances in order to arrange in the high list location the node with the smallest distance value (lines 7 to 12). Once the list is arranged with the new estimated current positions of all neighbouring nodes, the current sending node can find the best next forwarding node in the top of the list.

Algorithm 1: Calculation of the distance to destination of a candidate node using its estimated position Δt seconds after the last beacon message from that node was received.

```
Requirement: Neighbours list, destination position and current time.
Require     : $i = 1$, $curr = head$ #initial position
while $i <= \#list\ of\ neighbours$ do
    $tmp = curr$;
    $curr = curr \rightarrow next$;
    read current;
    if $(curr! = NULL)$ then
        $\Delta t = current\ simulation\ time$ - $floor\ (current\ simulation\ time)$;
        Calculate: $curr(x_{est}, y_{est})$; $tmp(x_{est}, y_{est})$;
        Distance1 $= curr(x_{est}, y_{est}) \rightarrow$ Destination;
        Distance2 $= tmp(x_{est}, y_{est}) \rightarrow$ Destination;
        if $(Distance1 < Distance2)$ then
            $head$=$tmp$;
            $tmp$=$curr$;
            $curr$=$head$;
        end
    end
    $i$++;
end
```

By doing that, we are able to estimate in a more accurate way the position of each candidate node when a packet needs to be sent at a specific moment t_s between the arrival moments of two consecutive hello messages.

4.5.3 Simulation results and discussion. 3MRP vs. 3MRP+

In this section, we present simulation results and analysis of results of our proposal 3MRP+ [40]. To evaluate the performance of this proposal, we use the *Vehicles in Network Simulator* (VEINS) [27] over the multilayer open source *OMNeT++* simulator [49], and the routes generator *Simulation of Urban Mobility (SUMO)* [52] with real maps extracted from *OpenStreetMaps* [5]. We have developed the basic structure of our routing tables based on the code [53]. The results obtained with the implementation of our proposed algorithms are presented in the next sections.

4.5.3.1 Simulation scenario to analyse the performance of 3MRP vs. 3MRP+

The urban scenario used in the tests corresponds to an area of Berlin, taken from Open-StreetMaps [5] (see Fig. 4.7). The size of the simulation area is 6.25 km^2 using a vehicle as source node that transmits a warning text message periodically to a fixed destination located at a roadside unit (RSU). Both source and destination nodes are positioned 1000 m from each other. The vehicles' density is 100 vehicles/km^2, which is considered a high-density value. Vehicles are randomly positioned, with different average speeds (14, 25 and 33 m/s).

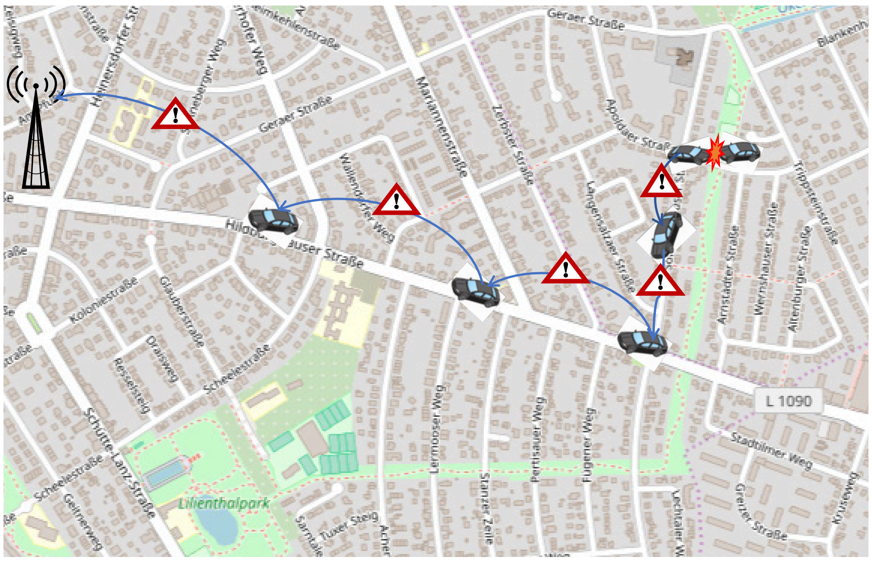


FIGURE 4.7: Map of a zone of Berlin, taken from OpenStreetMap [5], used in our tests. A warning message is sent from a crashed vehicle to a road side unit (RSU) through a VANET.

We have analyzed the performance of our new proposal 3MRP+ [40] and we have compared it to 3MRP [4]. We have done five simulations for each average speed under different scenarios with independent seeds each. Table 4.5 summarizes the main simulation parameters.

TABLE 4.5: Frequency of Special Characters

Map zone	**Berlin city**
Area	2.5 x 2.5 km^2
Density of vehicles	100 vehicles/km^2
Number of nodes	625 vehicles
Transmission range	340 m
Mobility generator	SUMO 21.0
MAC specification	IEEE 802.11p
Nominal bandwidth	6 Mbps
Source bit rate	CBR (10 pkts/s)
Beacon interval	1 s
Simulation time	100 s
Routing protocol	3MRP [4], 3MRP+ [40]
Average nodes' speed	14, 25 and 33 m/s

4.5.3.2 Simulation results

To carry out the performance evaluation, we have obtained the average percentage of packet losses and the average end-to-end packet delay. Fig. 4.8 depicts the average percentage of packet losses in 3MRP [4] compared to 3MRP+ [40] that includes the estimation of the current nodes' positions. We can see that 3MRP+ shows a lower percentage of average packet losses compared to 3MRP. 3MRP+ outperforms 3MRP in approximately 11%, 17% and 15% at 14, 25 and 33 m/s, respectively.

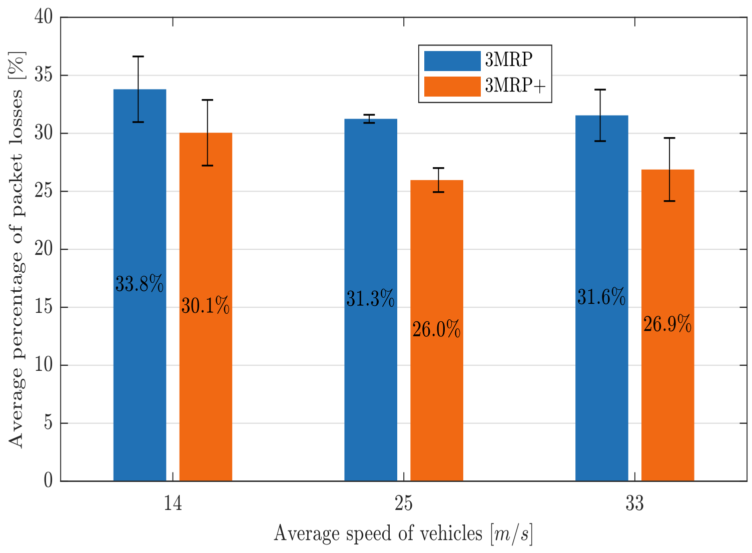


Figure 4.8: Average percentage of packet losses.

The reason is that as vehicles increase their speed, their positions also change rapidly. Therefore, the position error made in the transmission moment t_s by using the old node's position at t_1 is notably higher at high speeds (see Fig. 4.5). We can see that our new proposal 3MRP+ [40] improves efficiency since it decides the next forwarding node with the current position at the t_s moment when the current holding node forwards the packet. Another interesting benefit of our approach is the trade-off between accuracy in the node's position and overhead. In 3MRP we could achieve a better node's position closer to the current one if we increased the beaconing frequency. This way, the beaconing interval would decrease, and the difference between the real position and the reported one in the last beacon message would decrease too (see Fig. 4.5). That is, with 3MRP, the sender node could have a more accurate candidates' positions within its transmission range if beacons arrived at a higher frequency. However, there would be a too high cost of overhead. Contrarily, if the beacon frequency decreased, the sender node would have less accuracy regarding its candidates' positions but at the same time with a lower overhead incurred.

In 3MRP+ we solve this issue by accurately estimating the real neighbours' positions of a sender node at the precise moment t_s of sending the packet without modifying the usual beacon frequency. This solution has no increment in the overhead injected to the network. Let d_{err} be the distance error position of a specific node during time interval Δt between two consecutive beacons, also called hello messages (HM):

$$d_{err} = \sqrt{(x_{HM} - x_{P\_\Delta t})^2 + (y_{HM} - y_{P\_\Delta t})^2} \tag{4.6}$$

where (x_{HM}, y_{HM}) refers to the node's position according to its last hello message received; $(x_{P\_\Delta t}, y_{P\_\Delta t})$ is the estimated current node's position at t_s (see Fig. 4.5). The distance traveled by the node in Δt seconds depends on the node's speed v. In Table 4.6 we illustrate how an increase or decrease in the beacon period (T_b) will affect both d_{err} and overhead.

TABLE 4.6: Effect of the beacon period over distance error (d_{err}) and overhead

Beacon period	d_{err}	**overhead**
$T_b \uparrow at\ v > 0$	$\uparrow$	$\downarrow$
$T_b \downarrow at\ v > 0$	$\downarrow$	$\uparrow$
Using 3MRP+, T_b remains equal, $v > 0$	$\downarrow$	$=$

From Table 4.6 we can see that normally when the beacon period increases ($T_b \uparrow$), the overhead decreases ($\downarrow$), although the distance error due to the vehicle's mobility increases ($d_{err} \uparrow$). On the other hand, if the beacon time decreases ($T_b \downarrow$), as a consequence the distance error for a moving node decreases ($d_{err} \downarrow$); however, the overhead increases ($\uparrow$). Notice that in case that the speed of the node is equal to zero, $d_{err} = 0$. By applying our method, we are able to estimate with more exactitude the positions of the nodes without modifying the beacon period. Thus, we achieve a lower ($d_{err} \downarrow$) maintaining the same overhead. In this way, we can select more accurately the best forwarding node to route a packet.

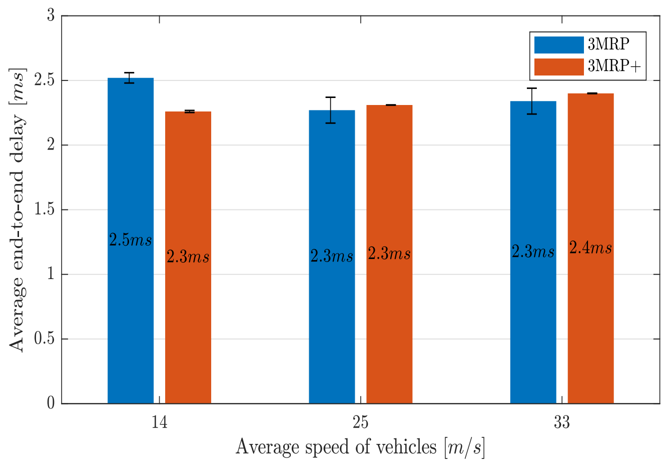


FIGURE 4.9: Average end-to-end packet delay.

Fig. 4.9 shows the average end-to-end packet delay. The delay is calculated based on those packets that successfully arrived at destination. We can clearly see that there is no differences between 3MRP [4] and 3MRP+ [40] regarding the average end-to-end delay values. Nevertheless, 3MRP+ shows better performance in terms of average packet losses (see Fig. 4.8). This is due to the accurate way of estimating the neighbour's positions at the exact moment of sending a packet avoiding to use the nodes' positions reported in the last hello messages received by the sender node. Overall, this first scenario explains how 3MRP+ works, and illustrates its advantages in terms of average packet losses and end to end delay. In the next subsection, the second scenario where different types of metric combinations are applied is explained in detail.

4.6 Conclusions

The results obtained in this chapter are twofold:

- We have analysed the weighted power mean function to see which is the best way to weight the several metrics considered to compute the multi-metric score for each candidate node. See section 4.4. The results of this contribution have been published in [40].
- We have developed a mechanism to correct the position of the candidate nodes in the moment of sending a packet. The node holding the packet can accurately estimate the current positions of its neighbouring nodes to take the best forwarding decision. We call our proposal 3MRP+. See section 4.5. The results of this contribution have been published in [39].

We have included these two contributions in the routing protocol 3MRP, previously developed by prof. Ahmad Mezher in his doctoral thesis [4].

The conclusions of each section are described below.

Improved selection of forwarding candidates
In section 4.4 we have described the importance of finding out the best way of combining different metrics in routing protocols. We have studied and analyzed different ways of weighting several metrics to obtain a multi-score value to score candidate nodes. This way, the current packet-holding node will be able to improve the selection of the best forwarding vehicle to be the next hop. Our analysis is done over the 3MRP routing protocol [4], which uses three metrics (available bandwidth, distance to destination and nodes' density). The goal of our contribution is to improve the performance of the routing protocol in terms of average percentage of packet losses and average end-to-end packet delay. Results show that the best way to combine those metrics is the geometric mean, with which the lowest average percentage of packet losses is obtained while keeping low the delay.

Our proposal 3MRP+
In section 4.5, a novel routing protocol called 3MRP+ [40] has been presented. 3MRP+ includes a method with which a sender vehicle can more accurately estimate the neighbour's positions at the exact moment of sending a packet. The benefits of our proposal are especially noticeable when the vehicles' speed increase. The new 3MRP+ proposal outperforms a previous version named 3MRP [4] by improving the number of packets received by the destination node, especially for high speed nodes. In addition, according with the simulation results 3MRP+ almost maintains same average end-to-end delay as 3MRP. The reason behind these good results is due to the realistic estimation of neighbour's nodes positions at the moment of sending a packet. Recall that we do not alter the beacon frequency, which means no extra overhead of messages injected to the VANET.

Chapter 5

Probability-based multi-metric routing protocol (ProMRP) for VANETs in urban scenarios

5.1 Results of this chapter

The results from the proposal presented in this chapter have been published in:

- **Leticia Lemus Cárdenas**, Ahmad Mohamad Mezher, Pablo Andrés Barbecho Bautista, Mónica Aguilar Igartua, "A Probability-Based Multi-metric Routing Protocol for Vehicular Ad Hoc Networks in Urban Scenarios", *IEEE ACCESS*, ISSN: 2169-3536, Vol.7, Iss.1, pp.178020-178032, 2019. [54]

5.2 Introduction

Routing information in VANETs in urban scenarios is more complex than in other environments, since it is necessary to consider multiple factors to evaluate candidate nodes that could potentially participate in the hop-by-hop forwarding path to destination. Therefore, it seems necessary to design routing protocols that are able to perform hop-by-hop forwarding (i.e., able to take local forwarding decisions instead of establishing end-to-end forwarding paths). Besides, routing protocols should take multi-metric decisions (i.e., they consider several metrics to take forwarding decisions) and be able to adapt to the inherent dynamic VANET conditions.

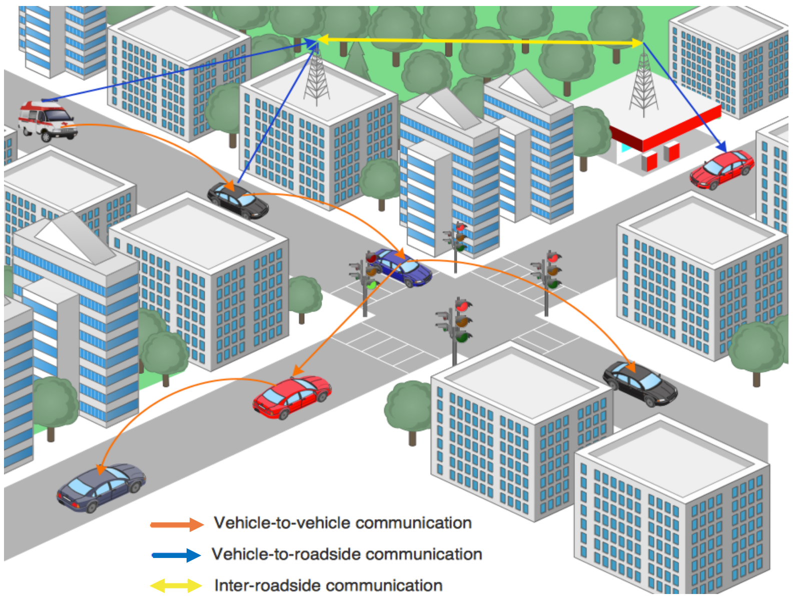

Vehicle-to-vehicle communication
Vehicle-to-roadside communication
Inter-roadside communication

FIGURE 5.1: Types of communications in VANETs in urban scenarios.

In this spirit, we focus our work on urban scenarios (see Fig. 5.1) where the presence of buildings, intersections and traffic lights make communications over VANETs specially challenging. We define our contribution in two parts:

1. Our forwarding algorithm includes four metrics, which are: (i) distance to destination, (ii) node's position, (iii) available bandwidth and (iv) nodes' density. Each metric is modeled with a probability density function based on a large number of representative simulations with a wide variety of configuration parameters. After designing the probabilistic distribution for each metric, the node currently holding the packet can estimate the probability for each candidate node to successfully forward the packet to its destination. Our novel proposal is named probabilistic multi-metric routing protocol (ProMRP) [54].
2. Furthermore, we have improved ProMRP with an accurate node's position estimation at the specific moment of forwarding the packet. This way, distances from each candidate node to destination, which are considered by the forwarding algorithm, are estimated with the current accurate positions instead of with those positions taken from the last beacon message received from each candidate node. We named this new version as enhanced ProMRP (EProMRP) [54]. Our proposal performs this mechanism at each forwarding node in a hop-by-hop scheme until the packet reaches its destination.

The rest of this chapter is outlined as follows. Section 5.3 presents relevant related works. Our proposal is explained in Section 5.4. Simulation results are analysed in Section **??**. Finally, Section 5.6 concludes this chapter and points out some future work.

5.3 Related work

Recent works have proved that the most efficient routing protocols for VANETs are those which consider several metrics to take forwarding decisions at intermediate nodes. Our research interest falls within proposals that consider multiple routing metrics to evaluate neighbour nodes to choose the best candidate node to forward a packet. Accordingly, we highlight in this section some recent interesting proposals concerning (i) multi-metric routing algorithms, (ii) probability-based estimation measures, and (iii) accurate estimation of the current node's position.

(i) Among the different proposals of multi-metric routing protocols available in the literature, we here highlight a few ones that are related to our approach. The main goal of multi-metric routing protocols is to consider several metrics to score neighbour nodes. Using that information, nodes decide which one is the best candidate to forward a packet towards its destination, making the routing process more efficient. In that sense, Path Aware GPSR (PA-GPSR) [55] considers the extension of its neighbour table (NT) with two additional tables. These two tables are called deny table (DT), which records those nodes that already forwarded the packet; and recent send table (RST), which records not only those nodes that already forwarded the packet but also the mode used for it. Using this extra information provided by the two NT extended tables, the routing protocol is able to manage efficiently the packet forwarding, avoiding both inappropriate routes to destination and packet loops.

The proposal called multi-metric geographic routing (M-GEDIR) [43] selects one of the neighbours as the next forwarding node based on nodes' positions, future positions, speed, distance to destination, signal strength and moving direction. In [56], authors propose a multi-metric routing protocol considering link lifetime, nodes' density, nodes' mobility and nodes' load (i.e., buffer queue length). They implement a hierarchical mechanism to combine multiple decision criteria such as the relative importance that each metric has with respect to the others. They use a weighted function to assign a weight to each one of the metrics. Nodes are scored according to a weighted multimetric score. Finally, the candidate node with the highest score is the most favorable to forward the packet. Their results improve the basic distance-based GPSR in terms of delay and packet delivery ratio. In [57] we can find another multi-metric proposal which considers a cluster-based forwarding approach using three metrics: throughput, vehicle's speed and available bandwidth. Those metrics are used to decide which neighbour node is the best candidate to forward each packet. On the other hand, the work [4] presents a proposal called multimedia multi-metric map-aware routing protocol (3MRP), which is one of the latest efficient proposals. This approach uses weighted metrics to select the next forwarding node. The metrics considered to score each candidate node are: available bandwidth in the link formed with each neighbour, nodes' density, trajectory, distance to destination and percentage of MAC packet losses.

(ii) Finally, we have identified three studies which show some relation to our probability-based approach, but with essential differences. In [58] authors propose two probability-based predictors to select the best forwarding node among the candidates in the list of neighbours. The first parameter refers to the probability that the signal to interference plus noise ratio (SINR) in a receiver node is larger than the receiving threshold during a period of time. The second parameter defines the probability of packet queue length (PQL) to be smaller than a maximum allowed value after an interval of time. Also, they propose a weighting function to calculate the utility value of each node. This utility consists in using the SINR and PQL variances as weights, and then averaging the total value for each node. Finally, the best candidate node is the one with the largest utility value. The [59] proposal considers two types of probabilities: (a) The forwarding probability measures the probability to access the channel; and (b) the successful forwarding probability, which refers to the probability that a node actually transmits. In the forwarding probability, the source nodes assigns values to its neighbours nodes so that the farthest node (from the source) has the highest forwarding probability. The probability of successful forwarding is the probability of exactly one node (within the transmission range of the source) transmitting at the beginning of an empty slot.

(iv) Taking the estimation of the node's position into account, VANET routing protocols can take more accurate forwarding decisions when they use metrics based on the node's position. For instance, in [60] authors estimate the future position of a node (after a time interval t) using the current node's location (x, y), the node's velocity (v_x, v_y), and the moving direction of the node θ. Every time a node receives a hello message, it calculates the new position of its neighbouring nodes, keeping a list of those nodes that approximate towards destination. In this way, they choose the next-hop forwarding node with the best future position (the shortest distance towards destination). A similar proposal that uses the same parameters to estimate the node's position is explained in [61]. Also, instead of arranging nodes based on their estimated future position, a weight is calculated for every new node's position. Such weight is based on the sum of three factors: P (relative distance to destination), q_1 and q_2 (angles formed with each neighbour and with destination). In the forwarding decision, the neighbour with the highest weight will be the next-hop node. Authors in [62], propose an algorithm to estimate the vehicle's movement in the near future and then selects the best neighbour. First, each node calculates its velocity (V_0) and the heading direction using previous position (x_0, y_0) and current speed at initial moment t_0. After that, the future position of a node is estimated adding the product of velocity and time interval Δt (time between two hello messages) to the current position. Unlike the two previous proposals, this future position is shared into the hello messages. In this way, all network nodes receive and save in their corresponding list the future position of all its neighbours. The next-hop forwarding node will be the one with the shortest estimated distance to destination in the near future. In [63] a system model to organise nodes in transmission zones (clusters) interconnected by headers nodes (HN) is proposed. Authors compare the positions of each intermediate node N_i (Ni_x, Ni_y) and destination node N_d (Nd_x, Nd_y), with their new positions $Pos_t(Ni)$ and $Pos_t(Nd)$ after a time (t). With this strategy, they configure end paths from a source node (S) to final destination node (D) through different zones and intermediate nodes. Again, the parameters used to calculate the node's position are: (x, y) coordinates, movement direction (θ) and velocity (v_x, v_y) in a time interval (t). Network nodes share and update this information via hello messages.

To summarize, in the literature there are several geographic routing protocols proposed for VANETs that consider several metrics to evaluate the best node to be selected as next-hop forwarding node. Examples of those metrics are distance to destination, available bandwidth, trajectory, vehicle's speed and nodes' density. Nonetheless, none of those proposals evaluates the probability that candidate nodes successfully deliver packets at destination as a function of the metrics' values.

In this chapter, we propose a novel probabilistic routing protocol named probabilistic multi-metric routing protocol (ProMRP) based on a probability-based forwarding algorithm to choose the best next forwarding node among the neighbouring candidates of the node currently carrying the packet. We focus our analysis on a realistic urban scenario using real maps. To the best of our knowledge, there is no proposal yet about a multi-metric routing protocol for VANETs that bases its forwarding decision on the successful delivery probability of the packet at destination. Besides, our proposal includes an estimation of future position of the nodes to further improve the forwarding decision. Finally, we have compared our work with the well-known GPSR and also with one of the most recent proposed routing protocol for VANETs named 3MRP [4]. In the next subsections the probabilistic routing algorithm is described in detail, also the its results and conclusions.

5.4 Design of our ProMRP proposal for VANETs

In this section we describe our proposal of a probabilistic forwarding routing algorithm for VANETs. This algorithm evaluates all candidate nodes and chooses the best candidate to forward the current packet. The selection is done based on the estimation of the probability of successful packet delivery at destination.

5.4.1 Motivation

After analyzing several proposals described in section 5.3, we observed that most of those that show a good performance in terms of packet losses and delay use several metrics to select intermediate nodes to forward packets, instead of just using the basic metric of the distance to destination. Certainly, it is crucial to consider several metrics in the packet forwarding to be able to address properly the special characteristics of VANETs.

Nevertheless, as far as we know none of the proposals consider the probability of packet successfully delivered as a metric in the forwarding algorithm. In this chapter we present our design of a probabilistic multi-metric forwarding algorithm with which nodes use several metrics to take forwarding decisions. This proposal outperforms other routing protocols in terms of average percentage of packet losses and average end-to-end packet delay. We have compared our proposals to the well-known GPSR [3] routing protocol as a reference, and to a recent proposal called 3MRP [4].

We claim that the sequence of events that represent the packet receptions in a hop-by-hop scheme in VANETs, can be represented as a probability distribution. This probability distribution describes the path that each packet will follow from source to destination.

This way, the best next forwarding node to forward a packet will be chosen hop-by-hop towards destination. First, we have carried out an off-line analysis over a representative data set taken from a large number of simulations under different representative scenarios. Then, the probability distribution of packet successfully delivered is obtained. Afterwards, we have implemented our probability-based routing proposal and, as section 5.5 depicts, the performance evaluation shows good results that outperform other proposals described in the literature [3] [4].

5.4.2 Modeling the probabilistic distribution of the considered metrics

The strategy employed by our algorithm is based on the probability distribution of the considered metrics. This requires a previous data organization and analysis to identify the categorical variables; that is, which variables have an impact on the subject of study.

Our aim is to find out the probability that a candidate forwarding node has to successfully deliver the current packet to its destination given some environmental parameters. The parameters that we have considered are: (i) distance to destination (traditional metric used to establish shorter forwarding paths); (ii) nodes' density (used to assess the topological connectivity); (iii) available bandwidth (computed from the percentage of time the channel is idle).

In order to provide a general description of our approach we will use an example, depicted in Fig. 5.2. Let us consider an output variable named Y which can take two values (0, 1) representing the occurrence of an event. In other words, the event is referred to whether a node will successfully deliver a message or not. Also, let us consider an input variable M with different environmental metric values that will have an influence in Y [64].

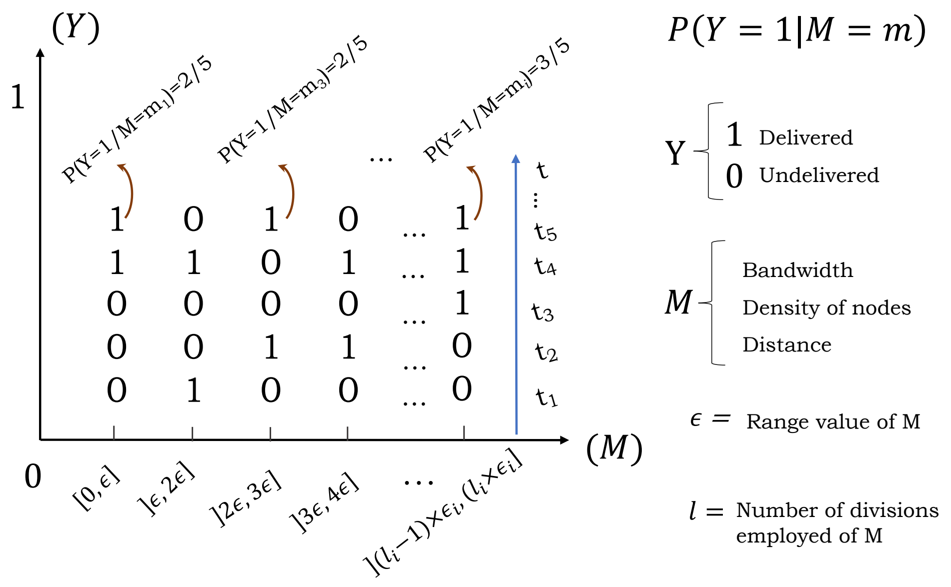


FIGURE 5.2: Estimation of the probability to successfully deliver a packet at destination given that the considered metric M equals m_i.

Fig. 5.2 shows a theoretical representation of an output sequence of random events throughout time. Let us suppose that we have n values of a metric M (e.g., distance to destination, available bandwidth, density of nodes) over time t. Then, we estimate the probability $P(Y = 1|M = m_i)$ of a candidate node to successfully deliver the packet to its destination ($Y = 1$) given a specific value m_i of the considered metric M at a given time t_j, where $1 \leq i \leq n$ and $1 \leq j \leq t$. Notice that it is quite hard that two or more values for the same metric at different time slot match. For the sake of simplicity, we divided the whole set of possible values for each specific metric into ranges in the form of $[0, \epsilon_i]$, $]\epsilon_i, 2\epsilon_i]$ until $](l_i - 1) \times \epsilon_i, l_i \times \epsilon_i]$ where ϵ_i is previously defined and l_i is the number of divisions employed between the minimum and the maximum value for each metric which is equal to $max(M)/\epsilon_i$. For instance, is the nominal bandwidth is 6Mbps, this means that the possible values of bandwidth for any node will be between 0 and 6 Mbps. So, having $\epsilon = 0.2$ means that the number of divisions will be 30 and the equal ranges will be as follows: [0, 0.2],]0.2, 0.4] until]5.8, 6]. Let us suppose that within the range [0, 0.2], our dataset has three values equal to 0.1 Mbps, 0.12 Mbps, 0.05 Mbps, with corresponding output Y values equal to 0, 1, 0, respectively. Thus, we can conclude that if we have any input value of bandwidth for a candidate node in the range of [0, 0.2], the probability that this candidate node successfully delivers the packet to its destination will be 1/3.

In Fig. 5.2, we can see that when the analyzed metric M takes the value m_1 between zero and $\epsilon = 0.2$ two packets (out of five) where successfully delivered. Therefore, the probability to successfully deliver a packet at destination given a specific value between zero and 0.2 $P(Y = 1|M = m_1) = 2/5$.

Three metrics have been taken into account to estimate (for each candidate forwarding node) the probability of successfully deliver a packet at destination. The node with the highest probability will be chosen as next forwarding node. This process is repeated hop-by-hop till the packet reaches its destination. In this way, the selection of the successive forwarding nodes is made adapting to the current network conditions. In the next subsection, the design of parameters and its implementation are explained in detail.

5.4.3 Design of parameters and implementation of ProMRP

To analyse the behavior of the different metrics considered in this chapter, a previous offline study over a real data set was done. We carried out a large number of representative simulations to prepare a consistent data set. The goal was to derive the probability density function (PDF) for each one of these metrics: (i) distance to destination, (ii) nodes' density and (iii) available bandwidth. To obtain our data set, we have considered an urban scenario where vehicles move with speeds in the range from 30 to 50 km/h. We have also considered different vehicles' densities and different types of streets.

Once the statistical model to represent each metric was obtained, we included those models in the forwarding algorithm of our proposed routing protocol. This way, vehicles can take their forwarding decisions according to estimations of the probability to deliver the packet to destination. The modeling of the PDF for each metric is described below. Notations are introduced in Table 5.1.

TABLE 5.1: List of variables used in the system.

Parameter	Definition
N_{gh}	Neighbour node.
D	Destination.
$d(N_{gh}, D)$	Distance between a candidate N_{gh} and D.
P_{dst}	Probability of packet successfully delivered at D for a given distance d.
$BW_{N_{gh}}$	Available bandwidth BW in the link formed with N_{gh}.
P_{BW}	Probability estimation of packet successfully delivered at the next-hop given BW.
$NV_{N_{gh}}$	Vehicles' density of the candidate N_{gh}.
P_{dns}	Probability of packet successfully delivered at D given $NV_{N_{gh}}$.
$\overline{P}_{N_{gh}}$	Score value of each candidate node N_{gh}.
T_b	Beacon period

Next, in the following sections we describe how each statistical model was obtained.

5.4.3.1 Exponential distribution to model the distance metric

The probability to successfully deliver a packet at destination for each possible next-hop forwarding node have been analyzed. This probability is obtained as a function of the distance from that node to destination. We claim that knowing in advance this probability can help to take better forwarding decisions.

A data set for the probability (Y) of packet successfully delivered at destination for a given distance ($d(N_{gh}, D)$) (d in short) from the candidate node to destination have been obtained. This data set was generated from many simulations with different vehicles' speeds (30, 60, 80 and 100 km/h) that cover the usual range of speeds we can find in urban scenarios. To have a wide range of distances in our urban scenarios, we set either highway-like roads, normal roads, and narrow streets.

We have organized the data set results in a graph, see Figure 5.3. In this figure, we show the PDF of the success in the packet delivery at destination, as a function of the distance between the candidate node and destination. The values taken from the data set (red line) are organized from the minimum distance (between a candidate node and destination) to the maximum one present in the data set. We can see that the PDF of packet successfully delivered (red line) fits well with an exponential distribution function expressed in Eq. (5.1) (see blue line). We have considered a sequence of random distances in the range from 200 to 850 meters, which are the minimum and maximum distance values obtained in the simulations. Notice that the probability to deliver a packet at destination tends to zero as the distance between a candidate node and destination grows.

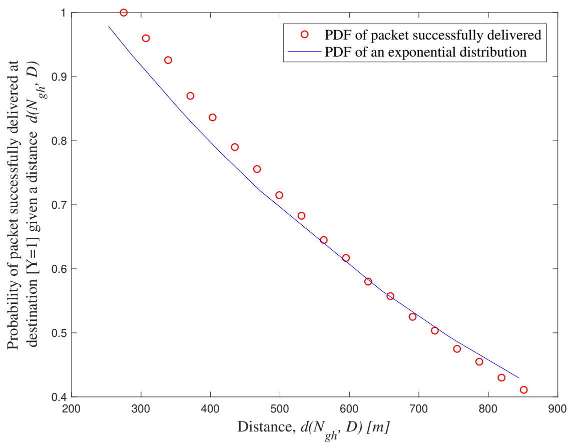


FIGURE 5.3: Probability density function (PDF) of the success in the packet delivery at destination, as a function of the distance $d(N_{gh}, D)$ between candidate node N_{gh} and destination D (red values taken from the data set). It fits well with the PDF of an exponential distribution (blue line).

$P(Y=1 \mid d)$ (P_{dst} in short) is the probability estimation of packet successfully delivered at destination, for a distance d between candidate node N_{gh} and destination D. Let λ be the parameter of the exponential distribution function.

$$P(Y=1 \mid d) = \lambda \cdot e^{-\lambda \cdot \frac{1}{d}}, \quad 0 \leq d < \infty \tag{5.1}$$

Using the *fitdist* function [65] of Matlab to adjust the values in our data set with Eq. (5.1), we got that the exponential distribution parameter λ equals $1.39 \cdot 10^{-3}$.
Once the neighbours' distances to destination d are obtained, the probability of succeed in the packet delivery at destination as a function of that distance d from each candidate neighbour N_{gh} to destination D, is calculated using Eq. (5.1).

5.4.3.2 Gaussian distribution to model the nodes' density metric

We generated another data set to analyse the effect of the neighbours' density in the PDF of the success in the packet delivery at destination. This data set was generated from a large number of simulations with different nodes' densities (50, 100, 150, 200, 250 and 300 vehicles/km^2) in an urban scenario, which cover most of the vehicles' densities in cities. Following the same methodology as in the previous section, we represent the PDF of the packet successfully delivered from the values taken from the data set, as a function of the nodes' density, see the red points in Fig. 5.4. We can see that those points adjust very well with a normal distribution, see the blue line in Fig. 5.4.

Accordingly, we obtained the parameters of a Gaussian curve $N(\mu, \sigma) = (175, 93)$ to fit with the data set values. The Gaussian curve (blue line) and the data set simulation values (red points), are shown in Fig. 5.4. Notice that the probability to deliver a packet at destination grows with the vehicles' density of the candidate nodes, since a higher network connectivity is preferred to success in forwarding the packet towards destination. This is true until a threshold (around 120 vehicles/km^2 in Fig. 5.4) upon which the number of collisions grows so much that hinders the forwarding of the packet.

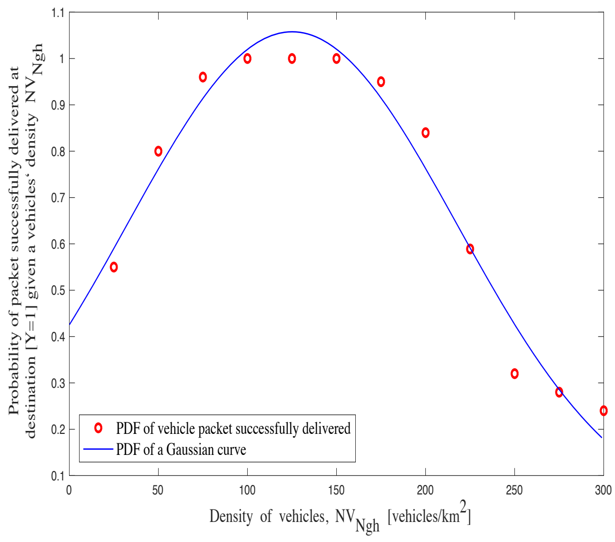


FIGURE 5.4: Probability density function (PDF) of the success in the packet delivery at destination (red values, taken from the data set), as a function of the vehicles' density of the candidates to forward the packet. It suits very well with the PDF of a normal distribution (blue line).

$P(Y = 1 \mid NV_{N_{gh}})$ (P_{dns} in short) is the probability that the packet arrives at destination by choosing a next-hop candidate N_{gh} given a vehicles' density that equals $NV_{N_{gh}}$.

$$P(Y = 1 \mid NV_{N_{gh}}) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{-\dfrac{(NV_{N_{gh}} - \mu)^2}{2 \cdot \sigma^2}} \tag{5.2}$$

$$0 \leq NV_{N_{gh}} < \infty$$

Finally, the PDF of the packet successfully delivered at destination, as a function of the vehicles' density can be described using Eq. (5.2).

5.4.3.3 Multinomial logistic regression to model the available bandwidth metric

In this section we assess the PDF of the success in the one-hop packet delivery at the next-hop forwarding node, as a function of the available bandwidth in the link formed by the node currently carrying the packet and each neighbour node candidate to be next forwarding node. Ad-hoc networks typically rely on hop-by-hop forwarding, where nodes use only local information to take forwarding decisions until the message reaches its destination. This strategy is simple and scalable. Besides, results are good enough while keeping a good trade-off among quality, overhead and simplicity [66].

Let us consider all those available bandwidth (BW) values that produce an output Y = 1 (delivered) or 0 (not delivered), according to the general scheme shown in Fig. 5.2. Results are represented in the red circles of Fig. 5.5. We can see that the results follow a distribution that fits with a logistic regression (LR), which can be described using Eq. (5.3). Once we observed this behavior, we applied a multinomial regression to find the coefficient estimates β_i. We found the categorical variable equal $\beta_0 = 0$ and $\beta_1 = -8.6707$. Adapting the LR model, we obtain this equation:

$$P(Y = 1 \mid BW_{N_{gh}}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot BW_{N_{gh}})}} \tag{5.3}$$

$$0 \leq BW_{N_{gh}} < \infty$$

$P(Y = 1 \mid BW_{N_{gh}})$ (P_{BW} in short) is the probability estimation of packet successfully delivered at the next-hop node given an available bandwidth BW in the link formed with neighbour node N_{gh}.

We plot the LR model and the data set results into the range of bandwidth values taken from simulations (0 to $6 \cdot 10^6$ bits/sec). In Fig. 5.5 we observe that the values taken from the data set simulation (red circles) fit well with the LR distribution model (blue line). Notice that the probability to deliver a packet at the next hop remains null until a threshold upon which the packet is successfully forwarded to the next hop.

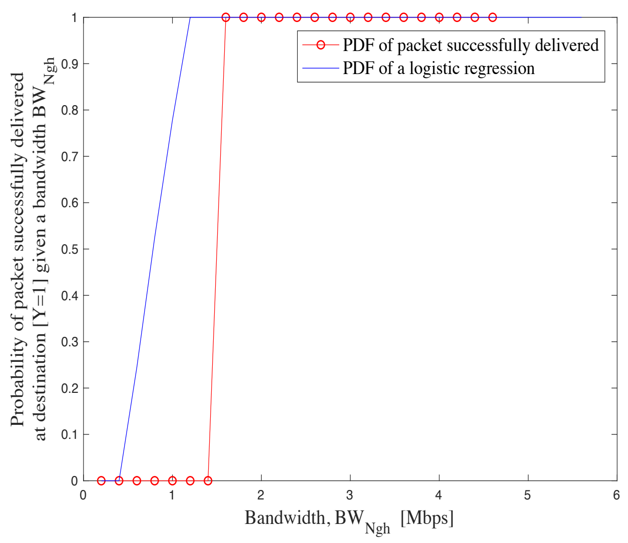


FIGURE 5.5: Probability density function (PDF) of the success in the packet delivery at the next-hop node, as a function of the available bandwidth in the link between the node currently carrying the packet and each candidate node to be next hop. Red circle values obtained from the data set. Blue line values obtained from the multinomial logistic regression (see Eq. (5.3)).

To estimate the available bandwidth in a link formed by two nodes, each node can measure its percentage of idle time in that link by sensing the common wireless medium. Each node includes this value in the next beacon message. We apply the available bandwidth estimator (ABE) according to [67]. Each node can estimate the bandwidth in the link formed with each one of its neighbours' upon the reception of their beacon messages (BMs). Finally, using Eq. (5.3) we can estimate the probability of packet delivery at the next-hop node as a function of the available bandwidth in the link formed by the node currently carrying the packet and each candidate node N_{gh}.

5.4.4 Probability-based forwarding decision

As we have said before, each statistical model obtained is used by the algorithm to take forwarding decisions. This forwarding decision works as follows. Each vehicle currently carrying a packet, will select a next-hop neighbour node to forward that packet towards its destination. To do so, nodes will use the three probability values described in the previous section.

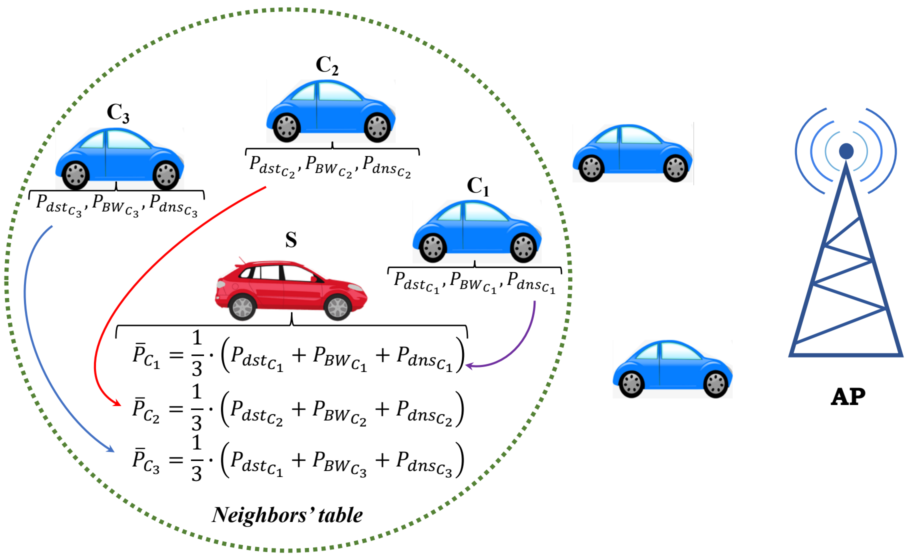

C₂
$P_{dstc_2}, P_{BWc_2}, P_{dnsc_2}$
C₃
$P_{dstc_3}, P_{BWc_3}, P_{dnsc_3}$
C₁
$P_{dstc_1}, P_{BWc_1}, P_{dnsc_1}$
S
$\bar{P}_{C_1} = \frac{1}{3} \cdot (P_{dstc_1} + P_{BWc_1} + P_{dnsc_1})$
$\bar{P}_{C_2} = \frac{1}{3} \cdot (P_{dstc_2} + P_{BWc_2} + P_{dnsc_2})$
$\bar{P}_{C_3} = \frac{1}{3} \cdot (P_{dstc_1} + P_{BWc_3} + P_{dnsc_3})$
Neighbors' table
AP

FIGURE 5.6: Updating the neighbours' list for vehicle S. An access point (AP) is the destination node of the packets.

Fig. 5.6 shows the three probability values (P_{dst}, P_{dns}, P_{BW}) calculated with Eqs. (5.1), (5.2) and (5.3), respectively. Those values are computed at each neighbour N_{gh} within the vehicle's transmission range. In this example, the destination is an access point (AP) used to communicate with the city's infrastructure.

The probability-based forwarding decision is taken hop-by-hop based on the information gathered in the neighbours' table. Vehicles follow our routing algorithm to choose the optimal next forwarding node. In the example depicted in Fig. 5.6, we can see a sender node S that periodically receives beacon messages from its neighbours in transmission range, (C_1, C_2 and C_3). Node S updates the table with the information received from its neighbour nodes. After that, the sender node evaluates those nodes and calculates the probabilities of successfully deliver the packet at destination through each one of the available candidates.

First, the node S currently carrying the packet (sender or intermediate forwarding node) calculates the probability of success in the delivery of the packet at destination, by applying Eqs. (5.1) and (5.2); and the probability of success in the delivery of the packet to the next-hop node using Eq. (5.3). Second, we apply an arithmetic mean on the three probabilities obtained in the previous step as follows:

$$\overline{P}_{Ngh} = \frac{1}{3} \cdot \sum_{i=0}^{3} P_i = \frac{1}{3} \cdot (P_{dst_i} + P_{BW_i} + P_{dns_i}) \tag{5.4}$$

Finally, the score value $\overline{P}_{N_{gh}}$ for each candidate node N_{gh} is obtained using Eq. (5.4) and the node with the highest score will be chosen to forward the packet.

5.4.4.1 Updating the neighbours' table

Each node updates its neighbours' table every time a beacon message (BM) is received. The sending period of BMs is set to be one second. The neighbours' table is sorted according to the probability score $\overline{P}_{Ngh}$. To update the neighbours' table, we follow the process described in algorithm 2. Then, nodes are arranged according to the probability-based score value described in algorithm 3.

Algorithm 2: Updating the neighbours' table.

```
Require: A new beacon message received with parameters: $ID$ of the $N_{gh}$, location
         $(x, y)$.
1  Start: $i = 1$
2  while $i <= \#$ list of neighbours do
3      review the neighbour Id;
4      if neighbour (ID) is in the neighbours' list then
5          Update the information;
6      else
7          Add the neighbour in the neighbours' list;
8      end
9      $i$++;
10 end
```

When a source node S needs to send a message to the AP (e.g., a warning message concerning an accident), it will first check in its neighbours' table if the destination node (i.e., the AP) was previously registered in the last beacon reception. If positive, the message will be delivered to the neighbour node with that destination ID. Otherwise, the neighbour node with the highest probability to success in the packet delivery at destination, will be chosen to forward the packet. This process will be repeated hop-by-hop until the packet reaches its destination.

5.4.5 Accurate estimation of the current nodes' position

In this section we describe an improvement to accurately estimate the position of nodes at the precise moment of sending a message in a VANET. Estimating the nodes' position is a strategy included in routing algorithms as a key parameter to improve the selection of forwarding nodes, trying to deliver the packet to destination through the best forwarding nodes. In that sense, we published a research work where related works were identified and a comparison was made with some of them to see if our proposal to estimate the current node's position improves the performance of the routing protocols [40].

Routing protocols, such as the ones described in Section 5.3, use information exchanged by nodes through the periodical beacon process. This information will be used to take forwarding decisions. Specifically, nodes can calculate or estimate the positions of the neighbouring nodes. Nevertheless, nodes only update their neighbours' table upon the

Algorithm 3: Sorting the neighbours' list according to the probability-based score.

Require: A new beacon message received with these parameters: ID of the N_{gh}, location (x, y), BW_{Ngh} and $NV_{N_{gh}}$

```
Start: $i = 1$ $curr = head$ # initial position
while $i <= \#$ list of neighbours do
    $tmp = curr$;
    $curr = curr \rightarrow next$;
    read current;
    if (curr! = NULL) then
        Calculate: $curr(Probability_{metric})$; $tmp(Probability_{metric})$;
        Score1 = mean $(curr(Probability))$;
        Score2 = mean $(tmp(Probability))$;
        if (Score1 > Score2) then
            $head=tmp$;
            $tmp=curr$;
            $curr=head$;
        end;
    end;
    $i$++;
end;
```

reception of a new beacon message (BM). This means that this information used to take the forwarding decision to send the current packet corresponds to the moment when the last beacon message was received, which might lead not to take the best forwarding decision at that moment. This is the key point of our algorithms that makes it different from other previous proposals. Notice that vehicles might have moved a considerable distance since the moment they sent the last beacon till the current moment when they need to send a packet. Besides, we attain better results at the cost of insignificant computational cost.

Let us consider a moment $t_1 + \Delta t$ when a source node S needs to send a message. Let us assume that the last BM was received at moment t_1. The S node will look for the packet's destination in its neighbours' list. If the destination is not in the neighbourhood, S looks for a proper neighbouring node to forward the message. Considering a beacon period $T_b = t_2 - t_1$ (set to one second in our simulations), let Δt *be* $0 \leq \Delta t \leq T_b$, as it is shown in Fig. 5.7.

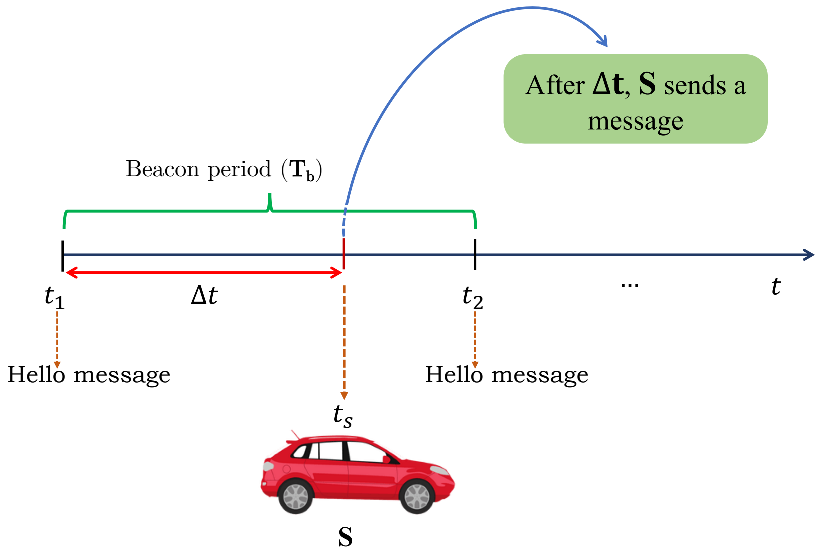


FIGURE 5.7: Time line representation of the beacon interval $T_b = (t_2 - t_1)$ and the event of sending the message at moment $t_s = t_1 + \Delta t$.

The information concerning the positions for each candidate node is taken from the last beacon received from each candidate node at moment t_1. This information will not be updated until the reception of the next beacon message at time t_2. Hence, the current node uses the previous known position information of each candidate node at time t_1 to select the next forwarding node to which it will send the message at moment $t_s = t_1 + \Delta t$. We claim that it is possible to accurately estimate the real position of the candidate nodes at moment t_s (current forwarding moment). The goal is to potentially select a better candidate node. We have included this new feature in our proposal ProMRP, resulting in a new improved version named enhanced ProMRP (EProMRP).

It is well-known that in the Euclidean space $\mathbb{R}^2$, the distance d between two points (x_1, y_1) and (x_2, y_2) is given by:

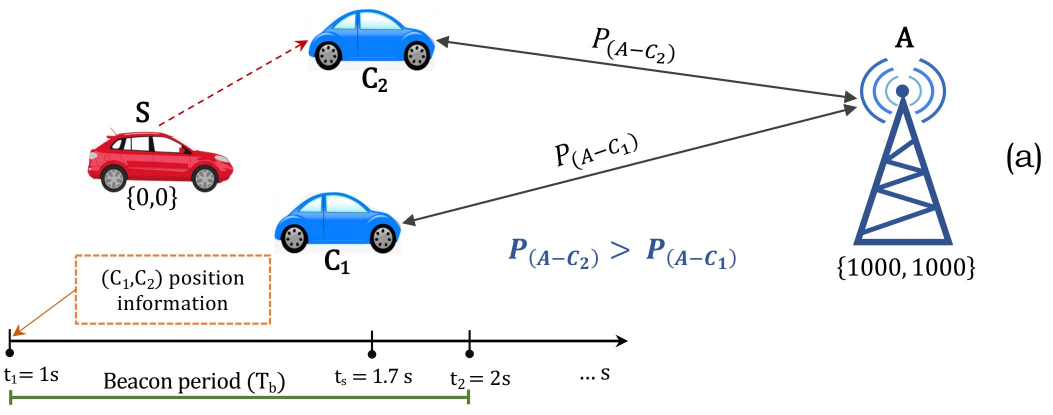
$$d = \left(\sum_{i=1}^{2} |(x_i - y_i)|^2 \right)^{\frac{1}{2}} \tag{5.5}$$

Let us suppose a sender node S located at position $(0, 0)$ needs to send a packet at moment $t_s = t_1 + \Delta t = t_1 + 0.7s$ to an access point A located at position $(1000, 1000)$, see Fig. 5.8. Let us consider that S has two candidate nodes C_1 and C_2 to forward the packet. Let us assume that the last beacon messages from both candidate nodes C_1 and C_2 were received by S in a certain moment within the last beacon period at moment $t_1 = 1s$. Then, at moment t_1 our proposal ProMRP would compute the probabilities $P_{(A-C_1)}$ and $P_{(A-C_2)}$ for distance metric of successful delivery the packet at destination using Eq. (5.1). In this equation, the distances between each candidate node and the access point A are computed using the nodes' positions (x_1, y_1) and (x_2, y_2) reported in their last beacon messages, as shown in Table 5.2. In this example $P_{(A-C_2)} > P_{(A-C_1)}$, so

ProMRP chooses C_2 as the next forwarding node to send the packet at moment $t_s = 1.7s$ (see Fig. 5.8 (a)).

TABLE 5.2: Parameters of two candidate nodes reported in their last beacon messages.

Candidate node	**x (m)**	**y(m)**	v_x **(m/s)**	v_y **(m/s)**
C_1	$x_1 = 500$	$y_1 = 500$	$v_{x_1} = 20$	$v_{y_1} = 20$
C_2	$x_2 = 510$	$y_2 = 510$	$v_{x_2} = 2$	$v_{y_2} = 20$



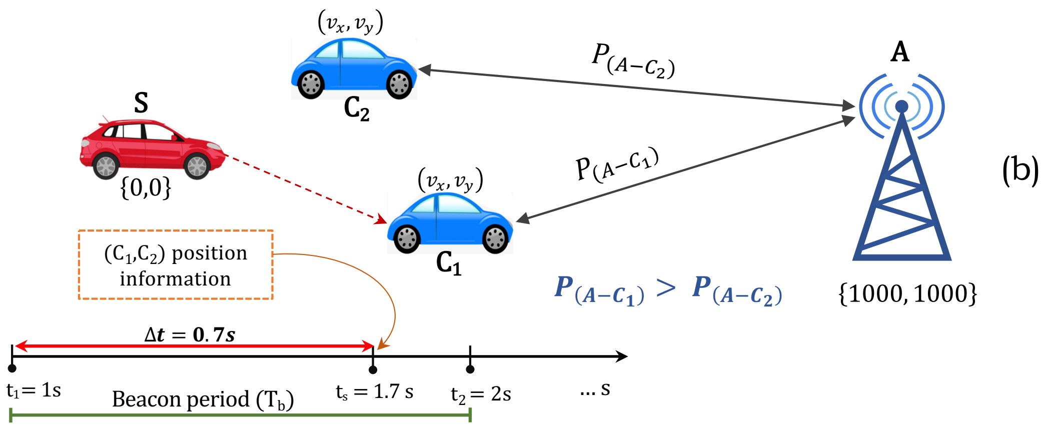


FIGURE 5.8: Selection of a candidate node according to (a) ProMRP or (b) EProMRP.

Nevertheless, if we take into account the v_x and v_y speeds of each one of the candidates to more accurately estimate their current positions at the exact forwarding moment ($t_s = 1.7s$) when the packet is intended to be sent, we would be able to take a better forwarding decision. Probabilities $P_{(A-C_1)}$ and $P_{(A-C_2)}$ of packet successfully delivered at destination as a function of the distances till destination, will be thus computed using Eq. (5.1) by applying the estimated current positions of C_1 and C_2. Those current estimated positions will be used instead of the last reported ones via beacon messages (see Table 5.2). Notice that the reported positions could be far from the current estimated positions at the sending moment $t_1 + 0.7s$. Thus, the reported positions could lead to a wrong selection for the best forwarding node. According to the example depicted in Fig. 5.8 (b), EProMRP would detect that at $t_s = 1.7s$, $P_{(A-C_1)} > P_{(A-C_2)}$. Therefore, C_1 would be selected as the best forwarding node instead of C_2.

Notice that at time $t_1 + \Delta t$, P_{dns} and P_{BW} remain with the same previous value for both C_1 and C_2 candidates. Therefore, the only term that changes is the one based on the distance (P_{dst}) by using the estimated current positions in Eq. (5.1). Hence, the final probability score ($\overline{P}_{N_{gh}}$) in Eq. (5.4) will be different from the previous one obtained using the last beacon received at t_1.

The benefits of our EProMRP proposal have the cost of a slight additional overhead. To perform the EProMRP operation, we need to add two additional fields (16 bytes each field) in the beacon message to allocate the speed's coordinates (v_x, v_y) of the forwarding candidate (see Table 5.2), together with the node's position (x_1, x_2) computed in the last BM sent. Nonetheless, this additional overhead is small and it payoffs the outcome shown since the source node S will be able to accurately estimate the current candidates' positions in the forwarding decision moment, see Fig. 5.8 (b).

For high mobility scenarios, an estimation on the current position will improve the performance of any routing protocol in which its decision to choose the best forwarding node depends completely or partially on the distance to destination metric [3], [67]. When node S needs to forward a packet, it must estimate the current position of the forwarding candidate nodes according to the equations described below, expressed for candidate node C_1 in Fig. 5.8:

$$x_{est_1}(\Delta t) = x_1 + v_{x_1} \cdot \Delta t, \quad 0 < \Delta t < T_b \tag{5.6}$$

$$y_{est_1}(\Delta t) = y_1 + v_{y_1} \cdot \Delta t, \quad 0 < \Delta t < T_b \tag{5.7}$$

where (x_{est_1}, y_{est_1}) is the estimation of the current position of candidate node C_1 at the sending moment used by the node currently carrying the packet (node S in Fig. 5.8). The node's speed is represented by $(v_{x_1},\ v_{y_1})$; Δt refers to the interval time elapsed since the last beacon from that candidate node was received; T_b is the beacon period.

To select the next forwarding node for a packet, our algorithm arranges the neighbouring vehicles of the node currently carrying the packet in a list, according to the score value $\overline{P}_{N_{gh}}$ for each candidate node N_{gh} obtained with Eq. (5.4). In that equation, our proposal EProMRP outperforms the computation of the term P_{dst}, i.e. the probability of successful delivery at destination as a function of the distance, for each candidate node. Those probabilities are computed with the estimated current positions of each candidate node.

The estimation of the node's position will be activated at the moment when a node needs to send or forward a warning message (see Algorithm 3). Each current distance position of the neighbour list is estimated and added to the distance probability computation. Now, the distance probability values of the nodes correspond to their updated positions. Therefore, the probability score could be more precise.

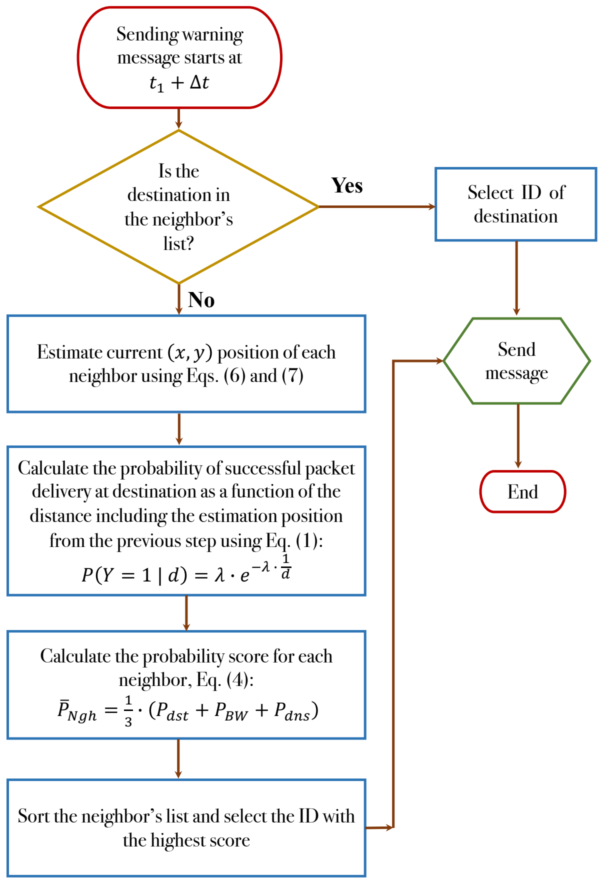

Sending warning message starts at $t_1 + \Delta t$
Is the destination in the neighbor's list?
Yes
Select ID of destination
No
Estimate current (x, y) position of each neighbor using Eqs. (6) and (7)
Send message
Calculate the probability of successful packet delivery at destination as a function of the distance including the estimation position from the previous step using Eq. (1): $P(Y = 1 \mid d) = \lambda \cdot e^{-\lambda \cdot \frac{1}{d}}$
End
Calculate the probability score for each neighbor, Eq. (4): $\bar{P}_{Ngh} = \frac{1}{3} \cdot (P_{dst} + P_{BW} + P_{dns})$
Sort the neighbor's list and select the ID with the highest score

FIGURE 5.9: Flow chart that shows the node selection sequence at moment $t_1 + \Delta t$.

As it is shown in Fig. 5.9, every time a message is sent (or forwarded), the node's position estimation will be done in those nodes that hop-by-hop forward the packet through the network until reaching destination. Thus, with EProMRP, all forwarding decisions are taken with a more accurate estimation of current neighbours' positions in the forwarding decision moment (see Fig. 5.7). Next, ProMRP and EProMRP simulations results are presented below.

5.5 Simulation results

In this section, we describe the simulation scenario to carry out the performance evaluation of our proposal and to discuss the results. Table 5.3 describes the main simulation settings of the urban scenario considered. All figures show confidence intervals (CI) of 95% obtained from five simulations per point, with each simulation having an independent mobility scenario. Simulations were conducted over VEINS [68], an open source inter-vehicular communication simulation framework, together with an event-based network simulator (OMNeT++) [69] and a road traffic simulator (SUMO) [9]. We implemented 3MRP [4] and GPSR [3] routing protocols for VANETs in the VEINS framework, in order to compare their performance with our proposals ProMRP and EProMRP. To carry out the simulations in a realistic scenario, we used a real city area obtained from the

Eixample/Gracia districts of Barcelona and we imported the real map from the OpenStreetMap [5] platform.

Table 5.3: Simulation settings of the VANET scenario.

Map Zone	Eixample/Gracia Distric of Barcelona
Area	2300 m x 2100 m
Density of vehicles	50 and 100 vehicles/km^2
Transmission range	340 m
Mobility generator	SUMO v.25 [70]
MAC specification	IEEE 802.11p
Nominal bandwidth	6 Mbps
Source bit rate	CBR (2 pkts/s)
Beacon interval	1 s
Simulation time	100 s
Routing protocols	GPSR [3], 3MRP [4], ProMRP, EProMRP
Vehicles' speed	20 km/h to 50 km/h
Metrics	distance to destination, vehicles' density and bandwidth
Simulation tool	VEINS [68] / OMNeT++ [69]

We analyzed the performance of our probability-based routing protocols compared to 3MRP and GPSR. The simulation area was 2300 $\times$ 2100 m. We have considered two vehicles' densities of 50 and 100 vehicles/km^2 with vehicles randomly positioned in the map. The vehicles' speeds are between 20 km/h and 50 km/h. Vehicles send packets to a fixed destination (an access point, AP), through which vehicles are able to report traffic information. Messages are forwarded using vehicle-to-vehicle communications in a multi-hop way until the message reaches the infrastructure (i.e., the AP). In our particular case we consider a warning message alerting on the existence of a traffic accident. We assume a crashed source vehicle (its sensors detected the accident) that sends a warning message concerning the accident in the scenario (see Fig. 5.10).

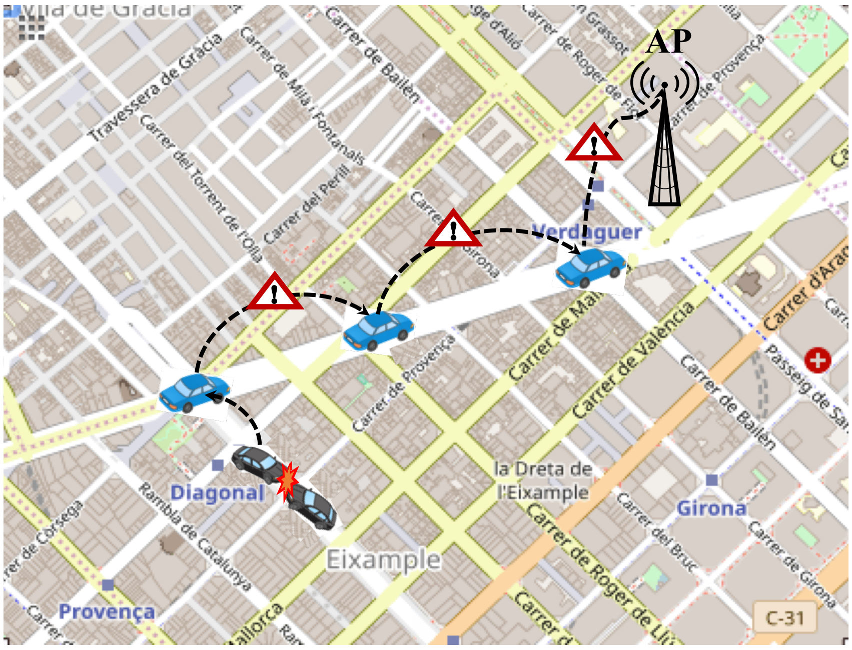


FIGURE 5.10: Simulation scenario of Barcelona. It includes an access point (AP) located in a principal avenue of the city, Av. Diagonal. Eixample/Gracia district map of Barcelona imported from the OpenStreetMap [5]

5.5.1 Average packet losses

Fig. 5.11 shows the average packet losses for our proposals ProMRP and EProMRP compared to GPSR and 3MRP, for different vehicles' densities. When the vehicles' density is low, their distribution is sparse and finding next forwarding nodes is not an easy task. For the higher considered vehicles' density (100 vehicles/km^2), the average packet losses decreases, since network connectivity improves compared to the sparser case (50 vehicles/km^2). This is true upon a maximum vehicles' density above which collisions increase overmuch and affect the packet transmission process. This threshold is around 120 vehicles/km^2 as it was shown in Fig. 5.4. We can see that the worst performance is shown by GPSR. Additionally, we can see that considering various metrics (3MRP, ProMRP and EProMRP) packet losses decrease in a considerable way compared to GPSR. Besides, by including our probabilistic distribution model for each considered metric in order to score each candidate node using Eq. (5.4) and take the corresponding forwarding decisions (ProMRP), losses decrease 50% with respect to 3MRP in the high density scenario.

The reason is that ProMRP achieves a more accurate selection of the best next-hop nodes. Consequently, the performance of ProMRP clearly outperforms 3MRP and GPSR in terms of packet losses. Furthermore, our proposal EProMRP is able to accurately correct the position of the candidate neighbours at the moment of sending the message. We can clearly see in Fig. 5.11 the benefits of including the estimation of candidates' positions on the obtained results, specifically in terms of lower packet losses. This improvement is important in those critical situations when the density of nodes is low.

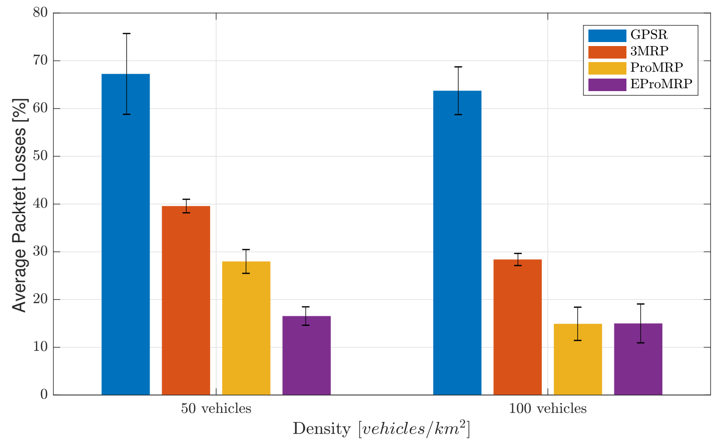


FIGURE 5.11: Average packet losses. Our proposals ProMRP and EProMRP clearly improve GPSR and 3MRP.

5.5.2 Average end-to-end packet delay

Fig. 5.12 shows the average end-to-end packet delay for the evaluated routing protocols (i.e., GPSR, 3MRP, ProMRP, and EProMRP) under both considered vehicles' densities. For a low density scenario, the node currently carrying the packet has a low number of neighbour nodes so the routing paths are unstable and unreliable producing higher losses. Nonetheless, considering more metrics than just the basic distance to destination (as the basic GPSR) we are able to guarantee a higher percentage of messages reception as shown in Fig. 5.11, while also keeping low the average end-to-end packet delay as Fig. 5.12 shows. However, although our both proposals ProMRP and EProMRP get similar results with respect to GPRS and 3MRP in terms of average packet delay, their percentage of packet losses is notably lower.

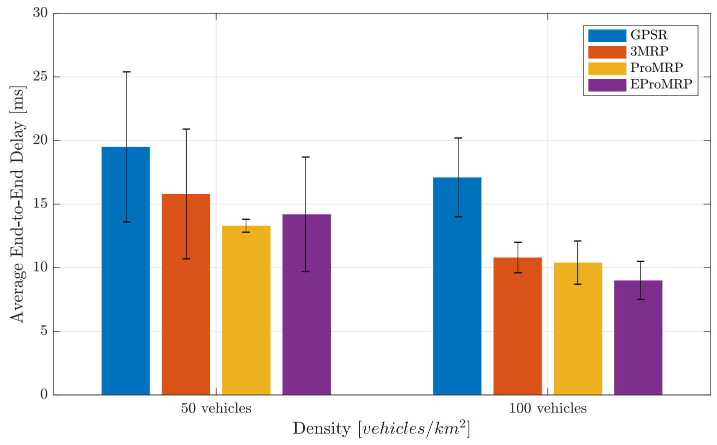


FIGURE 5.12: Average end-to-end delay. Our proposals ProMRP and EProMRP show similar delays than 3MRP.

On the other hand, including in EProMRP the correction in the node's position at the moment of sending a packet, shows that the choice of forwarding nodes is more efficient, specially in the high vehicles' density case. The delay in the low vehicles' density case also keeps a similar average delay compared to ProMRP.

5.5.3 Overhead

Fig. 5.13 shows the overhead incurred by each one of the routing protocols analyzed in this chapter. GPSR is the protocol with the lowest overhead (13%), although it produces the highest percentage of packet losses (around 65%) and the highest average packet delay (around 18 ms), according to figures 5.11 and 5.12, respectively. Both, 3MRP and ProMRP increase the overhead to 17% since they use two additional fields (16 bytes each field) in the hello messages to carry the three additional metric values (node's position, available bandwidth and nodes' density) used in the forwarding algorithm together with the distance to destination metric (already included in GPSR). Nevertheless, the benefits in packet losses and packet delay are notable as figures 5.11 and 5.12 show. Our proposal ProMRP shows packet losses around 15-25% and packet delays around 15-17 ms. Finally, our proposal EProMRP achieves the best performance in terms of packet losses (around 15%) and packet delay (around 10-15 ms), with a slightly higher amount of overhead around 21%.

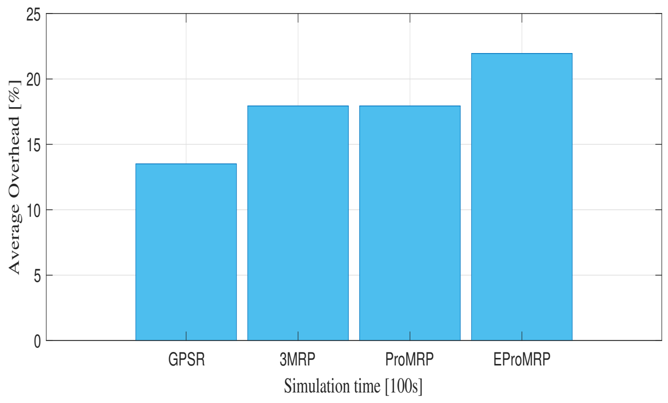


FIGURE 5.13: Routing protocol overhead of our proposals ProMRP and EProMRP compared to 3MRP and GPSR.

5.5.4 Computational cost and simulation time

Fig. 5.14 depicts the computational cost incurred by our proposals ProMRP and EProMRP, compared to GPSR and 3MRP. To compute the computational cost of each routing protocol, we made an analysis of the number of operations and time incurred by the forwarding algorithms used by the four routing protocols. The procedure used to derive the computational cost is detailed in Section 2.4.4. The computational cost of the forwarding algorithms used in GPSR and 3MRP is under 0.01 milliseconds, as it is shown in the zoom box of Fig. 5.14. Instead, our proposed routing protocols increase this computational cost in the ranges of 0.1 to 1 milliseconds, with a peak that reached 3 ms for a few packets. The slightly higher delay is the cost of the additional operations included in the forwarding algorithms. Nevertheless, that extra delay is still very low and it pays-off the benefits of our proposed routing protocols ProMRP and EProMRP.

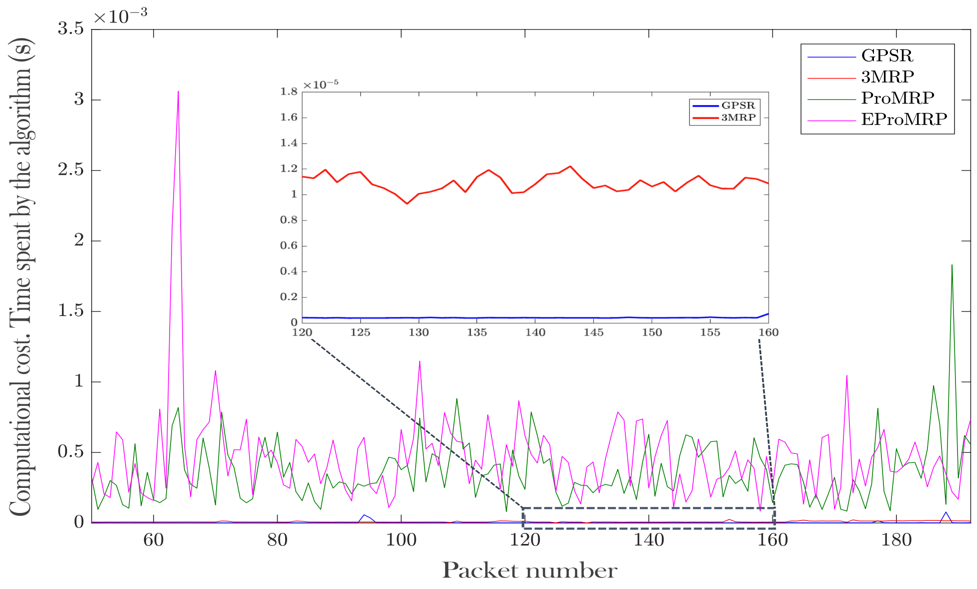


FIGURE 5.14: Computational cost squence of the routing protocols: GPSR, 3MRP, ProMRP and EProMRP.

5.6 Conclusions

In this chapter, we present a new routing protocol named probabilistic multi-metric routing protocol (ProMRP) for VANETs specially designed for urban scenarios. ProMRP includes three metrics (vehicles' density, distance to destination and available bandwidth) to take forwarding decisions based on a probabilistic scheme.

The probabilistic models to evaluate the three metrics have been derived previously offline. After that, our proposals use the obtained models to asses the three considered metrics in order to arrange the candidate nodes in the neighbourhood of the node currently carrying the packet. In this way, the protocol chooses as next forwarding node the best one that ensures the delivery of the packet with the highest probability, while keeping low the average packet delay. Besides, an algorithm to accurately estimate the current node position in the forwarding moment was included in the version named EProMRP. This new proposal further improves the performance in terms of lower losses, while keeping similar average end-to-end packet delays.

The modeling of the probability density functions of the three considered metrics was done based on a previous offline analysis from a significant number of representative simulations in urban scenarios. Thus, we can conclude that our proposal takes better forwarding decisions that guarantee the packet delivery at destination with higher probability than the other proposals evaluated.

Our algorithm could be adapted and implemented in any VANET routing protocol that uses a multi-metric algorithm to select the best next-hop node to forward information. Our EProMRP proposal not only corrects the node's current position when sending the packet, but it also takes the best forwarding decision to successfully guarantee the delivery of the packet.

Overall, this solution has shown the benefits obtained when using the forwarding decision based on a packet arrival success probability score. Furthermore, including an estimate of the node's current position increases routing efficiency in terms of packet loss with minimal cost. In the next chapter some classic machine learning models and a basic neural network are presented.

Chapter 6

Overview of the machine learning models (ML) used in our research

Machine learning has attracted increasing interest from the research community in the last decade until it becomes nowadays one of the hottest science topics in several fields.

Machine learning algorithms are able to extract relevant information hidden in big data that capture the complex relationship exhibited by inputs and labels. Nowadays, the most active topic of machine learning is deep learning. Recently, deep learning models have been used in many domains such as load forecasting, computer vision, and speaker recognition, among others. The exponential growth of data is helping deep learning techniques to provide higher performance solutions compared to the most learning algorithms. Additionally, training deep learning models on a huge amount of dataset could be extremely efficient by using available high-performance computing devices such as GPU or CPU cluster. Moreover, deep learning allows computational models to learn representations of data with multiple levels of abstraction. This approach provides more relevant information that will improve decision making processes.

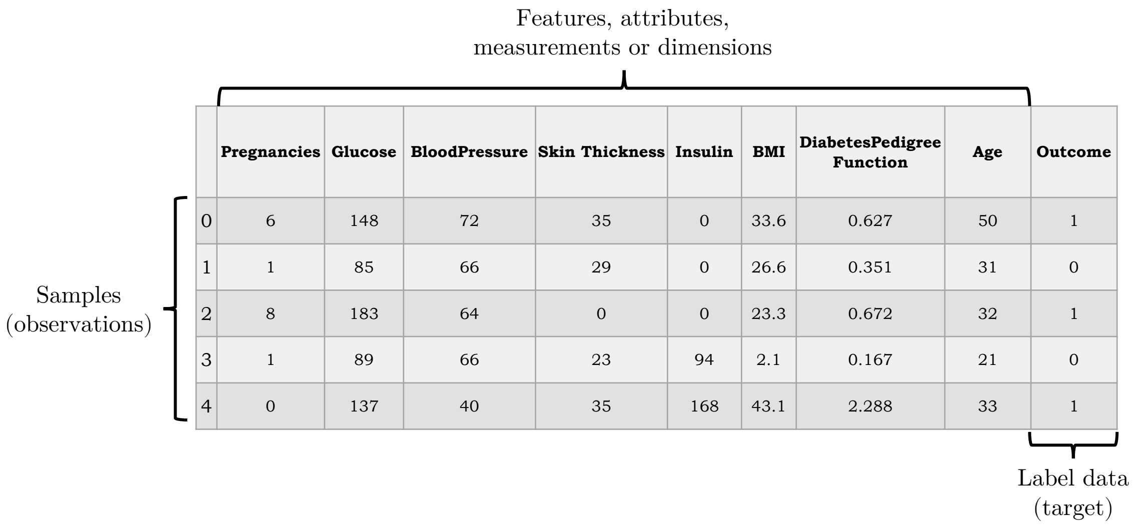
A thorough overview of machine learning algorithms is presented in this chapter. Traditional machine learning algorithms (i.e., decision tree, etc.) as well as artificial neuronal networks (i.e., Feedforward, CNN, etc.) are explained. Several performance metrics of machine learning models are presented and extensively explained. Finally, conclusions of this chapter are drawn at the end.

6.1 Introduction

Machine learning (ML) is a research field that deals with the theory, performance, and properties of learning systems and algorithms. Concretely, it is a tool that builds a computational model based on an observed past experience. In ML, the learning process corresponds to adjust values for specific parameters in such a way that the model matches best with the data it has already seen during the training stage.

The process of learning from data is called training process. Based on this training process, a general model through a particular set of parameters becomes specialized to the particular tasks that underlie the data. The excellent performance of ML algorithms depends on how much and how good the amount of data (dataset) is. Therefore, data collection is an essential step, since representative datasets vary not only from one problem to another but also from one time period to the next one. This data collection will depend on the type of problem to solve and it can be achieved from various repositories or by using monitoring and measurement tools [71].

Before starting building a machine learning model, the data needs to be organized and structured. A simple example of an organized and structured diabetes dataset is shown in Figure 6.1). The aforementioned dataset is composed of the measurements (samples or observations) for a set of patients [72]. Each sample contains the features (attributes, measurements or dimensions) and the labeled data (output). Features are the predictors used to train a ML model. For instance, the patient measurements (columns 2-9) will be the inputs of the ML model (features), and the remaining column will be the outcome (label data). The output represents if the patient is diabetic ("1") or not ("0").


Features, attributes, measurements or dimensions
Samples (observations)
Label data (target)

	Pregnancies	Glucose	BloodPressure	Skin Thickness	Insulin	BMI	DiabetesPedigree Function	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	2.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

FIGURE 6.1: Example of an extraction of some features values of diabetes dataset.

The general process flow to build ML algorithms is shown in Figure 6.2. Each block of the process will be extensively explained as follows:

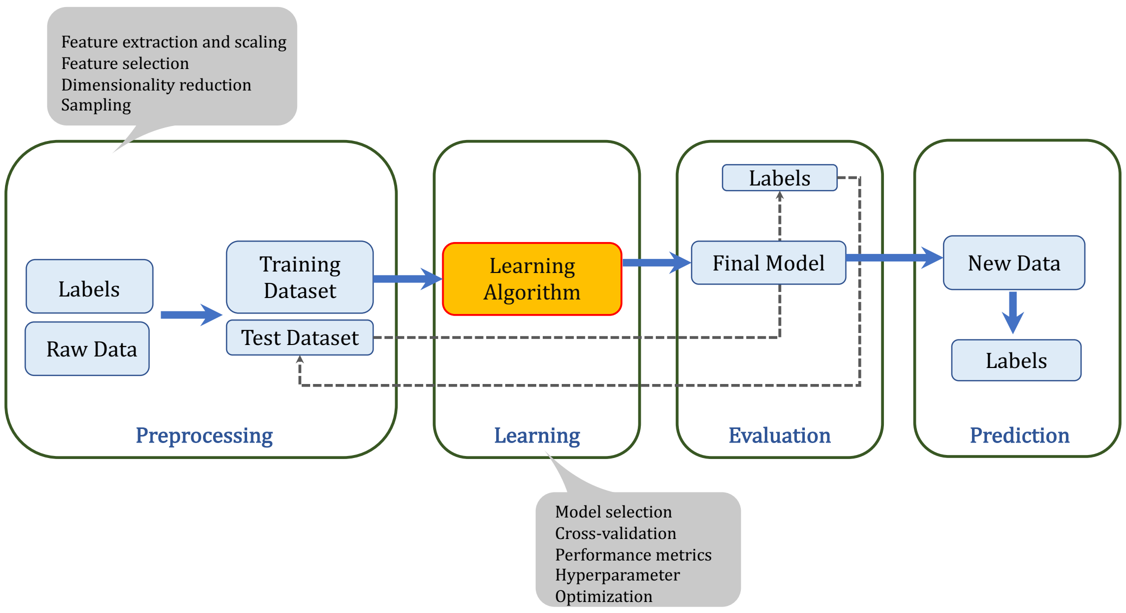


FIGURE 6.2: Process flow to build machine learning systems [6]
.

Preprocessing:
Within data preprocessing, we are able to organize and transform data into a suitable format used for machine learning purposes. To do so, values of the features should be normalized between zero and one. In some cases, those selected features may be highly correlated, and therefore, they could be redundant to a certain degree. To overcome this issue, several techniques can be used such as principal component analysis (PCA), factor analysis, and linear discriminant analysis (LDA), among others. Once one of these techniques has been applied, learning algorithms can run much faster.

The dataset now is ready to be divided into *training* and *testing* subsets.

i *Training set*: The sample of data used to fit the model. It is used to train and fit the configuration parameters of the machine learning model.

ii *Data validation*: The validation set is used to evaluate a given model, to fine-tune the model hyperparameters (i.e., parameters whose values are set before the learning process begins) of the ML model [8].

iii *Testing set*: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. It is used to evaluate the final model. A set of unseen data used only to assess the performance of the model.

About the dataset splitting ratio among the three phases (see Fig. 6.3), it mainly depends on two things: (i) the total number of samples in the dataset and (ii) on the model we are training. Furthermore, models with very few hyperparameters will be easy to validate and tune, so we can reduce the size of our validation set. For instance, in our decision tree algorithms (see section 6.2.2 we have two hyperparameters: tree depth and tree breadth; in our ANNs (see section 6.3 and chapter 7) we have two hyperparameters: number of hidden layers and number of neurons. A typical splitting ratio of the dataset into the three subsets, to have an initial reference, is 60% for training, 20% for validation and 20% for testing. Nonetheless, the ratio should be chosen after conducting several tests to see which are the most suitable dataset splitting ratios, taking into account the size of the dataset.

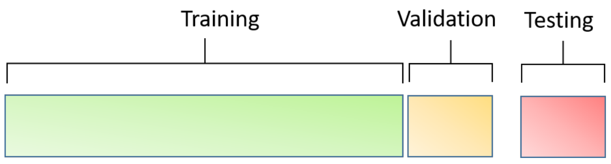


FIGURE 6.3: Dataset spliting ratio.

Learning:
In this phase, different models can be run and compared, and the selection of the best model is based on the values of a set of well-known performance metrics (e.g., accuracy, sensitivity, specificity), which are summarized in section 6.4. Commonly, accuracy is used to have a first idea of the performance of an ML model. The accuracy is defined as the proportion of correctly classified instances. Furthermore, different cross-validation techniques can be used where the training dataset is further divided into training and validation subsets to estimate the generalization performance of the model. Generalization refers to the ability of the model to adequately adapt to new data never seen before, obtained from the same distribution as that used to create the model. A popular cross-validation technique is K-fold, where every sample or record is used both in training and test sets, see Fig. 6.4 [7].

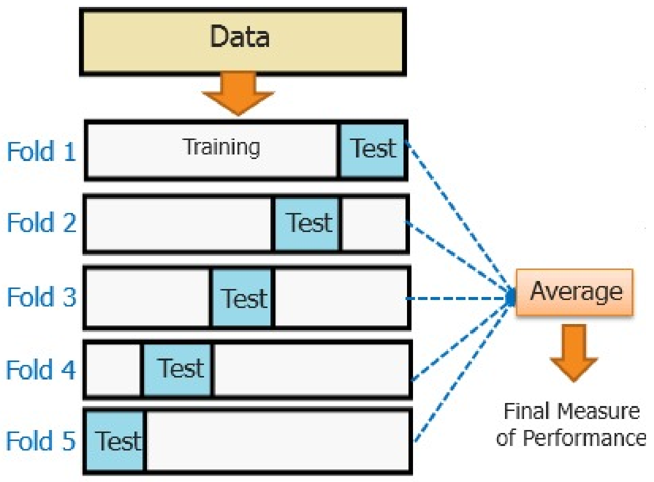


FIGURE 6.4: Example of an K-fold cross-validation method (with $K = 5$) to validate ML models [7].

Evaluation:
This phase deals with the evaluation of the trained model on the testing dataset. Performance metrics will determine whether the trained model is having a good performance or not. Conventional metrics to asses the performance of an ML model are the accuracy and confusion matrix, among others [71].

Prediction:
The final stage of a machine learning model consists in using the model to predict the output of a given problem.

In the next sections, many types of representative machine learning models will be described.

6.2 Machine learning models

This subsection will briefly summarize several machine learning concepts. More specifically, we will classify ML algorithms and describe two classical learning classification schemes. Obviously, there are many more ML models published in the literature. However, we just selected those with which we have compared our novel methodology to take the forwarding decisions based on ML models. For more details about ML models in general, several books [6, 73, 74] provide an extensive description and detailed explanations.

6.2.1 Classification of machine learning algorithms

Learning process is commonly classified in three categories, see Fig. 6.5: (i) *supervised*, (ii) *unsupervised* or (iii) *reinforcement learning*. This classification is based on how the data is used to train the models, as it is explained below:

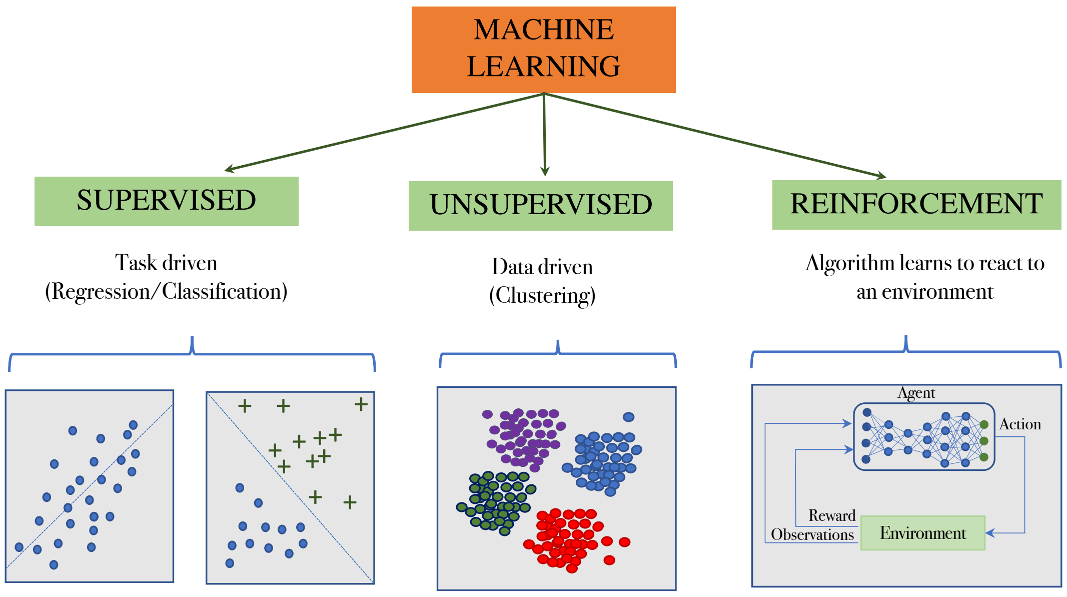


FIGURE 6.5: Examples of classical learning classification.

- **Supervised learning:** Supervised refers to a set of samples where the desire output patterns (labels) are already known [75]. Examples of supervised algorithms are logistic regression (LR), naive bayes, support vector machine (SVM), artificial neuronal networks (ANN) and random forest (RF). Application areas of this type of learning are fraud detection, e-mail spam detection, diagnostic and image classification.

- **Unsupervised learning:** This type of learning does not require labels to the observations. This approach is most suited for clustering problems to group data that has not been labelled, classified or categorized [71]. Application areas of this type of learning are dimensionality reduction e.g. principal components analysis (PCA), anomaly detection, big data visualization to make it easy to show and analyse large amounts of data.

- **Reinforcement learning:** It enables learning from the feedback received through interactions with the external environment. Here, an agent would learn in an interactive environment by trial and error using feedback from its own actions and experiences. The goal is to find a suitable action that will maximize the total cumulative reward of the agent [71, 76]. Examples of this kind of learning are Markov decisions process and Q learning. Application areas include gaming, financial sector, and robot navigation, among others.

6.2.2 Decision tree algorithms

Decision trees (DTs) are a type of supervised classification approach. It is a simple representation to classify inputs, where data is continuously split according to a certain parameter. Its name is given because its structure remembers the shape of a tree. A decision tree is a flowchart-like tree structure composed by: (i) internal nodes representing features (or *attributes*), (ii) *branches* representing decision rules, and (iii) *leaves* representing the outcome.

DTs are made up of a root and nodes, branches (edges), and leaves. Nodes represent circles and the branches segments that connect nodes. Decision trees start from the root node, and moves downwards through branches. The node where the chain ends is named as leaf node. Two or more edges can be extended from each internal node. Root and internal nodes hold a test over a given dataset attribute, and the edges correspond to possible outcomes of the test (i.e., the pre-classified data described in Fig. 6.1). The division into classes is decided upon the features that best divides the data. The data items are split according to the values of these features. This process is applied to each split subset of the data items recursively. The process terminates when all the data items in the current subset belong to the same class [77, 78].

DTs have two configuration parameters that determine the dimension of the tree, see Fig 6.6:

- *Tree depth*: The average number of layers (levels) from the root node to the terminal nodes (leaves). We refer to the average depth of the tree.

- *Tree breadth*: The average number of internal nodes at each level of the tree. We refer to the average breadth of the tree.

Tree depth and tree breadth concepts are indicators of the tree's complexity. That is, the higher their values are, the more complex the corresponding decision tree is [77].

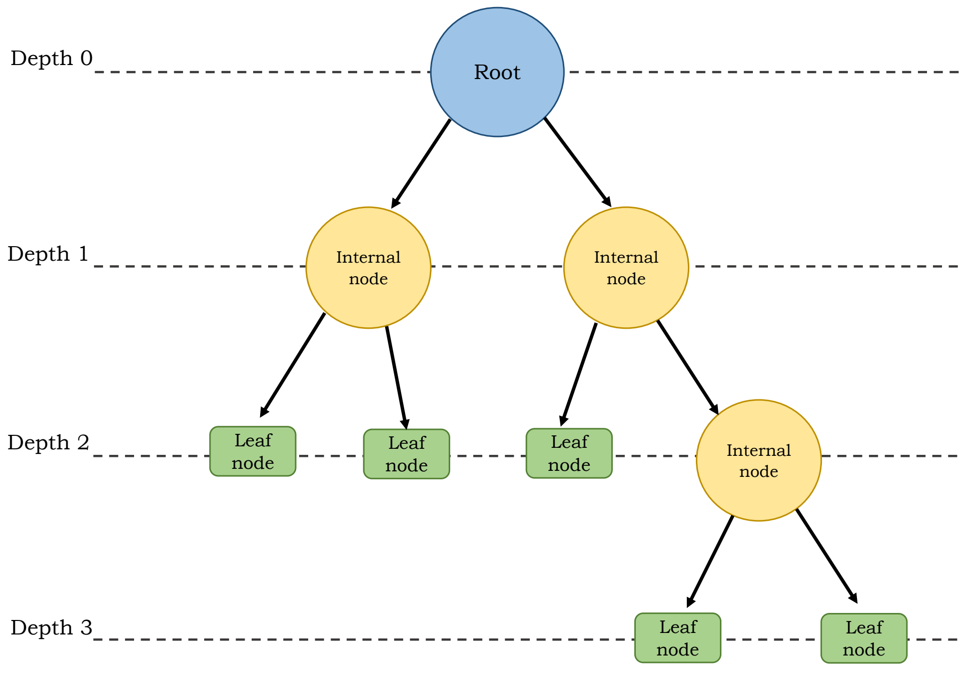


FIGURE 6.6: General structure of a decision tree.

Generally, a decision tree works according to the next sequence of steps [79]:

i. Selection of the best attribute (i.e., the best internal node in the tree) using attribute selection measures (ASM) to split the dataset records. The best attribute will be the one whose knowledge contributes with more information from the classification point of view.

ii. Take a decision of the best attribute according to the previous step, and split the dataset into smaller subsets.

iii. Build the tree by repeating this process recursively for each child (internal node) until one of these conditions match:

- All the tuples belong to the same attribute value.
- There are no more remaining attributes.

The tree shape is build following some split methodology. There are different methods to split decision trees giving a dataset. The most commonly used are: (i) Information Gain, (ii) Gain ratio and (iii) Gini Index [80]. In our research work we have used the Gini index, since it showed a better performance in our tests compared to the other two split methods.

(i). **Information Gain**: The information gain is used to determine which characteristic or attribute (input) of a data set provides the maximum information about a class (output). Its objective is to reduce the level of entropy (degree of uncertainty, impurity, or disorder) from the root node to the leaves nodes.

(ii). **Gain ratio**: It is a modification of the information gain that reduces its bias. It takes a number and size of branches into account when choosing an attribute. It corrects information gain by taking the intrinsic information of a split into account.

(iii). **Gini index**: It is a method that measures the frequency with which a randomly chosen item would be incorrectly identified. This means that an attribute with a lower Gini index should be preferred. The Gini index degree varies between 0 and 1. A value 0 denotes that all the elements belong to a certain class. Alternatively, if there is only one class a Gini index = 1 denotes that the elements are randomly distributed in several classes. Thus, in the building of the decision tree, it is preferred to choose the attribute or characteristic with the lowest Gini index as the root node.

Advantages and disadvantages of decision trees can be summarized as follows:

- **Advantages:** Decision trees are easy to understand and to see how they work. Their tree structure permits to handle both numerical and categorized information. Besides, they are easy to validate with usual performance metrics. Decision trees perform well with reasonable computing power resources, and they work very well with a large set of data.
- **Disadvantages:** A decision tree may create an overlay complex model, which depends on the dataset selected in the training phase. Sometimes the resulting model can not be easy to write (if it is too large) [81].

6.2.3 Bagged decision tree algorithms

Bootstrap aggregated (also called bagged) trees are a type of ensembled algorithms that can be applied when a basic decision tree model does not perform well. Sometimes, individual decision trees do not work well for two main reasons:

- On the one hand, they may have a high bias (under-fitting) that makes the model too simple to adequately capture the underlying structure of the data.
- On the other hand, the model may present too much variance (overfitting) because it has more parameters than can be justified by the data [8].

Bagged decision trees combine the results of several decision trees, which reduces variance and helps to avoid overfitting. Bagging can be applied to tree-based algorithms to enhance the accuracy of the predictions. Basically, the bagging method combines the predictions of several basic learners (small trees) to create a more accurate output. When using bootstrapping with the training dataset, N bootstrap samples are generated (1,2,...N), as it is shown in Fig. 6.7. Each new bootstrap sample will act as another independent dataset drawn for a true distribution. For each bootstrap sample, a classifier $b_i(x)$ is trained. Finally, a combined classifier will average the outputs from all these individual classifiers, and therefore, an ensambled model is obtained with less variance than its individual components.

$$b(x) = \frac{1}{N} \cdot \sum_{i=1}^{N} b_i(x) \tag{6.1}$$

By averaging weak learners' outputs, the expected outcome is not changed but its variance is reduced. Bagging can be used for classification and regression problems, and it is more effective with small datasets [82]

- **Bagging regression**: The final prediction is the average of the predictions of the models that are built over each bootstrap sample.
- **Bagging classification**: Voting is used to make final predictions. That is, the class output by each model (weak-learners or classifier) can be seen as a vote and the class that receives the majority of the votes is returned by the ensembled model.

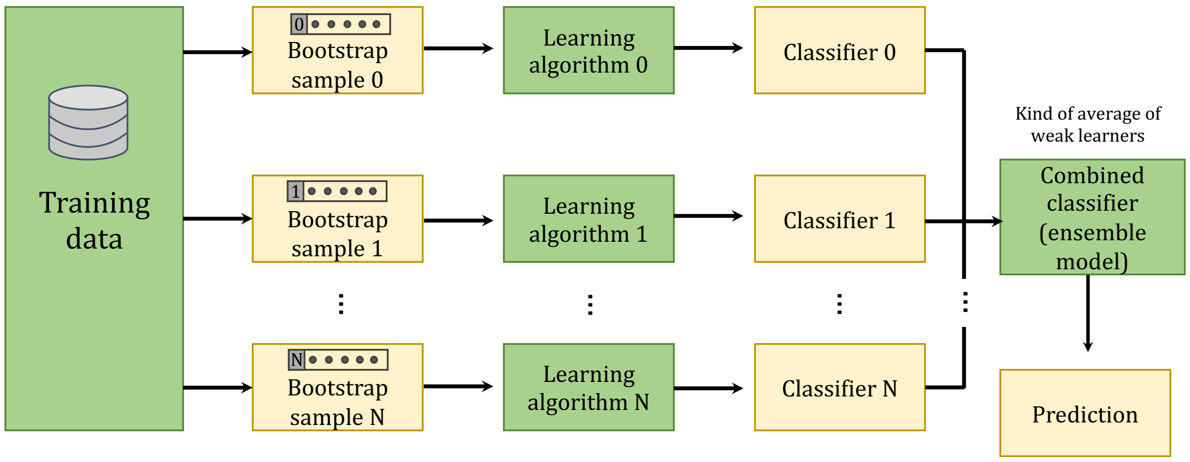


FIGURE 6.7: A diagram of the bootstrap aggregation flow.

In bagging methods, the combination of weak learners in the right way can create more accurate and/or robust models. It is important to highlight the fact that several instances of the same base model are trained in parallel on different bootstrap samples, and they are independent of each other. Then, they are aggregated in some kind of "averaging" process. This averaging operation is done over the (independent and identically distributed -i.i.d.-) fitted models. These bagging methods allows us to obtain an ensambled model with lower variance than its components. For this reason, base models with low bias but high variance are well adapted with bagging techniques. Advantages and disadvantages of bagged decision trees can be summarized as follows:

- **Advantages:** Bagging and ensemble algorithms in general can improve the accuracy of unstable models and decrease the degree of over-fitting.
- **Disadvantages:** Although bagging helps to reduce the variance, it is ineffective in reducing model bias. This method represents a more complex procedure to get a prediction.

In the next section, a more complex method to enhance the performance of ML models is described. They are known as artificial neural networks or deep learning.

6.3 Artificial neural networks

Artificial neural networks (ANNs) are today one of the hottest topics of artificial intelligence and machine learning. ANNs are computational models based on the structure of the brain, whose most significant property is their ability to learn from data. ANNs are a set of algorithms that emulate the basic function of the human neuronal system aiming to recognize patterns. These patterns are numerical, contained in vectors, and they represent real-world data such as images, sound, text and so on.

ANNs can be arranged as sequences of layers. Layers are composed of individual neurons, considered as the basic element in ANNs, similarly as the brain cells. The most widely used neuron model is the *perceptron*. A perceptron performs basic mathematical operations (see Fig. 6.8). Besides, perceptrons can be combined to form a multi-layer network, where nodes are neurons and the edges are connections between neurons.

In the following subsections, we present the ANN structure used in this thesis work. We would like to highlight that in our research work we have used ML techniques to design enhanced forwarding algorithms to be used in our proposals of routing networks in VANETs under urban scenarios. Therefore, we have not deepened in the mathematical knowledge of ML algorithms, but in understanding ML algorithms to attain the required knowledge of these algorithms to build our ML-based models. We have used specialized bibliography to study ML and specifically ANN [74, 83].

6.3.1 General configuration parameters of ANN

As stated previously, the analogy of ANNs to neurons comes from replicating its operation as a mathematical function that maps a given input to the desired output. This structure is known as *perceptron* (see Fig. 6.8). The concept of perceptron was introduced by Frank Rosenblatt in 1957. A perceptron is a simple mathematical function that takes a set of inputs (x_i), performs a mathematical operation (activation function) to combine the input from the data with a set of weights w_i, and outputs the result of that operation in y.

$$y = \sum_{i=1}^{N} (x_i \cdot w_i) \tag{6.2}$$

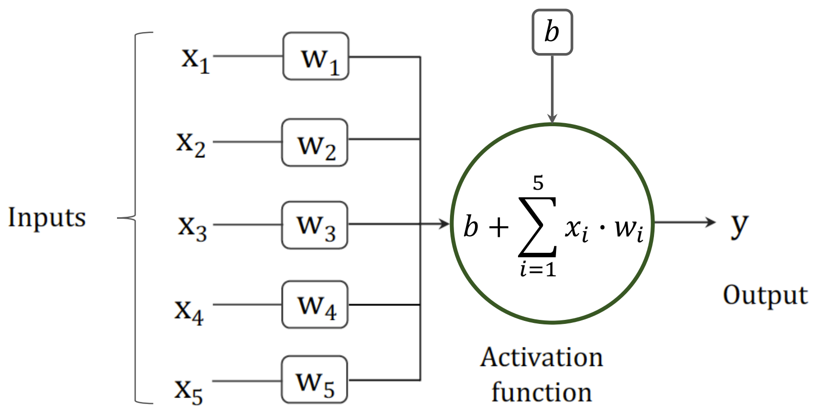


FIGURE 6.8: An example of perceptron with $N = 5$ inputs x_i and one output y.

Multiple perceptrons can be combined to form a layered architecture, usually named multi-layer perceptron (MLP). An MLP or neural network consists of this set of components, as it is represented in Fig. 6.10:

- An *input layer*, x. This layer accepts input features and provides information from the outside world to the network. It includes the numerical values that represent the input data.
- An arbitrary number of *hidden layers*. Nodes of this layer are not exposed to the outer world, they are the part of the abstraction provided by the ANN. These layers perform computation on the features entered through the input layer and transfer the result to the output layer.
- An *output layer*, y. This layer gives the information learned by the network to the outer world.
- A vector of real-valued weights w_i. They represent the strength of the outputs and the connectivity between nodes
- A bias b, which is a real number that adjusts the boundary from origin without any dependence on the input value.
- An activation function σ for each hidden layer. There are many types of activation function. All of them have to satisfy the requirement to be non-linear. See Fig. 6.9 and Table 6.1

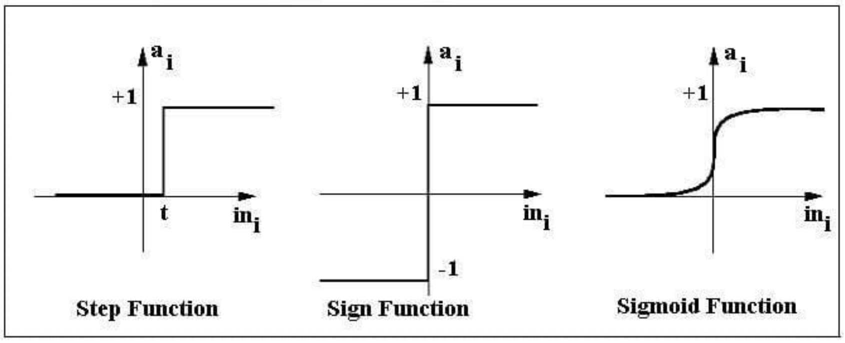


FIGURE 6.9: Examples of activation functions used in neuronal networks.

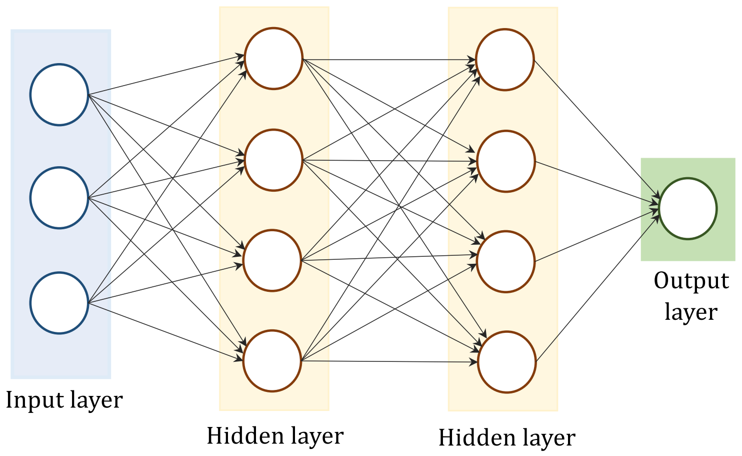


FIGURE 6.10: Example of a fully connected neural network with three layers: one input layer, several hidden layers and one output layer.

Basically, a hidden layer in an ANN is a layer in between input and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function. From figure 6.10 we can see that nodes in the hidden layer are fully connected to the input layer, and the output layer is fully connected to the hidden layers.

6.3.1.1 Activation functions

Activation functions are mathematical equations that determine the output in a neural network. These functions take an input value, transform the input value, and pass the transformed value to the next layer or to the final output layer to make a prediction. In Table 6.1 we present the most common activation function used in ANNs.

Activation function	Formula	Description
Binary step	$f(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases}$	This is a simple threshold base classifier. A threshold value is selected to decide the output, whether the neuron should be activated or deactivated [84].
Linear	$f(x) = a \cdot x$	The output is same as the input. It takes the inputs, multiplied by the weights for each neuron, and creates an output signal proportional to the input in the range of $-\infty$,$+\infty$. The variable a can be any constant real value [85].
Sigmoid	$Unipolar : g(x) = \frac{1}{1+e^{-x}}$ $Bipolar : g(x) = \frac{1-e^{-x}}{1+e^{-x}}$	This function is especially advantageous to use in neural networks that are trained by back-propagation algorithms. The term of sigmoid means "S-shaped curve. The logistic function is a common example of a sigmoid function. The unipolar form maps the interval $(-\infty, \infty)$ in the range $(0, 1)$, whereas the bipolar form is suitable for applications that produce output values in the range $(-1, 1)$ [84].
Hyperbolic tangent function (Tanh)	$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^x}{e^x + e^{-x}}$	It is defined as the ratio between the hyperbolic sine and the hyperbolic cosine functions, which equals to the ratio of the half-difference and half-sum of two exponential functions in the points x and $-x$. It is similar to the sigmoid function and its output range is defined between -1 and 1 [85].
Rectified linear unit (ReLu)	$f(x) = max(0, x)$	It is defined as the maximum value between zero and the input value. That is, the function returns 0 if it receives any negative input, whereas for for any positive value it returns that same value back [85, 86].

Leaky ReLu	$f(x) = \begin{cases} x, & x \geq 0 \\ 0.01 \cdot x, & x < 0 \end{cases}$	It is an improved version of the ReLU function. Instead of defining the function as 0 for negative values of x, it is defined as a small linear component of x. It fixes the problem of dying neurons, allowing a small non-zero gradient when the unit is not active [86].
Parameterised ReLu	$f(x) = \begin{cases} x, & x \geq 0 \\ a \cdot x, & x < 0 \end{cases}$	It is another variant of ReLU that aims to solve the problem of gradient's becoming zero for the left half of the axis. It introduces a parameter as slope of the negative part of the function [87].
Exponential linear unit (ELU)	$f(x) = \begin{cases} x, & x \geq 0 \\ a \cdot (e^x - 1), & x < 0 \end{cases}$	Another smoothed version of ReLU that modifies the negative part of the function slope. ELU uses a log curve to define the negative values [87].
Swish	$f(x) = \begin{cases} x \cdot sigmoid(x) \\ x/(1 - e^{-x}) \end{cases}$	Swish behaves similarly to the ReLU function, but it shows better performance in deeper models. The values for swish range from (-∞, ∞) [88]
Softmax	$\sigma(z)_i = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ $for\ j = 1, ..., K$	It is a combination of multiple sigmoids. It is often used in the final layer of a neural network-based classifier. It is a normalized exponential function that takes as input a vector of K real numbers. Then, the function normalizes the input into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. [86]

Radial basis function (RBF)	$y(x) = \sum_{i=1}^{N} w_i \cdot g(\|x - c_i\|)$	It is a real-valued function whose value depends only on the distance from the origin $g(x) = g(\|x\|)$. Alternatively, the value depends on the distance from some other point c, called center, so that $g(x, c) = g(\|x - c_i\|)$. $y(x)$ is the approximation function; N is the number of radials basis functions; c_i are the different center values; distances are weighted by appropriate coefficients w_i [89].
Conic section function (CSF)	$f(x) = \sum_{i=1}^{N+1} (a_i - c_i) \cdot w_i - cos(w_i) \cdot (\|a - c_i\|)$	It is based on the section of a cone, as the name implies. It takes a parameter to determine the angle value of the function. a_i is an input coefficient, c_i is the center of the function, w_i is the vector of weights used in the ANN, w_i is an opening angle that can be a value in the range of $(-\pi/2, \pi/2)$ and determines the different forms of the decisions borders [89, 90].

TABLE 6.1: Main activation functions used in ANNs.

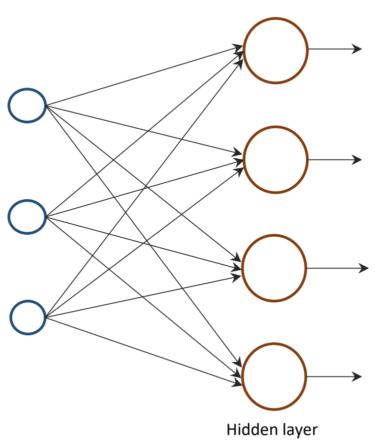
It is important to mention that the hidden units of the neural network need activation functions to introduce non-linearity into the networks. Non-linearity makes multi-layer networks effective. In the simplest case, each neuron performs the same mathematical operations using the same activation function. Alternatively, in more complex neural networks the activation functions used are different in the hidden layers than in the output layers [8, 91]. In our ANN model we have used two activation functions: (i) a sigmoid function in the output layer, and (ii) a simple ReLu function in the hidden layers. Both are well-known and commonly used activation functions that have shown to perform well [90]. In our experiments, they showed a good trade-off between simplicity and good results. In next chapter 7 we will present detailed explanations of our ML-based models to design forwarding algorithms used in our routing protocols for VANETs.

6.3.1.2 Neural network architectures

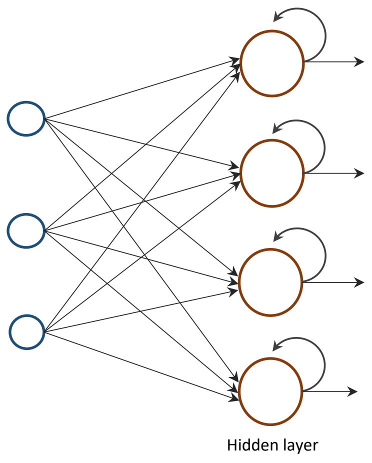
The architecture of an ANN refers to the way that individual artificial neurons are interconnected. It also can be named as topology or graph of the ANN. This interconnection can be done in several ways, resulting in numerous topologies. There is a general classification of ANNs according to the network architecture, in two categories: (i) *feed forward topology* and (ii) *recurrent topology* [85]. A brief explanation of these two classes is presented in the following.

(i). Feed-forward neural network (FNN): This is a graph where the information flows from input to output in only one direction. There are no limitations on the number of layers, activation functions used in each artificial neuron, or the number of connections between them. Therefore, they can be either fully-connected or semi-connected. Common examples are the Perceptron and the MLP. They are used in computer vision, facial recognition and medical diagnosis, among other applications [8, 85].

(ii). Recurrent neural network (RNN): This is a semi-cycle graph where some of the information flows not only in one direction from input to output, but also in the opposite direction. Information is transmitted both forward and backward. It uses its internal memory to process any sequence of inputs. They can be full-recurrent or semi-recurrent. RNN are used to recognize the sequential characteristics of a data and to use the pattern to predict the next potential scenario [85, 92].



(a) Forward neural network



(b) Recurrent neural network

FIGURE 6.11: Two basic artificial neural network architectures.

Furthermore, from those first two main classes, other types of architectures emerged.

- Deep feed forward (DFF) neural networks: They are also known as long short term memory-(LSTM) network. It is an improvement of RNN which extends their memory. It is configured into blocks of "long short term memory" that are capable of remembering values for any length of time [92].
- Convolutional neural networks (CNN): This type of network is similar to FNN where neurons have learn-able weights and biases. It contains hidden layers that include multiple convolutional layers, pooling layers, fully connected layers, and normalization layers. This kind of networks are useful in computer vision applications that involve image classification and object recognition [93].
- Radial basis function (RBF) neural networks: They are a kind of network that considers the distance of a point with respect to the center. They are used in power restoration systems [93].
- Modular neural networks: A modular neural network has a number of different networks that function independently and perform sub-tasks. They are used in target recognition [92].
- Self organized maps: They are designed to self-organize similar data which have not yet been classified. They use unsupervised learning in the training phase to produce a low-dimensional (typically two-dimensional) discretized representation of the input space of the training samples, called a map. This way, they make a dimension reduction. [92, 94].

Next, the main performance metrics used to evaluate the different machine learning models, including decision trees and neural networks, are presented.

6.4 Performance metrics to evaluate ML algorithms

As previously described in section 6.1, an ML algorithm must be evaluated and compared with other models to assess if it is useful for our implementation. It will basically depend on the trade-off between benefits and complexity. Different performance metrics are used to evaluate ML algorithms, commonly used with supervised learning algorithms. They are listed and explained below.

- **Confusion matrix**
  It is used to evaluate classification problems, where the output can be two or more classes. It is usually structured as a square matrix that consists of columns and rows that list the number of instances as ”actual class” vs. ”predicted class” ratios, see Fig. 6.12.

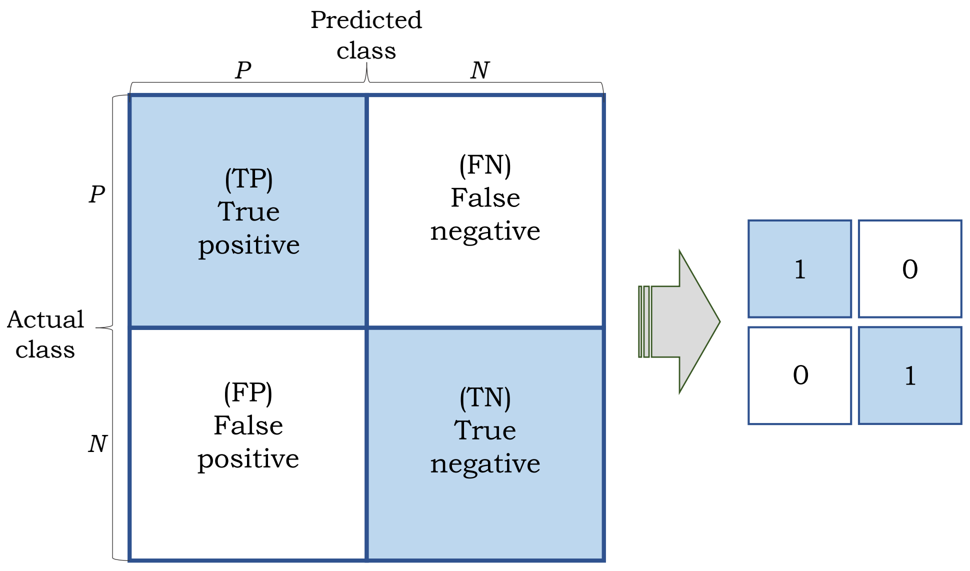


FIGURE 6.12: A confusion matrix example.

Let us define the terms associated with a general confusion matrix. For instance, considering the example of the diabetes dataset described in Fig. 6.1, the output value corresponds to 1 if the person has the disease and 0 if not.

  - *True positive* (**TP**): It represents the cases when the sample output is 1 (true) and it is predicted as also 1 (true). For instance, a patient that has diabetes, and the model classifies his/her case as diabetes.

  - *True negative* (**TN**): The cases when the actual class of the data point was 0 (false) and the predicted is also 0 (false). For instance, a patient that does not have the disease and the model classifies the case as not having the disease.

  - *False positive* (**FP**): Cases when the actual class of the data was 0 (false) and the predicted value is 1 (true). It means the model has predicted incorrectly. For instance, a patient that does not have diabetes and the model classifies his/her case as diabetes.

  - *False negative* (**FN**): Cases when the actual class on the data point was 1 (true) and the predicted value is 0 (false). The model predicts incorrectly with a "false". For instance, a person having diabetes and the model classifying his/her case as no-diabetes.

In this sense, depending on the problem to solve, the goal should be to minimize either FP or FN values. The confusion matrix is not used as a performance measure as such, but most of the performance metrics used in ML are based on its values:

- **Accuracy.** This metric tells us if a model has been trained correctly or not. Also, we can have an idea on how the model will perform in general. However, it works well only if there are equal number of samples belonging to each class. Accuracy may not be a good measure if the dataset is not balanced (i.e., both negative and positive classes have quite different number of data samples). This performance metric is calculated as the sum of correct predictions divided by the total number of predictions. This metric varies in the interval [0, 1]. In the example of the diabetes dataset described in Fig. 6.1, this metric answers the question *"How many patients did we correctly label out of all the patients?"*

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} \tag{6.3}$$

- **Error (ERR).** This metric calculates the sum of all false predictions divided by the total predictions. This metric varies in the interval [0, 1]. In the example of the diabetes dataset described in Fig. 6.1, this metric answers the question *"How many patients did we wrongly label out of all the patients?"*

$$ERR = \frac{FP + FN}{FP + FN + TP + TN} \tag{6.4}$$

- **True positive rate (Sensitivity or Recall).** This metric is useful for those cases when the cost of false negatives is high. This metric corresponds to the portion of positive data that are correctly considered as positive, with respect to all positive data [8]. This metric varies in the interval [0, 1]. In the example of the diabetes dataset described in Fig. 6.1, this metric answers the question *"Of all the people who are diabetic, how many of those we correctly predict?"*

$$Sensitivity = \frac{TP}{FN + TP} \tag{6.5}$$

- **False positive rate (Specificity).** This metric corresponds to the proportion of actual negatives which got predicted as negative, with respect to all negative data. This metric varies in the interval [0, 1]. In the example of the diabetes dataset described in Fig. 6.1, this metric answers the question *"Of all the people who are healthy, how many of those did we correctly predict?"*

$$Specificity = \frac{TN}{FP + TN} \tag{6.6}$$

- **Receiver operator characteristic (ROC).** A common way to measure the performance of a classifier is to use ROC curves. ROC is a probability curve used to select classification models based on the performance of the Sensitivity with respect to the Specificity, see Fig 6.13. The diagonal of ROC graphs means that classification models under this diagonal line are considered as worse as random guessing. The area under the ROC curve (AUC) is also a representative performance value. AUC represents degree or measure of separability. It tells how much the model is able to distinguish between classes. AUC takes values between 1 (perfect test) and 0.5 (useless test).The higher the AUC, the better the model is at predicting 0s as 0s, and 1s as 1s.

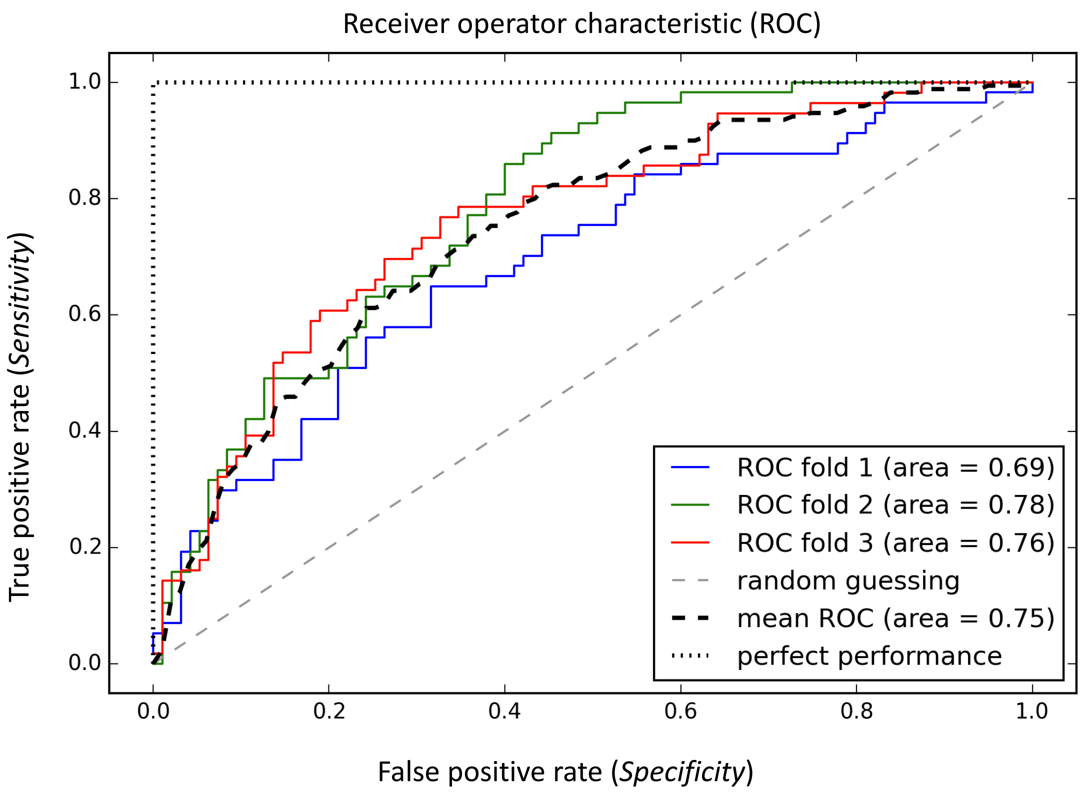


FIGURE 6.13: An example of an ROC curve [8]

.

The ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1 − FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. As reference, a random classifier would give points lying along the diagonal (FPR = TPR). Therefore, the closer the curve comes to the 45-degree diagonal of the ROC curve, the less accurate the test. On the contrary, a perfect classifier would fall into the top-left corner of the graph with a true positive rate of 1 and a false positive rate of 0.

However, the ROC curve does not measure well for imbalanced data. Notice that a dataset is imbalanced if at least one of the classes constitutes only a very small minority. The reason is that since ROC plots TPR vs. FPR and TPR only depends on positives, ROC curves do not measure the effects of negatives. [8].

- **Precision.** This metric helps to assess the performance of ML algorithms in cases when the cost of false positives is high (i.e., a false alarm in the detection of any disease). It corresponds to the number of correct positive results divided by the number of positive results predicted by the classifier. This metric varies in the interval [0, 1]. In the example of the diabetes dataset described in Fig. 6.1, this metric answers the question *"How many of those who we labeled as diabetic are actually diabetic?"*

$$Precision = \frac{TP}{TP + FP} \tag{6.7}$$

- **F1-score.** It is the harmonic mean between precision and recall metrics. This metric balances the use of precision and recall to provide a more realistic measure of a test's performance. F1-score becomes 1 only when precision and recall are both 1, so it is a more precise measure than accuracy. The harmonic mean since penalizes the extreme values [8]. This metric varies in the interval [0, 1].

$$F1 - score = (\frac{Recall^{-1} + Precision^{-1}}{2})^{-1} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{6.8}$$

- **Mean absolute error (MAE)**. This metric measures the average magnitude of the errors in a set of predictions, without considering their direction. It is the average over the test sample of the absolute differences between prediction ($\hat{y}$) and actual observation (y), where all individual differences have equal weight. It is always non-negative, and values closer to zero are better [95]:

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{6.9}$$

- **Mean squared error (MSE)**. The MSE of an estimator measures the average of the squares of the errors, i.e., the average squared difference between the predicted value ($\hat{y}$) and the actual value (y). It is always non-negative, and values closer to zero are better [8].

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6.10}$$

- **Median absolute error (MedAE).** The loss is calculated by taking the median of all absolute differences between the true value (y) and the predicted value ($\hat{y}$). It is always non-negative, and values closer to zero are better [95]:

$$MedAE(y, \hat{y}) = median(|y_1 - \hat{y}_1|, ..., |y_n - \hat{y}_n|) \tag{6.11}$$

- **Mean absolute percentage error (MAPE).** This metric measures the size of the error in percentage terms. It is always non-negative, and values closer to zero are better [96]:

$$MAPE = 100 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{6.12}$$

There are several metrics that can evaluate the performance of a model. Sometimes, the accuracy (measure of all the correctly identified cases) is sufficient. It is most used when all the classes are equally important (with a similar number of instances). Accuracy alone is not a useful measure whenever there is class imbalance. In such cases precision, recall, F1-score, area under

ROC curves can be useful. All of the above performance metrics described are commonly used to measure the performance of prediction models.

6.5 Conclusions

In this chapter, we have presented an overview of machine learning (ML) and artificial neural networks (ANNs). Both ML and ANNs are a subset of artificial intelligence (AI), the main difference is that ANNs are commonly applied to large datasets, and their architecture is more complex. We have provided a general description, taking into account the classical classification of the learning models and their more relevant algorithms.

Generally speaking, the learning process can be done in different ways such as task-driven, data-driven, or learns to react to an environment. In the first part of the chapter, we have described the main characteristics of two basic machine learning models that learn in a supervised way: (i) decision and (ii) bagged trees. However, to design more efficient algorithms with higher accuracy predictions, an advanced form of machine learning is needed, referred to as deep learning or (iii) artificial neural networks. In the second part of the chapter, deep learning, which is based on a layered system, is covered. An artificial neural network is a connection of several neurons that compute the information. These neurons capture the statistical structure and therefore they can create a joint probability distribution over the input variables. The main ANN components and the general configuration parameters were described.

This introductory chapter about ML generally addresses the theory on which our research work included in the next chapter is based. Note that, we are focused on methods and models that permit to predict a specific result based on the previous information learned (experiences on the past). In our context, it is to predict the best candidate node to forward data to destination. In the next chapter, our case study is presented. Also, we introduce our proposals of ML-based routing protocols for VANETs in urban scenarios.

Chapter 7

Multimetric predictive routing protocols for VANETs in urban scenarios

In this chapter, we present our ML-based proposals to improve a multimetric routing algorithm for VANETs in urban scenarios. We have collected data and created a dataset to train our machine learning models aiming to predict the VANET behavior from the point of view of packet delivery probability. An appropriate validation phase is used to evaluate the performance of our ML models. Afterwards, we use the designed ML-based forwarding model to predict in a VANET urban scenario which is the best candidate node to forward a packet towards its destination.

7.1 Results of this chapter

A manuscript with the results of the proposal presented in this chapter is in preparation. The possible title of the manuscript as well as the names of the authors are as follows:

- **Leticia Lemus Cárdenas**, Ahmad Mohamad Mezher, Pablo Barbecho Bautista, Juan Pablo Astudillo León, and Mónica Aguilar Igartua, "Multimetric predictive routing protocols for VANETs in urban scenarios", *In preparation.*

7.2 Introduction

Intelligent transport systems (ITS) have several components whose goal is to improve operation and road safety in urban and rural transport. Nowadays, a lot of information is collected by computational devices (e.g., sensors, cameras) available almost everywhere. This large amount of properly managed collected data along with machine learning techniques, can be used to develop useful tools to improve the aforementioned ITS. As observed in the previous chapter, machine learning is the application of artificial intelligence which allows computers to predict outcomes and behaviors automatically without the intervention of human beings. One of the main pillars of ML techniques is the ability to manipulate a large amount of data.

The scope of ML is very extensive due to the fact that the computational hardware capacity has increased considerably in the last years. Today, many ML-based applications offer a large amount of services, such as financial advice, marketing, sales, health-care, image recognition, social media, and so on. In the field of ITS, ML benefits focus on issues such as transport network, traffic flows, traffic signals, public transportation, driving styles, electric cars, car sharing, among others [97].

In this thesis work, we considered interesting to adopt machine learning models to improve routing in VANETs. The objective is to enhance the selection process of the next-hop candidate node to forward a packet to destination. Also, it represents an efficient way to make forwarding decisions on nodes based on predictions according to an ML algorithm.

Specifically, our contributions in this chapter are the following:

- We have created **datasets** based on the collection of different traffic metrics from VANET simulations in urban scenarios. The five collected metrics values are:
  - Available bandwidth
  - Distance to destination
  - Nodes' density
  - MAC layer losses
  - Nodes' trajectory

  Notice that those metrics are gathered by nodes themselves from their neighbourhood (i.e., in the region within their transmission range). Then, nodes periodically interchange those values in beacon messages with their neighbours. This way, nodes have a local knowledge of the VANET, which fulfils the decentralized operation required in VANETs.

- We have trained different **machine learning models** to assist our forwarding algorithms aiming to get the best accuracy related to an expected outcome. Specifically, we train our models so that the packet delivery probability is the maximum.

- Then, the designed **ML-based forwarding algorithm** with the highest accuracy will be implemented in our routing protocol. Afterwards, their prediction power will be evaluated in a VANET scenario where new data will be used to carry out a performance evaluation of our proposals. Thereby, three different models were selected in the test validation: (i) *decision tree*, (ii) *bagged decision tree*, and (iii) *multi-layer artificial neural network*. (i) Decision trees are suitable for modeling our forwarding decision problem as they are based on successive binary classifications that can be easily adapted to model our problem. Also, they are simple to model, provide a simple visual representation of the data. Furthermore, they are easy to be programmed for computer systems with IF/THEN/ELSE statements and are fast to be executed. (ii) Bagged decision trees are also suitable to model our problem at the cost of a higher computation complexity. Their structure and operation is similar to decision trees. They improve the predictive capability of a single decision tree since they depend on many decision trees instead of depending on a single decision tree, Finally, (iii) ANNs are more complex prediction models build from an assembly of nodes that look somewhat like the human brain. They have shown to perform better than decision trees in our case.

- Finally, we include a **performance evaluation** to assess the benefits and costs of our proposals. The proposed ML-based forwarding algorithms do not incur any additional overhead. We use just the same metrics gathered by the multimetric routing protocol by means of periodic beacon interchange. Consequently, our algorithm generates the same overhead as the basic multimetric routing protocol used to compare with. Results obtained with our novel ML-based routing protocols are better by far compared to our previous multimedia multi-metric map-aware routing protocol (3MRP) [4]. The benefits are better in terms of packet losses at the cost of a slightly higher end-to-end packet delay due to the calculation performed by the ML algorithm.

7.2.1 Motivation

The main motivation that led us to design the proposals presented in this chapter was to study a new way to take forward decisions in VANET nodes. Our contribution in this matter lies in the adaptation of the ML models designed taking advantage of the specific VANET characteristics.

To attain our goal, we have designed and tested different ML models aiming at improving the network performance in terms of average packet losses and average end-to-end packet delay. We focus our research work in urban scenarios where vehicles forward warning/reporting messages to an RSU following hop-by-hop V2V and V2I communications.

To the best of our knowledge, there is no equivalent work in the literature on ML-based routing protocols for vehicular networks. That is, no work has yet been published on improving multimetric routing protocols for VANETs where forwarding decisions are based on prediction using machine learning methods.

7.2.2 Related works

In this section, we present the state of the art on machine learning solutions proposed for VANETs to improve specific aspect. The wide search for related proposals has been focused on looking for works that fulfill two requirements: (i) works that implement multimetric strategies to select a next-hop node to forward information, and (ii) proposals whose strategies are supported with ML model. However, the result of our search was not successful.

It seems clear from our search that the topic in vehicular networks most likely to benefit from ML-based proposals is autonomous vehicles (AVs). Certainly, there are many recent proposals of ML models to assist driving in AVs. For instance, we highlight [98] that proposes a traffic control system based on random-forest (RF) predictions to determine AVs' routes to reduce congestion rates. Their proposal is able to reduce the burden on the human experts who monitor and control AVs. In [99], the authors design and analyze deep learning architectures based on convolutional neural networks (CNN) to enhance semantic image segmentation, which is used in autonomous vehicles for self-driving. They propose fully convolutional neural (FCN) models which are trained to obtain the maximum accuracy and lowest training time, keeping precise their FCN model to segment objects.

Authors in [100] propose a ML model to predict vehicles' mobility. More concretely, they propose a centralized routing scheme with mobility predictions in VANETs assisted by a software-defined network (SDN) controller using artificial neuronal networks techniques. The idea is to use the

SDN controller to estimate and predict the vehicles' arrival rate measured at road side units (RSU) or at base stations (BS). Their proposal shows benefits in terms of successful transmission probability and average packet delay in both V2V and V2I communications.

Another example is presented in [101], where the authors propose data delivery virtualization using reinforcement learning (RL). They evaluate each one of the one-hop link status considering the vehicles' speed, vehicles' density in the same direction, and average channel condition. Those metrics are used to select cluster head (CH) nodes that will act as sender nodes. Then, they design a game-theoretical multi-hop routing protocol that includes their RL model.

Authors in [102] present a proposal on link prediction by implementing supervised machine learning techniques. A support vector regression is implemented to predict the vehicles' trajectory considering four mobility cases of the vehicle's speeds: (i) high and constant speed, (ii) turn case speed, (ii) parking or intersection speed, and (ii) when the vehicle starts to move. Their proposal predicts link failures using machine learning and the vehicles' trajectory knowledge.

In [103] we can see a proposal named reliable self-adaptive routing algorithm (RSAR) to design a Q-learning based routing protocol. The basic idea is to use Q-learning algorithm to assess the path from each source's neighbor to destination using continuous interaction with the external environment. With their Q-learning based algorithm they register a table with scores (Q) of the node's neighbors, computed from measures of link-availability, link life-time, and distance to destination. Then, the neighbor with the highest Q-value is selected to forward the packet towards destination. Similarly, we find another proposal presented in [104] where the authors introduce a novel routing protocol for urban VANETs called RSU-assisted Q-learning-based Traffic-Aware Routing (QTAR). QTAR learns the road segment traffic information based on the Q-learning algorithm. In QTAR, a routing path consists of multiple dynamically selected high reliability connection road segments that enable packets to reach their destination effectively. Simulation results show higher average packet delivery ratios and lower average end-to-end delays with respect to other traffic-aware routing proposals.

The previous proposals attack several forms to enhance vehicular communications. Some of them require extra infrastructure or additional signaling overhead to assist the monitoring or to transmit messages. We pose that a vehicle could take a smarter decision to forward a packet if it knew in advance the probability to successfully delivery the packet regarding each candidate node to be chosen as the next forwarding hop. In the next sections, we describe our proposal to choose the best forwarding node in VANETs based on an ML-based prediction algorithm.

7.3 Dataset collected from multimetric measures in vehicular urban scenarios

In any ML-based project, the first step prior to develop any model is to collect data and organize it according to the ML model requirements. Furthermore, building a tagged dataset is not easy in practice, as it strongly depends on the type of data to be observed. Most of the available datasets regarding vehicular networks are those related to mobility patterns. Unfortunately, there are not available datasets that deal with data packet flows in VANETs. In consequence, it was firstly necessary to gather V2V and V2I information from a realistic urban scenario using our OMNeT++/Veins/SUMO/OSM simulator. Specifically, our goal was to obtain the probability of successful packet delivery under specific characteristics of the environment.

To achieve our goal, we have used the same scenario previously presented in Section 5.5 to carry out a large set of VANET simulations over an area of Barcelona city map (see Fig. 7.1). From each simulation we have collected the values of five metrics, which are detailed in sec 3.2.2.1. The general objective is that VANET nodes can take the best forwarding decision taking those five metrics into account in the routing protocol. The five metrics considered are:

- **Available bandwidth**. Estimated in the link formed by the node currently holding the packet and each candidate node in the neighbourhood to be the next forwarding hop.
- **Vehicle's density**. It is the number of neighbours within the transmission range.
- **Distance to destination**. Euclidean distance computed from each neighbour to the packet's destination.
- **Vehicle's trajectory**. It informs about the moving direction and speed towards destination, computed for each candidate node in the neighbourhood to be the next forwarding hop.
- **Average MAC losses**. Percentage of packet losses computed in the MAC layer.

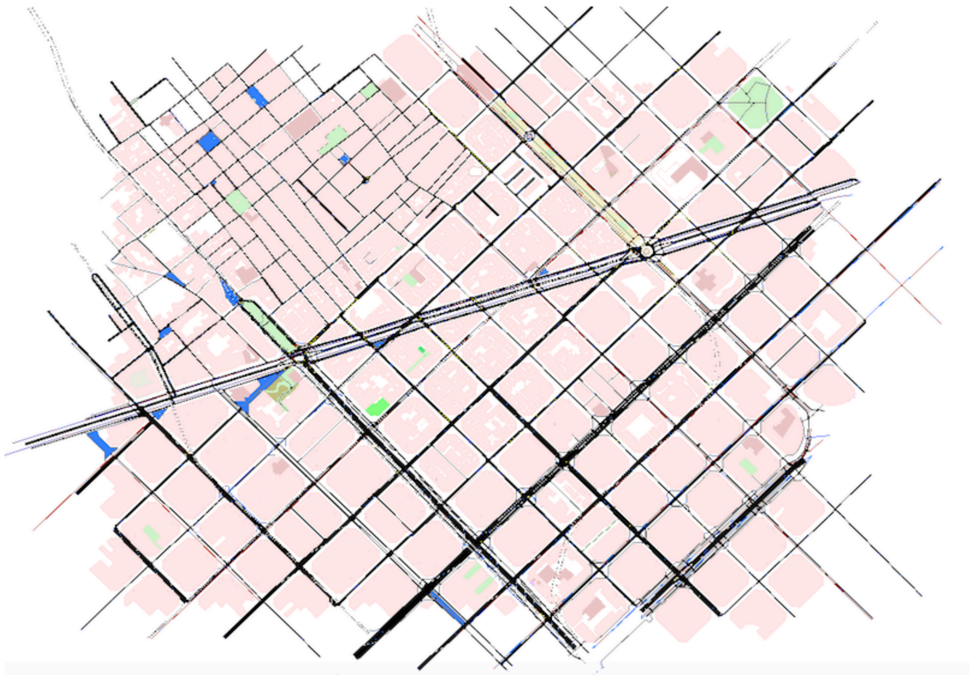


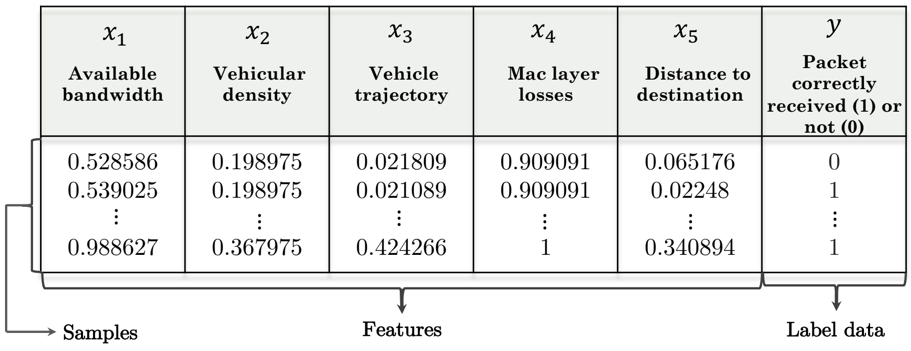
FIGURE 7.1: Urban area in Barcelona city. Map extracted with SUMO [9]. Simulation area of 2300m x 2100m.

To collect these metrics from VANET nodes, we rely on periodic exchange of beacon messages between neighbour nodes. In the considered scenario, reporting messages (e.g., regarding traffic statistics or a traffic accident) are forwarded hop-by-hop from a source vehicle to a destination (RSU) (see Fig. 5.10). To build a representative dataset, it is necessary to get a wide range of results (i.e., the probability of packet successfully delivered) for all possible values in the five considered metrics (see the list above). This way, we relate output values (probability of packet successfully delivered) for each vector of input values (five considered metrics). Therefore, we have considered different possible configuration settings in our VANET scenarios. Among these configuration features we highlight the vehicles' density ranging from low to high values (20 to 300 $vehicles/Km^2$), the average vehicles' speed in urban roads (20 to 100 km/h) and different positions of the source nodes in the map area. This way, we will have a complete representative dataset that covers the maximum number of possible values in the five considered metrics (distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses). Recall that those five metrics are the inputs of our ML-based predictive algorithms to assist routing protocols in VANETs (see Fig. 7.2).

The data collection process requires the following steps:

- Extract the values of the metrics in one hop. Obtain the corresponding output value, i.e. the percentage of packets delivered for that configuration.
- The dataset can be organized in five columns with the values for the five metrics ($x_1, ...x_5$) as inputs, and another column with the output values $Y(0, 1)$,
- Our dataset consists of 6000 records, where each column corresponds to an output value (Y). This output value means whether the message was received (1) or not (0) at destination.
- Once we have enough information in our dataset, the machine learning models can be trained and tested. The criteria used to determine whether enough data has been collected or not, is basically the values of accuracy (training and testing) obtained by the ML-based forwarding-decision models. We have created a large number of scenarios with different configuration settings to fully cover the range of values for each routing metric (i.e., distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses). These values will be the inputs of the ML models, while the output will be the prediction of the packet successfully delivered or not at destination.

We have collected normalized values for each metric. This is necessary so that the algorithms can operate with them under the same range of values each, so that algorithms can mix them, compare them, and use them in operations. The resulting dataset has the format shown in Figure. 7.2).



x_1 Available bandwidth	x_2 Vehicular density	x_3 Vehicle trajectory	x_4 Mac layer losses	x_5 Distance to destination	y Packet correctly received (1) or not (0)
0.528586	0.198975	0.021809	0.909091	0.065176	0
0.539025	0.198975	0.021089	0.909091	0.02248	1
⋮	⋮	⋮	⋮	⋮	⋮
0.988627	0.367975	0.424266	1	0.340894	1

Samples
Features
Label data

Figure 7.2: Appearance of our dataset. Features and labels for the training and testing phases in our ML models.

Note that our target problem is clearly a binary classification problem because the expected outcome has two possible values (0 and 1). Therefore, decision tree and bagged tree models are suitable options to tackle our goal. ANN also will be used to model our ML-based forwarding algorithms, since they can handle binary data even better than decision trees. Of course, the best fitted model will be the one that most accurately fits our data. Then, the next step is to train classification algorithms and evaluate them through the typical machine learning performance metrics summarized in chapter 6. Finally, those algorithms are implemented in our proposal of VANET routing protocol and evaluated with our simulation framework OMNeT++/Veins/SUMO/OSM.

Table 7.1: Main characteristics and total time spent to build our dataset, including data collection time and human processing time.

Main features of the dataset collection process	**Description**
- Data size	713.2 KB
- Number of records	6000 (see Fig. 7.2)
- City map	Representative area of Barcelona city (see Fig. 7.1)
- Hardware (personal computer)	Processor Intel® i7 Base clock frequency 3.4 GHz x 4 RAM 32 GiB OS type 64 bits
- Total running time in the simulation process	3672 hours
- Human processing time	Around 850 hours in total, during 7 months

To visualize the cost in time and human effort that has been required to generate the dataset, Tab. 7.1 presents a summary with the most important characteristics that describe the process of collecting the dataset to build our dataset. In the next sections, the training, validation testing and implementation phases of our ML-based forwarding algorithms are presented.

7.4 Design of ML prediction models to enhance VANET routing protocols: Decision trees and ANN algorithms

Nowadays, researchers have access to large amounts of data that can be used to design useful services and applications to be reverted to the users who generated that data. Big data is a combination of diverse kind of data collected by organizations that can be mined for information and used in machine learning projects, predictive modeling and other advanced analytics applications [105]. Besides, current available computation power makes it possible to perform a large amount of tests and experiments to come up with efficient machine learning and neural network models in a short time. In this section, we will describe the process for training the three learning models (i.e., decision tree, bagged tree and artificial neural network) that we have chosen in this thesis to design our ML-based algorithms to enhance multimetric routing protocols for VANETs. We have performed different training runs over different models in order to choose the best classifier for our dataset. Our dataset has been collected from a large amount of representative simulations for VANETs in realistic urban scenarios. The best classifiers have been chosen in terms of accuracy, prediction power, ease implementation and complexity. Our goal is twofold:

- On the one hand, our objective is to find a ML-based forwarding-decision models that learns and predicts better using our dataset, see section 7.3. To attain this objective we have used Matlab [106], since this framework already has a full library with ML algorithms and it is easy to use. We have used Matlab to design our ML-based algorithms, to test them and validate them.
- On the other hand, we need a ML-based forwarding-decision models that can enhance multimetric routing decisions in urban VANETs. Once we have done the design and analysis of our algorithms, we have implemented them in our OMNeT++/Veins/SUMO/OSM simulator to asses a performance evaluation of our proposals in realistic urban scenarios.

7.4.1 Prediction model based on decision tree

Decision trees are prediction algorithms that are suitable to solve classification problems. As previously described in Sec. 6.2.2, decision trees have different parameters that can be adjusted and, depending on how they are configured, can lead to different performances. Among them, the dimension or depth of the tree is one of the most important configuration parameters. It should be noted that the higher the tree depth, the better the prediction power. However, this implies a significant increase in the number of $if - then - else$ operations. Consequently, depending on the selected number of operations, the structure of the tree becomes more or less complicated to be implemented in a simulator. In our design we have analysed the trade-off between accuracy and complexity of the tree, so that we obtain the minimum dimension in the ML models than satisfies a given accuracy and ROC curve.

We have used the MATLAB [106] classifier learners set to fulfil the preprocessing and learning phases. Besides, we have done the validation of the dataset. To this end, we have implemented the cross-validation method to avoid overfitting. **Cross-validation** is a well-known technique that protects against overfitting by partitioning the dataset into folds and estimating the accuracy of each fold, see Fig. 6.4. This strategy is highly recommended before training any ML-based model, see sec. 6.1. After that, we have selected and trained the learning algorithms to obtain the final models. Finally, we have chosen the best ML models with the highest accuracy value.

Figures 7.3 and 7.4 show the ROC curves of the two trained decision tree classifier models. As we explained in section 6.4, a perfect classifier would fall into the top-left corner of the graph with a true positive rate of 1 and a false positive rate of 0. Contrarily, the closer the ROC curve is to the 45-degree diagonal, the less accurate the test will be. Besides, the closer to 1 is the area under the curve (AUC), the better is the classifier.

Several decision tree models have been tested varying their configuration parameters. We show in figures 7.3 and 7.4 two representative cases for different values of the tree depth. According to the the ROC curve and the AUC value, it can be seen that the bagged decision tree with a tree depth of 20 performs better than the simple decision tree with a tree depth of 6. Anyway, both classifiers show a high percentage of accurate predictions.

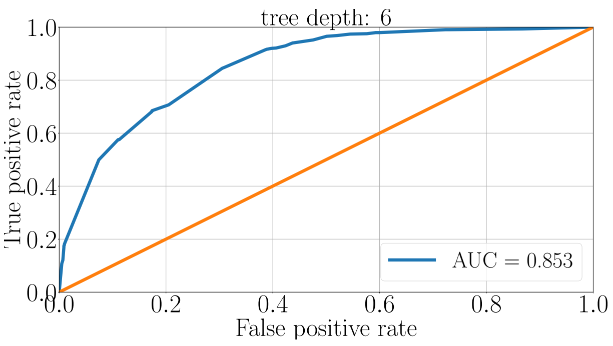


FIGURE 7.3: ROC of our decision tree model with tree depth = 6.

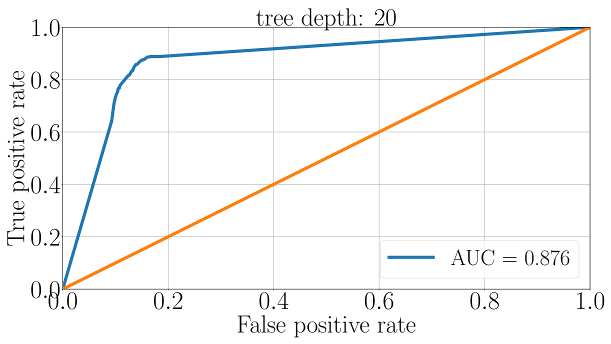


FIGURE 7.4: ROC of our bagged decision tree model with tree depth = 20.

To analyse if our model presents overfitting, we have obtained the accuracy curve of our decision tree models trained and tested with our dataset, see Fig. 7.2. For instance, in Fig. 7.5 we show the accuracy curve of our bagged decision tree, as a function of the tree depth, which varies from 1 to 50. In general, a lower tree depth will make the model faster but not as accurate, whereas a higher tree depth increases accuracy but risks overfitting and may be slow. In the figure we can see that the accuracy gets the maximum value from 20 and over. Furthermore, we can see in Fig. 7.5 that the model is clearly overfitted. Note that if a model shows high precision during the training phase but does not show similar precision during the testing phase, this indicates overfitting. The **overfitting** problem occurs if during the training phase, the model learns the complexity of the training data in such detail that it creates a complex function that can almost map all input data to output data correctly, with very little error. In the case of overfitting, the model shows high precision during the training phase but falls with low precision with the test or data not seen.

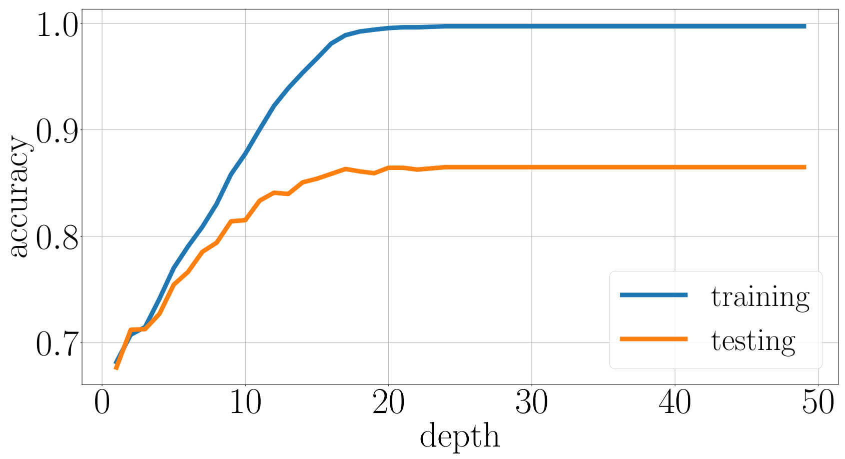


FIGURE 7.5: Accuracy curve for our bagged decision tree, as a function of the tree depth. Notice the overfitting.

We have selected two ML-based algorithms to model our framework: (i) a simple **decision tree** with a tree depth of 6 (which we name as ML_6), and (ii) a more complex **bagged tree** with a tree depth of 20 (which we name as ML_{20}). According to Fig. 7.5 we have chosen a tree depth of 20 for the bagged tree, since from that value above the maximum accuracy is already achieved in the training phase. We have used them to assist our multimetric routing protocol for VANETs in urban scenarios. Recall that our goal is that the node currently holding the packet can efficiently choose the better next hop to forward the packet to destination. That best next hop will be the neighbour which provides the higher chance to successfully deliver the packet at destination, according to our ML-based prediction model.

Table 7.2 presents the accuracy results of the training and learning phases corresponding to our two ML-based algorithms. In this table, the characteristics of the configuration used to train the ML algorithms are described.

TABLE 7.2: Training parameters and characteristics of the decision tree models used.

Training phase characteristics	**ML decision tree algorithm**
- Data size: 6000 samples - Predictors (5): available bandwidth, distance to destination, nodes' density, MAC layer losses, nodes' trajectory. - Response classes (2): packet delivered or not = [1, 0]	- Model 1: Simple decision tree, tree depth= 6 (ML_6) - Split criterion: Gini's diversity index (see sec. 6.2.2) - Accuracy: 76.2%
- Validation: 20-fold cross-validation (see Fig. 6.4)	- Model 2: Bagged decision tree, tree depth = 20 (ML_{20}) - Ensemble method: Bagged tree - Accuracy: 89%

As a starting point, we have studied and tested classical ML-based prediction models, such as decision trees. The best ML algorithm that we have found for our dataset (see sec. 7.3) is a bagged decision tree of tree depth = 20. However, this algorithm shows overfitting (see Fig. 7.5) and its accuracy can still be improved (see table 7.2). Therefore, the next step will be to find a model that learns better than these two previous classifiers (see Appendix A.1). We have investigated and designed a more complex and power kind of learning algorithms: artificial neural networks (ANN). In Appendix A we show the three ML-based models (decision tree, bagged decision tree and ANN) designed and the correspondent Phyton code that we have generated.

In the next subsection, our neural network design is explained. To do so, we have used the same training dataset (see sec. 7.3) than with our ML-based algorithms.

7.4.2 Prediction model based on a multi-layer feed-forward neural network

ML defines a set of algorithms that parse data and learn from the dataset to build a model with which later be able to predict results. ANN is a subset of machine learning algorithms classified as deep learning models, which are designed to analyze data with the logic structure like how we humans would draw conclusions.

Artificial neural network techniques belong to the set of supervised learning algorithms and are implemented to solve, among others, classification problems. In this sense, we have used our collected dataset to generate and obtain an artificial neural network model. The neural network model was selected based on the highest accuracy value obtained for the training and testing dataset. Next, we have followed the steps of an MLP construction, which was previously summarized in section 6.3.1.

7.4.2.1 ANN architecture design

The model selected corresponds to a feed-forward neural network type with uncycle nodes and hidden layers that are fully connected, see section 6.3.1.2 for more details. In this way, each neuron receives input from all neurons from the previous layer. As mentioned before, an ANN is composed of different layers, and how they are configured is related to the features and labels of the training dataset. First, the set of features (x_i) are our independent variables that form the first layer (input layer). Second, the output dependent variable (Y) corresponds to the output layer. Finally, the hidden layers are not related to the features and labels of the training dataset. Therefore, the values of the hyperparameters (i.e., number of hidden layers and number of neurons) will be adjusted. The goal here is to tune those parameters to achieve a high metric value, e.g. the accuracy. We have tested several neural network configurations before obtaining the final design of our ANN model.

On the other hand, the activation and error function must also be selected. As previously shown in Table 6.1, there are different activation functions. Nonlinear functions are commonly used and widely applied, such as Sigmoid [84] and ReLU [87] functions. In particular, ReLUs are simpler and faster to process. Note that, in a neural network the hidden layers may require a high number of neurons, where each one must execute an activation function. Therefore, a faster activation function is convenient in the neurons that compose the input and hidden layers. In our ANN we have chosen the ReLU activation function in the hidden layers, whereas the Sigmoid is selected as the activation function for the output layer.

The learning process of the ANN designed is described in the following:

- **Back propagation**. It corresponds to random output prediction, comparing those predictions with the true output. Thus, weights and bias are updated until the predicted output comes closer to the true output. This updating process is executed based on the calculation of error or cost function. For our case, we have used the mean square error (MSE), see eq. (6.10) [8]. This error function finds the difference between the true value and the propagated values. The objective then is to minimize the MSE. Hence, the smaller the cost, the more accurate the predictions are. Besides, the neural network loss is minimized. That is, every time the error is calculated, it is required that weight values in the intermediate layers be recalculated, as many times as necessary until the error cost be the minimum, tending to zero if possible. To do so, it is necessary to use an adjusting function to update those weights values of the intermediate layers. We have included in the learning process the *Adam optimizer* to update network weights iterative based in training data. The Adam optimizer [107] is an adaptive learning rate optimization algorithm that was designed specifically to train deep neural networks. Those values are updated at each epoch of the training stage. This algorithm is commonly used for its fast efficiency in deep learning compared to the classical stochastic gradient descent algorithms [108].

- **Hyperparameter selection**. As already mentioned, our neural network topology is based on feed-forwarding learning, where the layers are fully-connected. Remember that we have defined in advance the characteristics of the input and output layers. These layers are based on the features and classes of the collected dataset, see sections 6.3.1.2 and 7.3. Besides, two important parameters that define the network dimension still have to be configured: (i) the number of hidden neurons and (ii) the number of hidden layers. We have performed different ANN configurations varying the number of hidden neurons and layers to find the best set of values in terms of validation metrics and complexity. In this sense, the validation accuracy and validation loss metrics were calculated for each configuration of number of hidden neurons and layers.

We have computed the validation metrics of the combination of (1, 2, 3, 5, 10) hidden layers with (5, 10, 32, 64, 128, 256) neurons at each hidden layer. Fig. 7.6 shows the accuracy of the three best results with the correspondent hyperparameter configurations. As it can be seen, the three hyperparameter combinations show similar values of accuracy. However, the most simple model (3 hidden layers, 128 neurons per hidden layer) has the highest accuracy. Note that a more simple neural model means a lower computational cost in terms of number of operations and activation functions in each layer. Therefore, we have selected the configuration model with a number of hidden layers set to 3 with 128 neurons at each hidden layer.

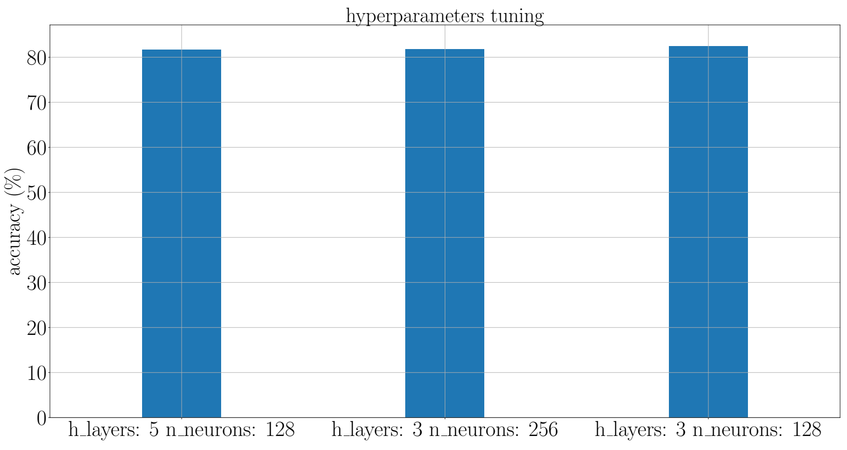


FIGURE 7.6: Accuracy obtained in our ANN model for different hyperparameter configurations: number of hidden layers and number of neurons.

7.4.2.2 Training network

The training process in a neural network is the same as for any machine learning algorithm. That is, we have to provide a set of features (inputs) and one or more labeled classes (outputs).

In our context, we will use our organized dataset previously used for the decision trees (see Fig. 7.2). However, an important point to take into account is how long (number of epochs) it is necessary to train the ANN. The goal is to train the network long enough to be able to learn the mapping from inputs to outputs but at the same time we should avoid having overfitting of training data. Nevertheless, overfitting can occur when the model fits the data in the training set well while incurring a major generalization error in the testing phase, see Fig. 7.5. Therefore, it is important to detect at what point in the training the model falls into overfitting.

A good practice is to validate the error calculation during the training process by using some validation technique. To avoid overfitting, we have used the **early-stopping validation technique** [109, 110]. The goal is to detect when overfitting starts during supervised training of a neural network; training is then stopped before convergence to avoid the overfitting (early stopping). Usually, the original training dataset is divided into a new training dataset and a validation set. Also, the number of training cycles (epoch) that the model will repeat through all training samples, must be set-up.

Fig. 7.7 shows the evolution of the loss function (MSE) throughout the epochs. Here we can see the training loss (blue line) and the validation loss (orange line) around each training cycle (epoch). In training, the value of the error may decrease as the number of repetitions increases (see blue line). However, its real performance to unseen data (see orange line) may no longer change. As it can be seen, the validation loss converges after 200 epochs. It indicates that the model can no longer accurately predict beyond that number of repetitions. Therefore, we are overfitting the model for epoch>200.

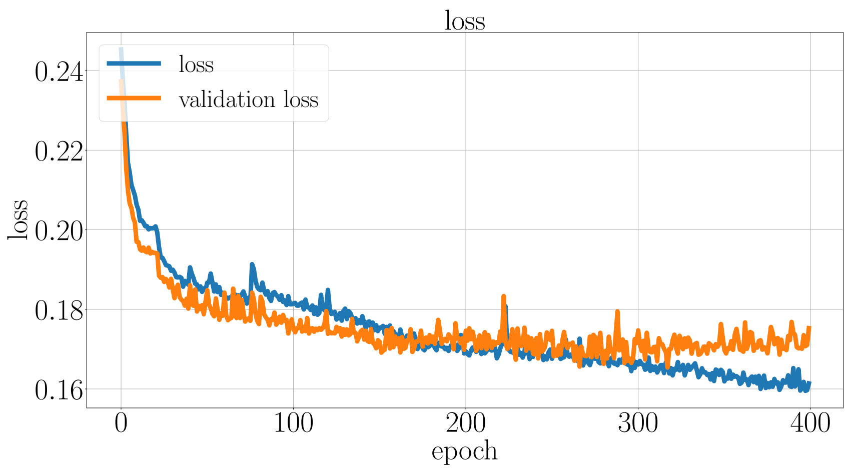


FIGURE 7.7: An example of overfitting detection during the training phase.

By using the early-stop technique, the training will stop when the validation loss does not decrease anymore. Besides, the more epochs are run, the more the model will improve. With both the validation set and the number of epochs, the early stopping technique will avoid overfitting the model. Thereby, we obtain a sufficient number of epochs that our model needs to reach the maximum possible level of convergence. The building, training, and validation of our ANN model have been implemented using the **Keras** [111] tool. With this tool, we have obtained the final neural network model, ready to be implemented in our OMNeT++/Veins/SUMO/OSM network simulator.

The configuration parameters used in the training phase are described in Table. 7.3. In this table we can see that we have used a dataset split ratio of 80% for training, 10% for validation and 10% for testing. See section 6.1 for further information.

Table 7.3: Set of configurations parameters for the neural network training.

	Parameter	**Value**
Model architecture	Neural network type	Feedforward
	Layer type	Dense
	Input and intermediate activation function	ReLU
	Output activation function	Sigmoid
	Number of neurons	128
	No. of input layers	1
	No. of hidden layers	3
	No. of output layers	1
Learning process	Loss function	Mean square error (MSE) Eq. 6.10
	Optimizer	Adaptive moment estimation (Adam) [108]
	Evaluation metric	Accuracy
Training model	Training split	80% of the dataset
	Validation split	10% of the dataset
	Testing split	10% of the dataset
	Number of epochs	200
	Validation method	Early stopping

7.4.2.3 Validation network

To validate the final design of our ANN model, we have obtained the ROC curve and the area under the ROC curve (AUC), shown in Fig. 7.8. As it was explained in section 6.4, the AUC is a parameter that tells how much the model is able to distinguish between classes, i.e. two classes in our case ("1" means packet successfully delivered and "0" means not). The AUC evaluates the goodness of a test. It takes values between 0.5 (useless test) and 1 (perfect test). Thereby, the test with the largest area under the curve is preferable. In our case, the AUC has a value of 0.887, which means that there is an 88'7% probability that the diagnostic made to predict a true positive value is more correct than that of a randomly chosen false positive. Note that, the ROC represented in Fig. 7.8 corresponds to the ANN configuration that obtained the highest accuracy value in the training phase, see section 7.4.2.1.

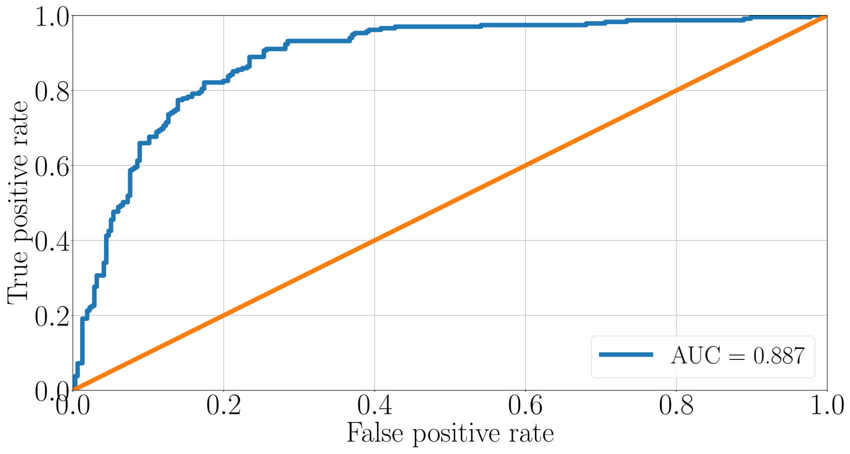
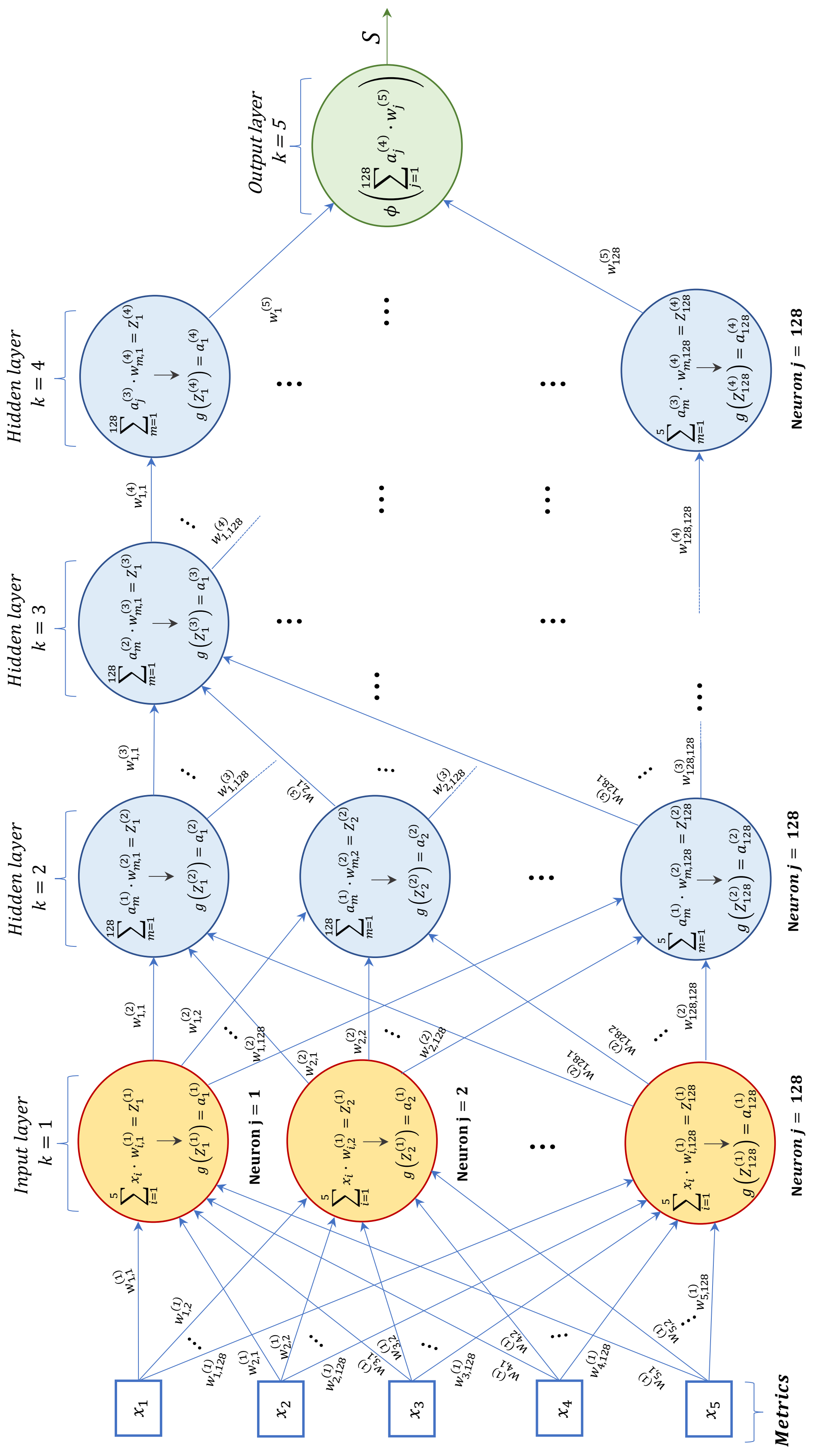


FIGURE 7.8: ROC curve and area under the ROC curve (AUC) of our ANN model once trained.

Finally, we have obtained our ANN model in relation to our collected dataset. The ANN has 1 input layer, 3 hidden layers, 1 output layer (i.e., ANN with 1/3/1). and 128 neurons at each hidden layer. Fig. 7.9 shows the scheme of the final model architecture that we ave designed following all the previous steps. Recall that our ML-based forwarding-decision models is going to be used to assist the node in the vehicular network currently holding the packet to take the better forwarding decision. That is, which is the best next hop to forward the packet and successfully delivered at destination.



Metrics
x_1
x_2
x_3
x_4
x_5
Input layer $k = 1$
Hidden layer $k = 2$
Hidden layer $k = 3$
Hidden layer $k = 4$
Output layer $k = 5$
Neuron j = 1
Neuron j = 2
Neuron j = 128
$\sum_{i=1}^{5} x_i \cdot w_{i,1}^{(1)} = Z_1^{(1)} \rightarrow g(Z_1^{(1)}) = a_1^{(1)}$
$\sum_{m=1}^{128} a_m^{(1)} \cdot w_{m,1}^{(2)} = Z_1^{(2)} \rightarrow g(Z_1^{(2)}) = a_1^{(2)}$
$\sum_{m=1}^{128} a_m^{(2)} \cdot w_{m,1}^{(3)} = Z_1^{(3)} \rightarrow g(Z_1^{(3)}) = a_1^{(3)}$
$\sum_{m=1}^{128} a_j^{(3)} \cdot w_{m,1}^{(4)} = Z_1^{(4)} \rightarrow g(Z_1^{(4)}) = a_1^{(4)}$
$\phi\left(\sum_{j=1}^{128} a_j^{(4)} \cdot w_j^{(5)}\right)$
S

Figure 7.9: Scheme of our designed ANN model. The model will be included in the forwarding decisions of the routing protocol for VANETs.

In Fig. 7.9 we show the scheme of our ANN model designed to assist routing protocols in VANETs. Our proposal will help the current node (i.e., a vehicle) holding the packet to take the best forwarding decision so that it chooses the neighbour that gives the highest chance to successfully deliver the packet to destination. In that figure, we can see the following information:

- S: The output of our ANN. We have 2 classes: "1" means packet successfully delivered at destination and "0" means it is not.
- i: It represents the input number, $1 \leq i \leq 5$. In our case we have five features: distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses.
- j: It represents the neuron number, $1 \leq j \leq 128$.
- k: It represents the layer number, $1 \leq k \leq 5$ (1 input layer, 3 hidden layers and 1 output layer).
- x_i: They represent the inputs to the first layer (input layer): available bandwidth, distance to destination, nodes' density, MAC layer losses and nodes' trajectory.
- $w_{i,j}^{(k)}$: Weights used in the ANN model. They represent the connections between layers and show the strength of a particular node. $w_{i,j}^{(k)}$ is the weight correspondent to input i, neuron j, layer k.
- $\sum_{i=1}^{5} x_i \cdot w_{i,j}^{(1)} = Z_j^{(1)}, 1 \leq j \leq 128$: It is the sum of weighted inputs at the input layer.
- $\sum_{m=1}^{128} a_m^{(k-1)} \cdot w_{m,j}^{(k)} = Z_j^{(k)}, 1 \leq j \leq 128, 2 \leq k \leq 4$: It is the sum of weighted inputs at the hidden layers.
- $g(z_i^k) = a_j^{(k)}$: They represent the activation functions at the hidden layers. We have used the ReLu function, see section 6.3.1.
- $\phi(\sum_{j=1}^{128} a_j^{(4)} \cdot w_j^{(5)})$, $1 \leq j \leq 128, 1 \leq k \leq 4$: It represents the activation function at the output layer. We have used the Sigmoid function, see section 6.3.1.

So far we have finished with the design of our ANN model. The next step is to include our ANN model in the routing protocol to assist vehicles to take the best forwarding decision.

7.5 Scenario description for performance evaluation

We have configured the simulation scenario in our OMNeT++/Veins/SUMO/OSM simulation platform, to carry out a performance evaluation of the proposals presented in this chapter. We assume that network nodes in VANETs know their position and have road map information. We also assume that nodes periodically interchange geographic coordinates, nodes' density, and speed with each neighbour node within their transmission range, using hello messages sent once per second. When a node receives a hello message, it updates its neighbours' table. The node calculates and normalizes a set of five metric values regarding each neighbour: available bandwidth, distance to destination, nodes' density, MAC layer losses and nodes' trajectory.

The node currently holding the packet computes the five metrics for each neighbour. Afterwards, the node uses our tree ML-based algorithms to predict the output (i.e., the packet will be delivered or not at destination) for each neighbor. With this information, the node will arrange its neighbours' list and will take the best forwarding decision, as we will explain in this section. The three ML-based forwarding-decision models are: (i) **decision trees (ML_6)**, (ii) **bagged decision tree (ML_{20})**, and (iii) **artificial neural network**. Once we have implemented the three ML-based forwarding-decision models in the routing protocol, we have evaluated them in two simulation tests. All the city maps have been generated from the OpenStreetMaps (OSM) [5].

Test 1. The first test corresponds to the performance evaluation of the designed ML-based forwarding-decision models in the same scenario from which the dataset was collected (see Fig. 7.1).

   a. An urban area of Barcelona city. It is the scenario used to generate the dataset (see Fig. 7.1), but with a relatively usual oreography for a urban scenario, i.e. it has wide, narrow, regular and irregular streets. Simulation area is of $2300 \times 2100\ m$.

Test 2. The second test corresponds to evaluating how meaningful is the dataset. For this purpose, we will use the ML-based forwarding-decision model with which the routing protocol offered the best results. We are interested in knowing how generalized our dataset is, thus, we need to test the correct prediction capacity of our model in new scenarios. In other words, we will evaluate how flexible is our best ML-based forwarding-decision model and assess if it is able to perform well in other scenarios. For that, we have configured two new city maps in our simulation framework:

   a. An urban area of Berlin city. It is similar in size as the scenario used to generate the dataset (see Fig. 7.1), but with a more complex oreography of the scenario (see Fig. 7.10). Simulation area is of $2500 \times 2500\ m$.

   b. An urban area of Rome city. The area is larger compared to the Barcelona scenario, with a very complex oreography of the scenario, with a lot of irregular streets (see Fig. 7.11). Simulation area is of $3500 \times 2400\ m$.

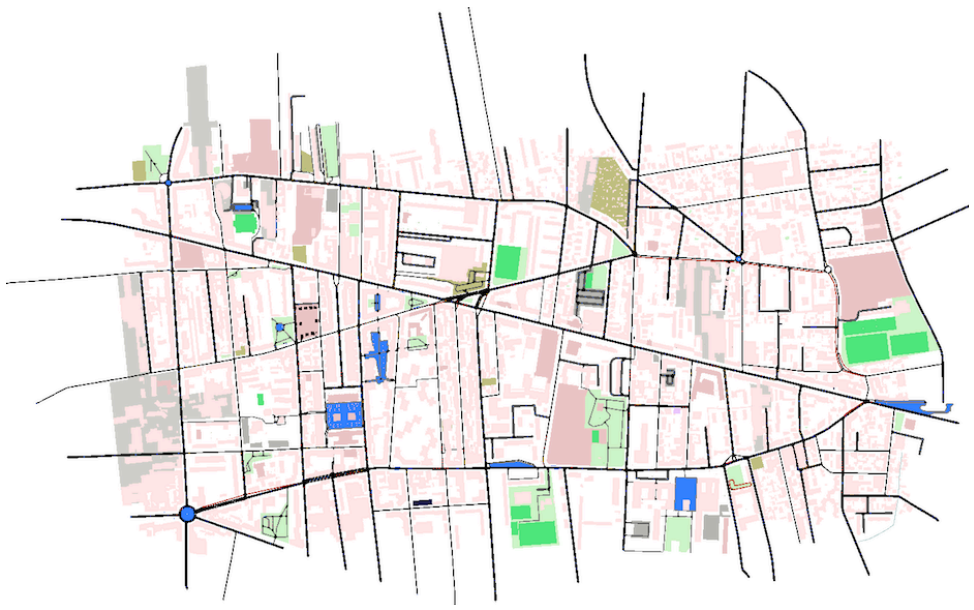


FIGURE 7.10: Urban area of Berlin city. Map extracted with SUMO [9]. Simulation area is of 2500×2500 m.

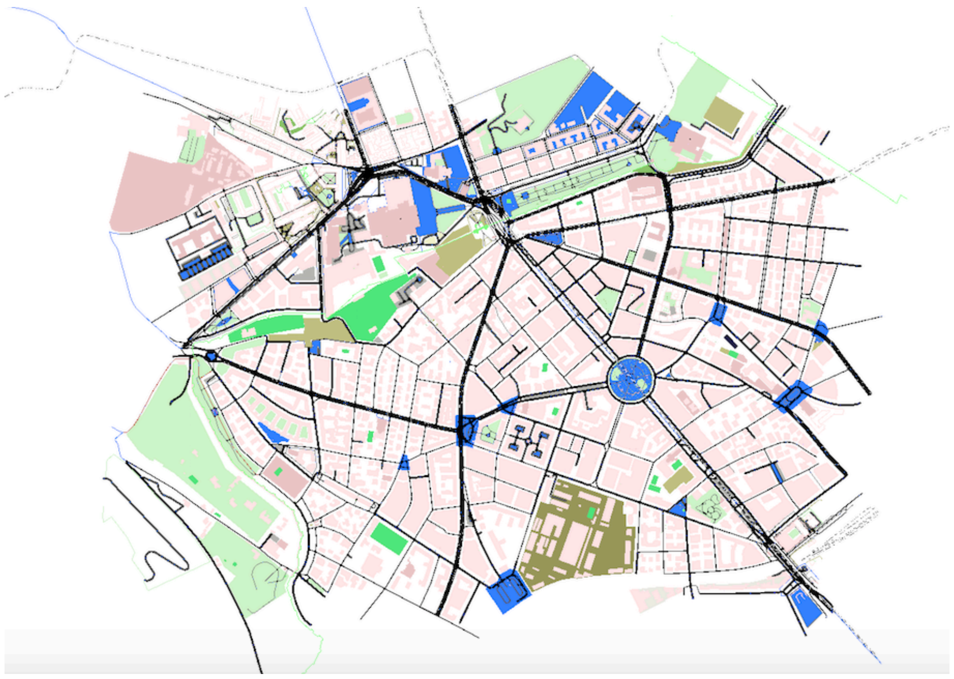


FIGURE 7.11: Urban area of Rome city. Map extracted with SUMO [9]. Simulation area is of 3500×2400 m.

To carry out all the simulation tests, we have configured the simulation parameters with the values presented in Table 7.4. On the other hand, Table 7.5 summarizes the main features of the three city maps (i.e., Barcelona, Berlin and Rome) and the different routing protocols that we have used in each simulation phase: multimetric predictive ML-based routing protocols (MPML), and multimetric predictive ANN-based routing protocol (MPANN).

TABLE 7.4: Simulation parameters used to evaluate the performance of the ML-based forwarding-decision models.

Parameters	**Values description**
Vehicles' density	50, 100 150 and 200 $vehicles/km^2$
Data transmission rate	6 $Mbps$
Bandwidth	10 $Mbps$
Message size	Beacon: 155 Bytes, Emergency: 259 Bytes
Source type	Constant bit rate (CBR)
MAC standard	IEEE 802.11p
Simulation time	100 s
Runs (95% confidence intervals)	5 repetitions per point
Simulation tools	OMNeT++ [28]/Veins [68]/Sumo [70]/OSM [5]

TABLE 7.5: Simulation scenarios and VANET routing protocols used in each performance evaluation.

Simulation scenarios	**Description**
Phase 1:	
Simulation urban area and map dimension	Gracia district of Barcelona city, 2300 m x 2100 m
Routing protocols	3MRP [4], MPML (see section 7.6.1), MPANN (see section 7.6.2)
Phase 2:	
Simulation urban area and map dimension	(i) Berlin city area (2500 m x 2500 m) (ii) Rome city area (3500 m/ x 2400 m)
Routing protocols	3MRP [4] and MPANN (see section 7.6.2)

7.6 ML-based proposals to improve forwarding decisions in VANET routing protocols

In this section, we present simulation results of our proposals of routing protocols that include our ML-based forwarding models. In the urban scenarios implemented in the OMNeT++/Veins/SUMO/OSM framework, we consider four vehicle densities (50, 100, 150, and 200 $vehicles/km^2$). We have implemented our proposals of ML-based forwarding models: (i) decision tree (ML_6), (ii) bagged tree (ML_{20}) and (iii) artificial neural network (ANN). These forwarding algorithms have been included in our VANET routing protocols so that nodes can take the best forwarding decision based on their predictions.

We have applied our three proposals of ML-based forwarding algorithms, over a previous proposal named multi-metric map-aware routing protocol (3MRP) developed by the directors of this thesis [4] [15]. All the figures with simulation results present 95% confidence interval (CI) obtained from 5 repetitions per point,

- Firstly, we have implemented the decision tree (ML_6) and the bagged decision tree (ML_{20}) models in 3MRP to see which one will perform better. As we will see in section 7.6.1, the bagged decision tree outperforms the decision tree when they are included in 3MRP to assist the forwarding algorithm. This way, we have called our proposal "bagged decision tree + 3MRP" as multimetric predictive ML-based routing protocol (MPML), see section 7.6.1.
- Secondly, we have applied our ANN-based forwarding algorithm in 3MRP. As a result, we have obtained our proposal named multimetric predictive ANN-based (MPANN) routing protocol, see section 7.6.2. Results show that MPANN outperforms both 3MRP and MPML.
- Finally, MPANN and 3MRP are compared under two other different city maps, see section 7.7.2.

The performance evaluation of the routing protocols for VANETs is done in terms of average packet losses and average end-to-end packet delay. Besides, we have also computed the computational cost of the different proposed forwarding algorithms. With all these measures, we will be able to analyze the results and see the balance between benefits and costs in the performance of our proposed routing protocols especially designed for VANETs.

7.6.1 Multimetric predictive ML-based (MPML) routing protocol

We have implemented our decision tree algorithms, previously presented in section 7.4.1, over our previous proposal of routing protocol for VANETs named 3MRP [4]. The objective is to propose a new prediction forwarding algorithm to select the best candidate node to forward the packet in the next hop. Besides, different proposals will be evaluated aiming to select the best option.

We have set a realistic VANET scenario to carry out the performance evaluation. In the first experiments, we have tested both the simple decision tree model (ML_6) and the bagged decision tree model (ML_{20}) running over the 3MRP [4] routing protocol to select the best next-hop node to forward messages. However, as we will see in the simulation results, we found out that none of the two models offered better results than 3MRP [4] in terms of packet losses. We realized that the reason was that many candidate nodes in the neighborhood were equally scored by the algorithm, so there was a draw among several candidates. To break the tie, we included an additional rule to select the most suitable next hop forwarding node. The process is described as follows.

- It is well known that any ML model learns from a specific dataset made up of different features and classes. In our context, features represent the five metrics (i.e., available bandwidth, distance to destination, nodes' density, MAC layer losses and nodes' trajectory), whereas classes represent two values ("1" means packet successfully delivered and "0" means it is not).

- We have evaluated the importance of each metric as an additional parameter to take the forwarding decision.
- We claim that knowing the importance of each metric in the dataset can help us to solve the issue described above. That is, the additional rule that we have applied in case of draw among several neighbours is to choose the one that shows the highest significant metric.

Fig. 7.12 shows the importance of each metric in the dataset. That is, we have assessed the influence of each metric on the results to see which one is the most decisive. To do so, we have used the **L1 regularization** method. Regularization is a set of techniques that can prevent overfitting in ML algorithms and thus improve the accuracy of the model when facing completely new data from the problem domain. The L1 regularization method (also known as Lasso regression) weights the vector values of a dataset to bring the features without useful information to 0, remaining just those more important metrics. In this way, the incidence distribution of each one of the features is reflected showing a sorted hierarchical importance [112].

According to the results shown in Fig. 7.12, we decided to include the two most decisive metrics (i.e., vehicles' density and trajectory) to assist the ML-based forwarding algorithms to break the tie in case of draw when the node currently holding the packet has to take the forwarding decision. This mechanism is activated only when two or more nodes have the same output value in the evaluation of the decision tree model. In the next subsection, simulation results are presented and analysed.

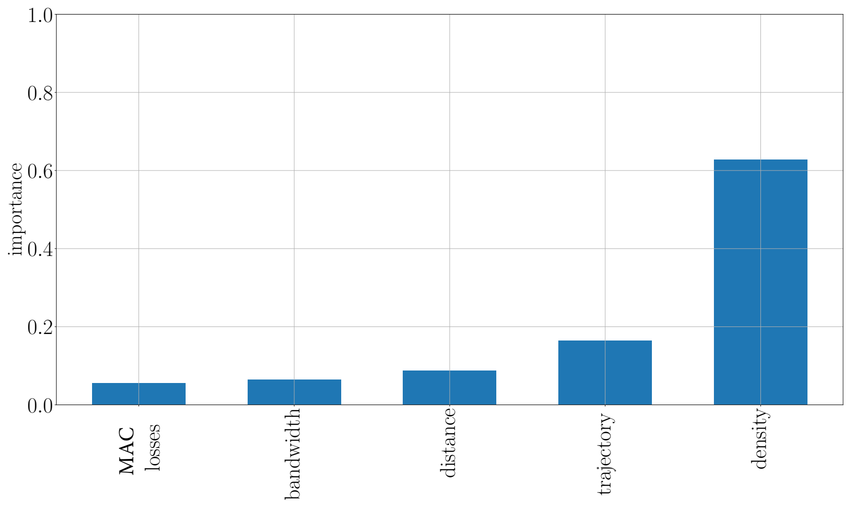


Figure 7.12: Evaluation of the importance of each feature (routing metric) in the forwarding decision: available bandwidth, distance to destination, nodes' density, MAC layer losses and nodes' trajectory.

7.6.1.1 Simulation results. MPML routing protocol.

We have evaluated several versions of our 3MRP [4] [15] routing protocol for VANETs. Each version has incorporated one of our trained ML-based forwarding models with its corresponding configuration. The ML-based forwarding models that we have tried in this performance evaluation are the ones that showed the best performance, see section 7.4. These are the configuration that we have tested:

- First, we have tested an improved 3MRP by incorporating our basic ML-based forwarding models:
  - (i) 3MRP+ML_6: 3MRP routing protocol including the ML-based forwarding algorithm ML_6, i.e. decision tree with tree depth=6.
  - (ii) 3MRP+ML_{20}: 3MRP routing protocol including the ML-based forwarding algorithm ML_{20}, i.e. bagged decision tree with tree depth=20.
- Then, we have evaluated both ML-based forwarding models taking also into account the two most important metrics (vehicles' density and trajectory) to break the tie for equal prediction values of neighbours. This process has been explained at the beginning of this section 7.6.1.
- Therefore, we have these additional combinations:
  - (iii) 3MRP+$ML_6 + Dns$: 3MRP routing protocol including the decision tree with tree depth=6 and vehicles' density as tie-breaker metric ($ML_6 + Dns$).
  - (iv) 3MRP+$ML_{20} + Dns$: 3MRP routing protocol including the bagged decision tree with tree depth=20 and vehicles' density as tie-breaker metric ($ML_{20} + Dns$).
  - (v) 3MRP+$ML_6 + Tjr$: 3MRP routing protocol including the decision tree with tree depth=6 and trajectory as tie-breaker metric ($ML_6 + Tjr$).
  - (vi) 3MRP+$ML_{20} + Tjr$: 3MRP routing protocol including the bagged decision tree with tree depth=20 and trajectory as tie-breaker metric ($ML_{20} + Tjr$).
- The best configuration of all those ML-based forwarding models operating over our 3MRP routing protocol, will be selected as our proposal named **multimetric predictive machine learning-based (MPML)** routing protocol.

Figs. 7.13 and 7.14 show the obtained results in terms of average packet losses, for the different ML-based forwarding algorithms working over the 3MRP routing protocol. We have carried out simulations varying the vehicles' density, ranging from a sparse network (50 $vehicles/km^2$) to a congested network (200 $vehicles/km^2$). We can see that results for the bagged decision tree with tree depth=20 (ML_{20}) and its two variations (always below 40%) are better than those for the decision tree with tree depth=6 (ML_6) and variations (below 60%). This is true except for a very high vehicles' density (200 $vehicles/km^2$), although this density is too high and it is unlikely to happen in urban VANETs. As an alternative, an intermediate solution would be to implement a more complex hybrid mechanism able to switch between $3MRP + ML_{20} + Dns$ in normal conditions and $3MRP + ML_6 + Tjr$ under very high vehicles' density situation (e.g., a traffic jam). However, in next section we will see that ANN notably improves both ML_6 and ML_{20} even including the variations with the tie-breaker metrics.

In addition, we can see in general that after including a tie-breaker metric in the forwarding decisions, a slight improvement has been obtained. The improvement is more noticeable for the bagged decision tree with tree depth = 20. However, in some cases, results are variable and inconsistent. Note that simple decision trees can easily be unstable because small variations in the input data might result in a completely different tree. Conversely, bagged decision trees are able to mitigate this effect. Certainly, we can notice more stable results for 3MRP using ML_{20} and using ML_{20} with its two tie-breaker metric variations, see Fig. 7.14.

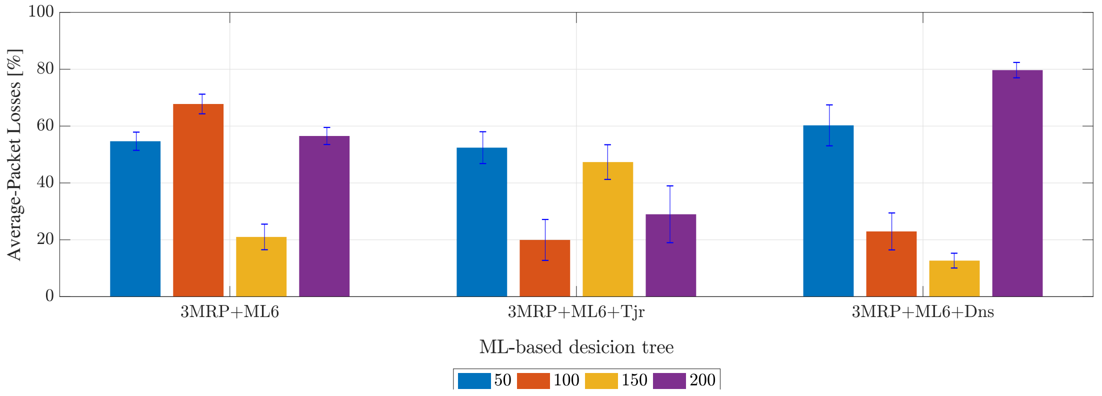


FIGURE 7.13: Average packet losses obtained with 3MRP [4] including the decision tree classifier with tree depth = 6 levels (ML_6). $ML_6 + T_{jr}$ and $ML_6 + D_{ns}$ are variations that include a tie-breaker routing metric (trajectory and vehicles' density, respectively) to break possible ties in the neighbours' score. (5 rep./point, 95% CI).

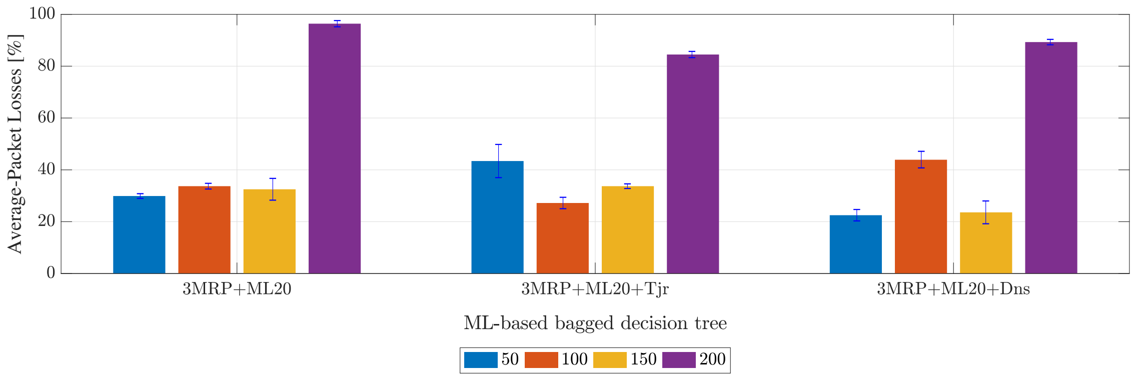


FIGURE 7.14: Average packet losses obtained with 3MRP including the bagged decision tree classifier with tree depth = 20 levels (ML_{20}). $ML_{20} + T_{jr}$ and $ML_{20} + D_{ns}$ are variations that include a tie-breaker routing metric (trajectory and vehicles' density, respectively) to break possible ties in the neighbours' score. (5 rep./point, 95% CI).

Figs. 7.15 and 7.16 show the average end-to-end packet delay. We can appreciate that in both cases considering the trajectory metric together with the ML-based forwarding algorithm helps to decrease the delay. Certainly, the trajectory prioritizes those vehicles moving faster towards destination, so consequently the delay will decrease.

Overall, these simulation results show that the best ML-based forwarding algorithm to assist our previous 3MRP [4] multimetric routing protocol is obtained when 3MRP includes the $ML_{20}+Tjr$ ML-based forwarding algorithm, i.e. the bagged decision tree with tree depth = 20 together with the trajectory as tie-breaker metric. We have called our approach as **multimetric predictive machine learning-based (MPML)** routing protocol. Anyway, we can see in Fig. 7.14 that $3MRP + ML_{20} + Tjr$ offers packet losses that still are too high, around 40% and even higher up to 85% for very congested VANETs. Therefore, we need to find a better prediction model to enhance the 3MRP performance in VANETs. In next section, we present our approach of a multimetric routing protocol for VANETs that includes our proposal of an ANN-based forwarding model.

Notice that in this section we are evaluating our ML-based proposals decision tree and bagged decision tree, as well as their variations including one tie-breaker metric (trajectory or vehicles' density). This way, we will see which is the optimal combination that is going to be named as MPML. In section 7.7 we will carry out a complete performance evaluation of MPML compared to our proposal MPANN (see next section), and also GPSR and 3MRP as references.

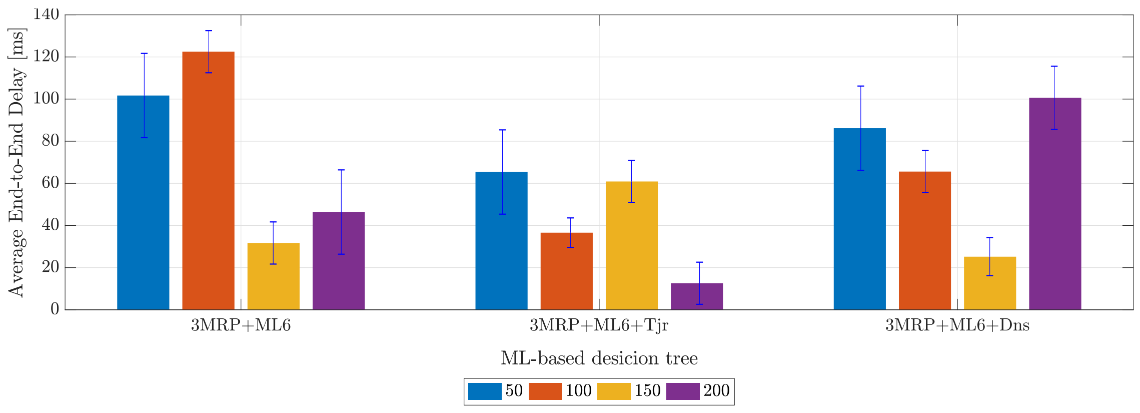


FIGURE 7.15: End-to-end average packet delay of 3MRP using our decision tree classifier with maximum depth equal to 6 levels. (5 rep./point, 95% CI).

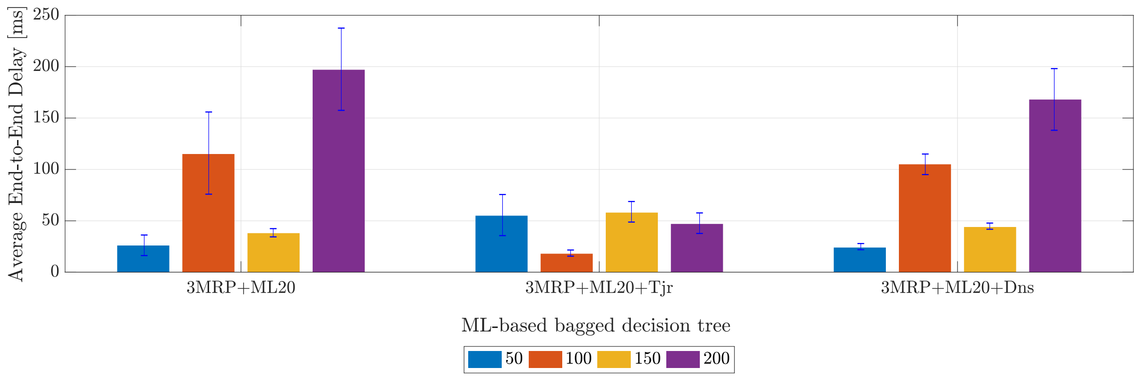


FIGURE 7.16: End-to-end average packet delay of 3MRP using our decision tree classifier with maximum depth equal to 20 levels. (5 rep./point, 95% CI).

7.6.2 Multimetric predictive ANN-based (MPANN) routing protocol

In this section we present simulation results of our proposal named **multimetric predictive artificial neural network-based (MPANN)** routing protocol. MPANN includes our ANN-based forwarding model, described in section 7.4.2, in our previous 3MRP [4] routing protocol to significantly enhance its performance.

MPANN has been trained with the parameters described in Table. 7.3. Similarly to our previous proposal (MPML), MPANN has been evaluated using the area of Barcelona represented in Fig. 7.1. In the following subsections, MPANN will be evaluated in terms of average percentage of packet losses and average end-to-end packet delay.

7.6.2.1 Simulation results. MPANN routing protocol

Fig. 7.17 shows the average percentage of packet losses for each vehicles' density for our MPANN routing protocol, compared to our proposal MPML presented in the previous section. As it can be seen, especially for very high (200 vehicles/km^2) and very low (50 vehicles/km^2)) vehicles' densities, the average packet losses are lower by far than those obtained with our MPML proposal. Specifically, for MPANN, losses are below 23% for very sparse (50 vehicles/km^2) and very congested (200 vehicles/km^2) vehicles' densities. In particular, MPANN notably outperforms MPML when the vehicles' density is 200 vehicles/km^2. For vehicles' densities is medium-high (100 to 150 vehicles/km^2), packet losses for MPANN are approximately 16% lower than for MPML. Note that for 100 to 150 vehicles/km^2 there is a good vehicles' density that improves network connectivity with a low chance of collisions, which favours communications.

Notice that in this section we are evaluating our new proposal MPANN compared to the best ML-based proposal using decision tree that we presented in the previous section, i.e. MPML. In section 7.7 we will carry out a complete performance evaluation of MPML compared to our proposal MPANN (see next section), and also GPSR and 3MRP as references.

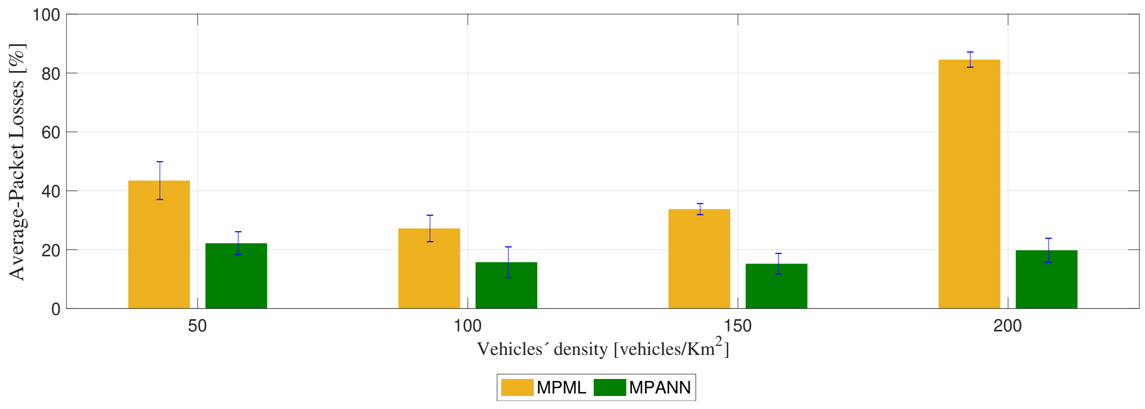


FIGURE 7.17: Average percentage of packet losses for MPANN and MPML routing protocols under different vehicles' densities. (5 rep./point, 95% CI).

Fig. 7.18 shows the average end-to-end packet delay for MPANN vs. MPML under different vehicles' densities. Note that for a very sparse density of vehicles there is a low connectivity between network nodes, which increases packet delivery delay at destination. Additionally, in case of a very high vehicles' density, the network is more congested, which also increases packet delivery delay at destination. The average end-to-end packet delay for MPANN ranges from $35ms$ to $42ms$, whereas it ranges from $28ms$ to $82ms$ for MPML. Besides, a lower delay is obtained with medium-to-high densities (100 to 150 vehicles/km^2). In those cases, the network is not congested and has a suitable number of neighbours to forward messages, and the average end-to-end packet delay is low (under 20 milliseconds).

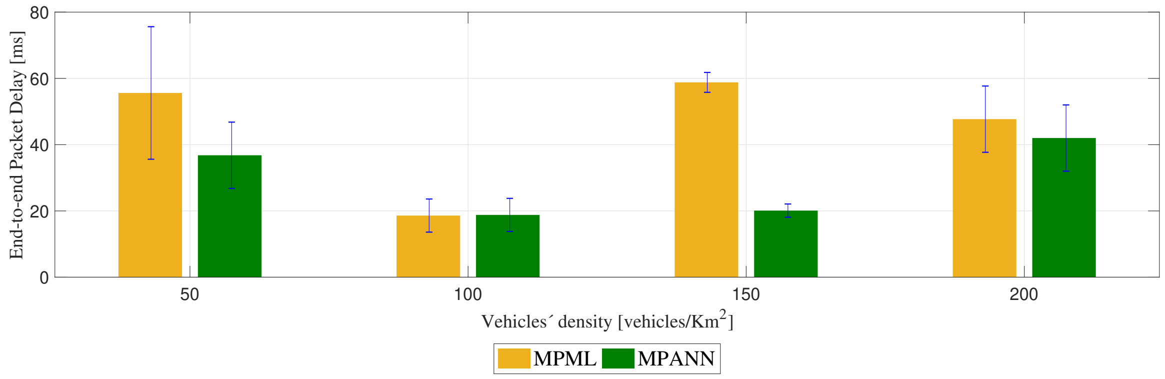


FIGURE 7.18: Average end-to-end packet delay for MPANN and MPML routing protocols under different vehicles' densities. (5 rep./point, 95% CI).

To sum up, we can see that our ANN-based forwarding algorithm performs significantly better than ML-based forwarding models using decision trees. Results show that our MPANN proposal is able to adapt better than MPML to the inherent VANET conditions, predicting more efficiently with unseen data. In the next subsections, a further comparison with four routing protocols for VANETs is done: our two ML-based forwarding algorithms MPML and MPANN, compared to the well-known GPSR and also to 3MRP (from which our proposals are developed) as a reference.

7.7 Performance evaluation of MPANN and MPML under different urban scenarios

In this subsection, we will carry out a performance evaluation of proposals under different VANET scenarios, which were described in section 7.5. The performance evaluation of each routing protocol is evaluated in terms of average percentage of packet losses and average end-to-end packet delay. We have also measured the run-time to perform the simulations, the computational cost (in terms of time), and the overhead. In this way, we will be able to analyze the pros and cons of our proposals, weighing the benefits and costs incurred.

We have organized our simulations in two groups:

Test 1. First, we will compare our proposals MPANN and MPML with GPSR [3] as a reference and to our former proposal 3MRP [4]. We will use the same urban scenario depicted in Fig. 7.1 as it was the scenario employed to generate the dataset used to train our ML-based forwarding models. Additionally, we have included the well-known GPSR [3], and the 3MRP [4] routing protocols in the evaluation, and they are compared with our ML-based routing protocols MPML and MPANN.

Test 2. Second, we will evaluate and compare our best proposal of ML-based routing protocol (MPANN) in other VANET scenarios different than the scenario used to train the designed ML-based forwarding models. In this way, we can see how adaptable our approach is to new scenarios with unseen data. In this evaluation we will only include the ML-based forwarding model that showed the best performance in the previous Test 1. To this end, we have considered two new scenarios with different characteristics to carry out this new performance evaluation:

- Berlin. Area similar to the Barcelona city map used in Test 1, but with a more complex oreography. Details were given in section 7.5. The Berlin city scenario is shown in Fig. 7.1.
- Rome. Area much more complex than the Barcelona map. Details were given in section 7.5. The Rome city scenario is shown in Fig. 7.11.

7.7.1 Test 1: Analysis comparison of our proposals of VANET routing protocols in the Barcelona scenario

This set of simulations were carried in the same simulation scenario used to collect the data and create the dataset with which we trained the ML-based models, see Fig. 7.1. Of course, we have used different simulation seed so the simulations are different, although the simulation settings (map, area, street layout...) are the same.

We show results for four vehicles' densities, ranging from very sparse VANETs (50 vehicles/km^2) where it is hard to find vehicles around and connectivity is low, to very congested VANETs (200 vehicles/km^2) where the chance of packet collision is high. In this way, we model various VANET scenarios, including low traffic intervals during the night or the weekend, normal traffic flows during weekdays or even congested situations during traffic jams.

7.7.1.1 Evaluation of the average percentage of packet losses and the average end-to-end packet delay

Fig. 7.19 shows the average percentage of packet losses for the four routing protocols for VANETs analyzed: GPSR [3], 3MRP [4], MPML (sec. 7.6.1) and MPANN (sec. 7.6.2). We can see that GPSR, which uses only distance to destination as routing metric, in general offers the highest packet losses between 62% to 80%. We can see that 3MRP [4] shows better results than both the reference GPSR and our proposal MPML in terms of packet losses. With MPANN losses are below 25% in all vehicles' densities, whereas MPML shows losses below 40% except for 200 vehicles/km^2 with losses reaching 80%.

Obviously, our proposal MPML is not able to outperform our former multimetric proposal 3MRP specially for a very high vehicles' density (200 vehicles/km^2). This might happen in ML-based models that use decision trees, whose hyperparameters (i.e., tree depth and tree breadth) have to be chosen and whose dataset should be as much representative and large as possible. Configuration and learning phases are very sensitive to ensure a good accuracy in the correct prediction of the output during the performance evaluation phase.

Fortunately, our ANN-based proposal MPANN is able to outperform 3MRP in all vehicles' densities. This fact saved us of having to do many more simulations to substantially enlarge the dataset. Note that to obtain our dataset (which has 6000 registers) we have dedicated about 840 hours of human work and about 3600 hours of simulation, see table 7.1. In Fig. 7.19 we can see that for MPANN, where the forwarding decisions are based on deep learning, a significant improvement is obtained for all vehicles' densities. With our MPANN routing protocol, the average packet losses are less than 21% for all densities.

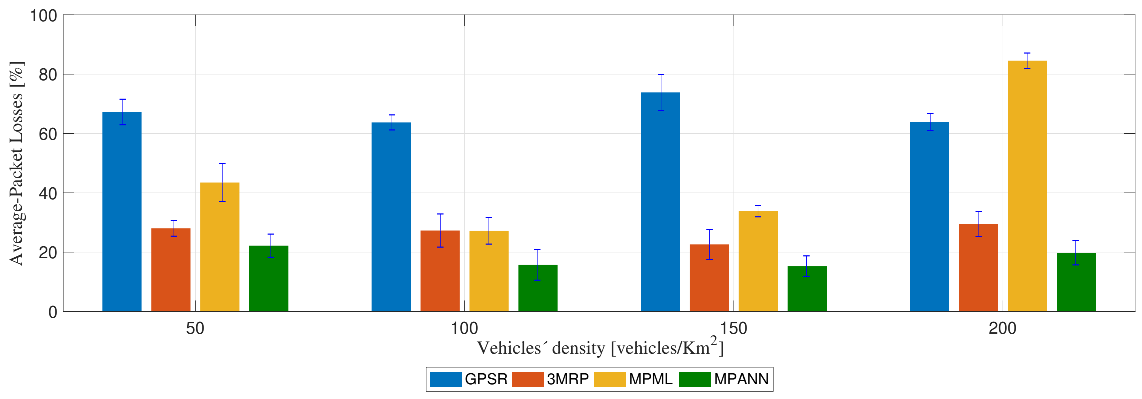


FIGURE 7.19: Average percentage of packet losses for different vehicles' densities. (5 rep./point, 95% CI).

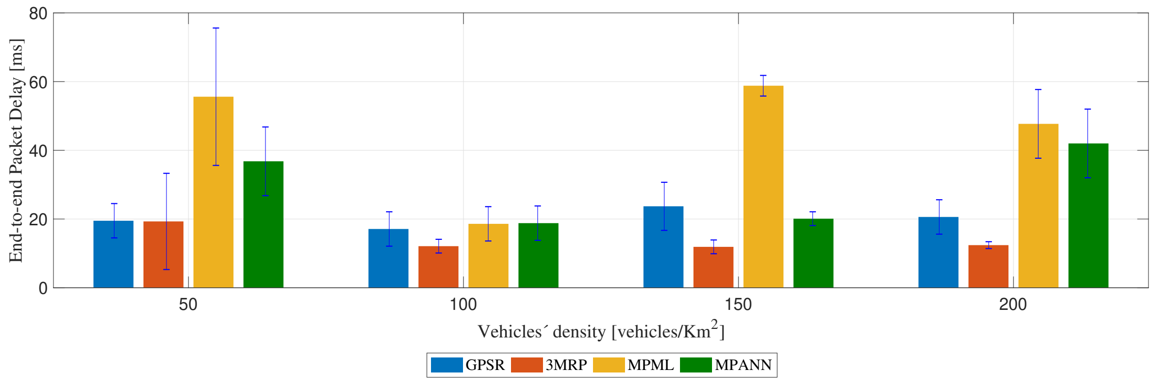


FIGURE 7.20: Average end-to-end packet delay for different vehicles' densities.

Fig. 7.20 shows the average end-to-end packet delay for the same four routing protocols, and for diverse vehicles' densities. We can see that the highest delays correspond to our routing protocols, i.e. MPML and MPANN. The reason is that since MPANN is able to lose less packets than 3MRP, many of those packets followed longer paths and got to destination, consequently increasing the average end-to-end packet delay. Notice that the ML-based predictive models included in MPML and MPANN do not consume any significant amount of time, as it is shown in section 7.7.1.3.

The increment on delay in MPANN is very small (maximum 40 ms) compared to 3MRP. Nevertheless, this increase pays-off given the reduction in packet losses especially for MPANN. To further assess the pros and cons of our proposals, we show in the next subsections the evaluation of the four routing protocols in terms of running time and computational cost.

7.7.1.2 Evaluation of the simulation running time

The running time represents the time that each routing algorithms needs to complete a running simulation for a specific configuration of the simulation settings, i.e. time to obtain a point in each figure. OMNeT++ [28] is an event-based network simulator, and the time spent for each simulation depends on the number of participating nodes. For each node that forwards the message, a set of operations must be executed. The running time increases as the number of nodes increases. Also, the running time increases as the network area increases since longer forwarding paths are potentially needed to reach destination (number of hops potentially would increase).

Fig. 7.21 shows the running time taken for each routing protocol to perform 100 simulation seconds, i.e. simulation time to obtain a single point in the figures of the performance evaluation. For this experiment, we have considered an intermediate vehicular density of 100 $vehicle/km^2$. Simulations were carried out in a server with the features presented in Table 7.6. In that figure we show the average of 10 simulations including CI of 95%.

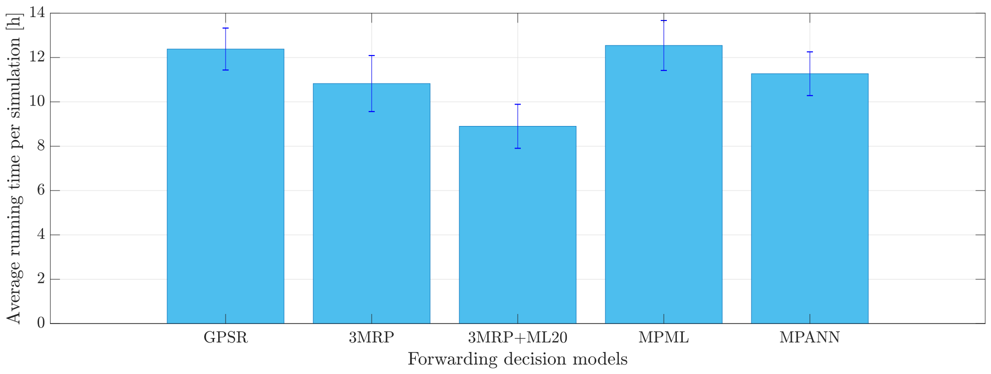


FIGURE 7.21: Run-time consumed in a 100 sec-simulation in OMNeT++/Veins/SUMO/OSM. Vehicles' density = 100 $vehicles/km^2$, 95% CI.

TABLE 7.6: Hardware features of the server used to run simulations to obtain Fig. 7.21.

Server	Ubuntu 16.4 LTS
RAM	251.4 GiB
Processor	Intel® Xeon(R) Gold 6138
Base clock frequency	2.00 $GHz \times 47$
OS type	64 bits

Fig. 7.21 shows that GPSR [3] experimented the highest running time, about 12 hours per point on average. This protocol selects the forwarding nodes based on the distance metric, and when it switches to the perimeter routing mode, the delay can increase. With this perimeter mode, long routes to the destination node (RSU) can be taken; even in the worst case, routing loops may appear. Therefore, there are more events to be processed, which means a higher running time. On the other hand, 3MRP [4] requires less running time than GPSR. Although 3MRP performs more operations since it handles five routing metrics (distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses), the message forwarding is more efficient. That is, 3MRP selects better forwarding nodes at every hop towards destination, resulting in lower hop-count will and fewer nodes have to process messages.

Notice that in Fig. 7.21 we have included the running time spent by the 3MRP routing protocol using our ML_{20} ML-based forwarding algorithm (3MRP+ML_{20} column). That is, we were interested in assessing the effect of including the trajectory as tie-breaker metric in the ML_{20} algorithm. We can see that 3MRP+ML_{20} took almost 9 hours to finish the simulation whereas MPML (i.e., 3MRP+$ML_{20} + T_{jr}$) took 12 hours. This 3-hours extra running time is the price to pay with MPML, which is able to decrease packet losses to 21% (see Fig. 7.19) while keeping low packet delays below 40 ms. (see Fig. 7.20). In the same figures we can see that 3MRP [4] offers packet losses below 30% and packet delays below 20 ms. That increase in delay comes from including the trajectory as tie-breaker metric, which needs additional comparisons with every neighbour. Anyway, the additional simulation time needed by MPML over 3MRP only affects the researcher that evaluates the different proposals. What really matters is the increase in computational cost that implementing our proposals would cause in real nodes. Given the impossibility to use real smart vehicles in our experiments, in next section we present the computational cost in terms of time required by our proposals. As we will see, this computational time is really insignificant.

Also notice that 3MRP+ML_{20} shows less running time (almost 9 hours) compared to 3MRP (almost 11 hours). Although the ML predictor also considers the same metrics than 3MRP for route calculation, this computation is only based on a set of comparisons (*if-then-else*) that do not demand significant extra computational time. Instead, 3MRP computes a multimetric score for each candidate node using weights over the five corresponding metrics. That computation is done for every node that has to forward a packet. The computation of the weighted multimetric score used by 3MRP takes longer than the *if-then-else* comparisons used by ML_{20}.

Finally, we can see that MPML (12 hours) took 1 hour more than 3MRP (11 hours) and MPANN (11h:30m) took 30 min more than 3MRP. The extra delay incurred by MPML is due to the ML-based forwarding algorithm (the time needed by the set of *if-then-else* comparisons) and the extra arrangement of candidate nodes according to the tie-breaker metric of trajectory. The extra delay incurred by MPANN is due to the ANN-based forwarding algorithm and the extra arrangement of candidate nodes according to the tie-breaker metric of trajectory.

Nonetheless, this running time is just a researcher work time needed to assess the performance of any new proposal. Once the performance evaluation has been analysed, what really matters is the computational time needed by the nodes (vehicles) to run our routing protocols. This measure is also important to be taken into account in the trade-off between pros and cons of our proposals. As we will see in the next two sections, the computational cost incurred by our proposals MPML and MPANN is below 0.2 msec, which is irrelevant considering the improvements obtained in terms of percentage of packet losses and packet delay.

7.7.1.3 Computational cost, in terms of computational time

In this subsection, we have calculated the computational cost in terms of time of our ML-based forwarding algorithms, i.e. ML_6, ML_{20} and ANN. This metric is frequently used to measure the efficiency and viability of a designed algorithm. To calculate the computational cost we have analyzed the time incurred by each one of the forwarding algorithms used in the routing protocols. We have made two experiments to obtain the computational time of our proposals using Matlab: (i) A first experiment varying the vehicles' density, and (ii) a second experiment for a variable vehicles' density throughout time. For these two experiments, we used the hardware described in Table 7.7, see subsection 2.4.4. In both experiments, we generate a traffic of 2 packets/sec. In this way, we are able to replicate in Matlab an equivalent sequence of packets transmitted that we nodes would process in our OMNeT++/Veins/SUMO/OSM simulation framework.

Table 7.7: Hardware specifications of the personal computer used to obtain the computational cost in terms of time.

Computer	macOS Sierra Version 10.12.6
Processor	Intel i7
Base clock frequency	2.9 GHz
Memory	8 GB 1600 MHz

(i). **First experiment**. In Matlab, we have generated random values (in the range [0, 1]) for each normalized routing metric (i.e., distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses). Vehicles have a transmission range of 340 meters. We have considered four different values of the vehicles' density: 50, 100, 150, and 200 vehicles/km^2. We have used those values in Matlab to replicate an equivalent process of evaluating each neighboring node when sending a packet that we would process in our OMNeT++/Veins/SUMO/OSM simulation framework.

Fig. 7.22 shows the computational cost (in terms of time) of the evaluated ML-based forwarding algorithms designed for VANETs. ANN is the ML-based forwarding algorithm with the highest computational cost. ANN shows an increment of 0.02 msec in most of the cases with respect to ML_{20} and ML_6. However, these values represent insignificant time ranges, compared to the end-to-end average packet delays shown by ANN in the range 20-40 milliseconds, as it is depicted in Fig. 7.20. As it can be seen in Fig. 7.22, the maximum value (time demanded by the CPU) is 0'028 msec. obtained when the vehicles' density is the maximum considered (200 vehicles/km^2), and thus the number of operations is maximum too. In contrast, for the other two ML-based forwarding algorithms the computational cost in average is in the range of 0.002 msec. The delay increment is due to the cost of the additional operations included in the forwarding algorithms, especially in ANN.

(ii). **Second experiment**. In this Matlab test, we have set the number of neighbours per vehicle to be a random value in the range [1, 50]. This range of number of vehicles in OMNeT++/Veins/SUMO/OSM would correspond with a vehicles' density defined in the range [2'75, 138] vehicles/km^2 for a transmission range of 340m. In Fig. 7.23 we can see that ANN produces a computational time of 25 μsec., whereas with ML_6 and ML_{20} it is around 1μsec. Even ANN gives 25 more times computational time than the others, this value still is very small compared to the packet flow rate, so packets can be processed quickly without any problem.

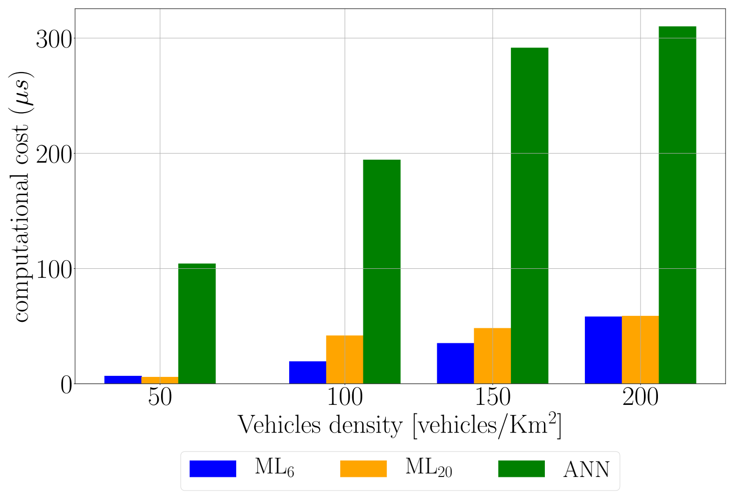


FIGURE 7.22: Computational cost (in time) of our ML-based forwarding algorithms measured in Matlab, for four vehicles' densities.

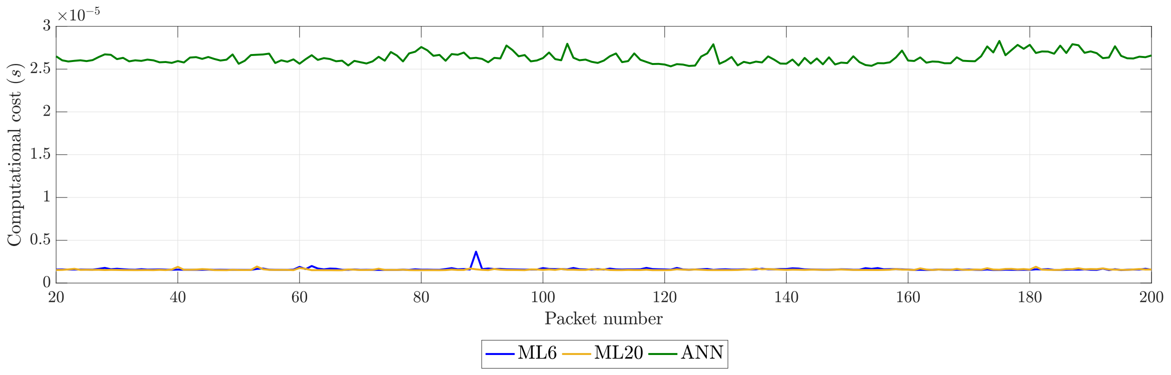


FIGURE 7.23: Computational cost (in time) of our ML-based forwarding algorithms measured in Matlab, for each incoming packet measured in a general node.

In summary, using Matlab we will emulate the same sequence of operations done over each packet and for each neighbour as we would have had in the simulation framework. Then, in Matlab it is easy to calculate the computational time dedicated per packet by the different ML-based forwarding algorithms. Then, we can calculate the total computational time necessary to process all the packets sent in a 100-sec simulation.

We have obtained the time spent by each ML-based forwarding algorithm (see Figs. 7.22 and 7.23). This way, we can assess the trade-off between benefits (shown in section 7.7.1.1) and costs in terms of computational time. Effectively, the computational time needed is very low and it rewards the benefits of MPANN in terms of average percentage of packet losses (see Fig. 7.19) and average end-to-end packet delay (see Fig. 7.20).

7.7.1.4 Overhead

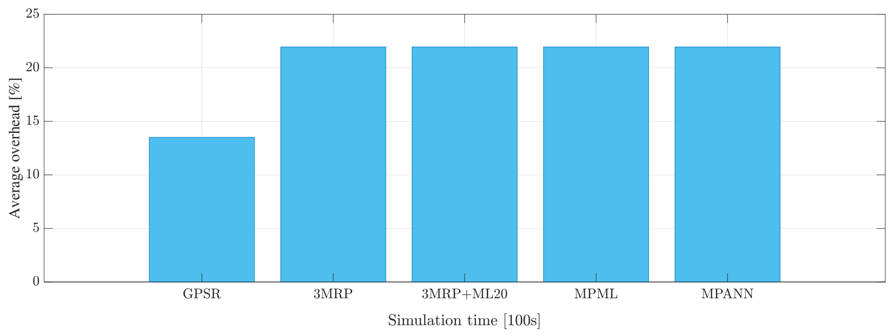


FIGURE 7.24: Overhead incurred by each ML-based routing protocol (3MRP+ML_{20}, MPML and MPANN) compared to GPSR [3] and 3MRP [4].

Fig. 7.24 shows the overhead incurred by each one of the routing protocols analyzed in this performance evaluation. Although GPSR [3] has the lowest overhead (14%), it produces the highest percentage of packet losses and highest average packet delay, see Fig. 7.19 and Fig. 7.20, respectively. On the other hand, the other routing protocols (3MRP, 3MRP+ML_{20}, MPML and MPANN) increase the overhead to 22% (i.e., a relative 57% higher amount). The reason is that those four routing protocols use additional fields (whose size is from 2 to 8 bytes) in the hello messages to calculate the routing metrics. That is, our proposals MPML and MPANN use the same additional fields included in our former protocol 3MRP [4], so there was no need to increase the overhead. Those fields are updated by each node at the moment of sending the current hello message. Those additional fields correspond to record the (v_x, v_y) velocity coordinates of the node, a field for the MAC loss layer value, and a field for the nodes' density value. The explanation about the routing metrics (distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses) calculation were given in section 3.2.2.1.

These routing metrics, together with the distance to destination (included by default in GPSR [3]) are used by the forwarding algorithm to take the forwarding decision and choose the next node to forward the packet. Besides, the ML-based routing protocols (MPML and MPANN) do not produce any additional overload compared to 3MRP, since they use the same beacons structure already established in 3MRP [4]. Although our prediction models include a small extra cost in terms of packet delay (delay in MPANN is maximum 20 msec. higher than in 3MRP, according to Fig. 7.20), the benefits obtained in terms of packet losses for MPANN are significantly notable, as it is shown in Fig. 7.19. This improvement is achieved in MPANN without having introduced any additional overhead with respect to 3MRP (see Fig. 7.24).

7.7.2 Test 2: Performance evaluation of the proposed MPANN routing protocol in other city maps

In this second test, we have considered two different VANET scenarios where we have evaluated our proposal MPANN (see section 7.6.2) and our previous proposal 3MRP [4] routing protocols. These scenarios are different from the Barcelona city (Figs. 7.1), the scenario used to gather data and create our dataset. In this last phase of our performance evaluation we want to assess the flexibility of our proposals over other city maps. We have evaluated the performance of those routing protocols in an area of Berlin (Figs. 7.10 (similar in size and more complex in street layout) and in an area of 7.11) (larger in size and much more complex in street layout).

Our goal is to validate if the prediction model of our ML-based forwarding algorithms can classify correctly data in different city maps (different from the map from which we took the data to train train the models). Besides, we have compared only protocol MPANN with the multimetric 3MRP [4] because the neural network model performs better than the decision trees, as we have already shown in the previous sections of this chapter.

Fig. 7.25 shows the average percentage of packet losses for both MPANN and 3MRP [4] routing protocols, for the three different city maps, and for four vehicles' densities. As it can be seen, our MPANN routing protocol performs better in the three urban scenarios, even in the more complex city map of Rome. Notice that for both routing protocols, the highest percentage of packet losses corresponds to the Rome scenario, since the area of Rome is more complex with narrow and one-way streets. The percentage of packet losses with MPANN is around 55% for a sparse vehicles' density and it improves for the 150 vehicles/km^2 density to around 30%, since the network connectivity improves. Overall, we see that MPANN outperforms 3MRP in around 5-20%.

On the other hand, MPANN over the Berlin map predicts more accurately than 3MRP, for the four vehicles' densities. MPANN achieves an average packet losses from 8% for 100 vehicles/km^2 to a maximum of 18% for 200 vehicles/km^2.

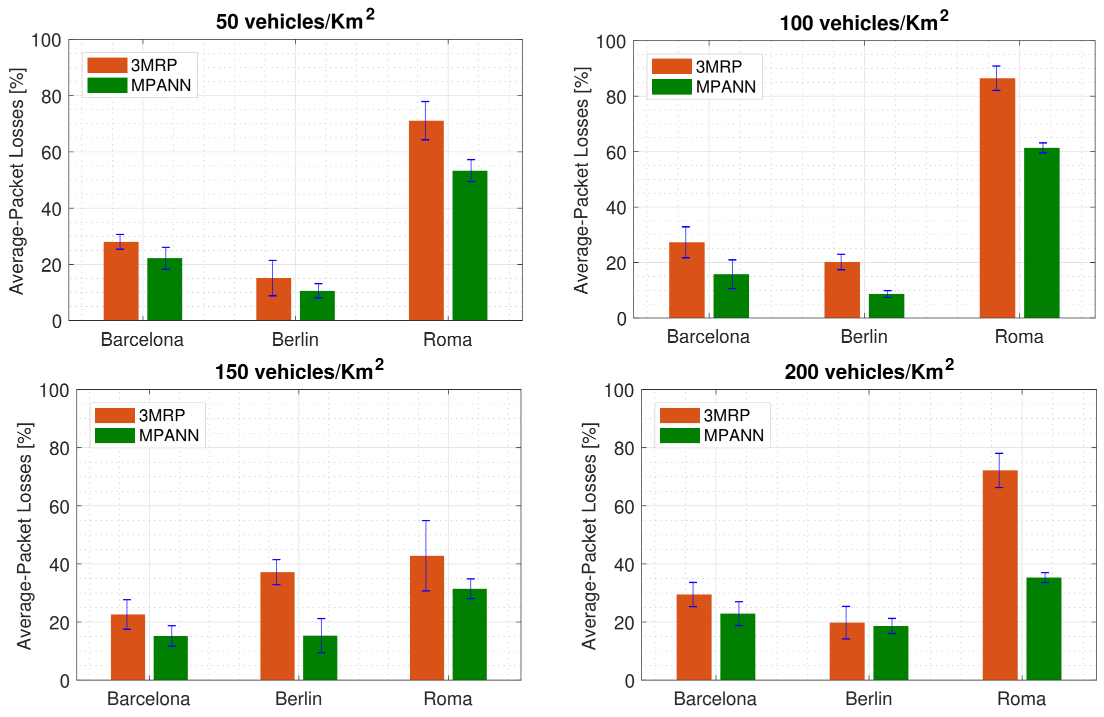


FIGURE 7.25: Average packet losses for MPANN vs. 3MRP [4] in the 3 city maps considered. (5 rep./point, 95% CI).

Fig. 7.26 shows the average end-to-end packet delay for both MPANN and 3MRP [4] routing protocols, for the three city maps and for the different vehicle densities. On the one hand, for the same map used to create the dataset (Barcelona city area), delays with MPANN are higher than with 3MRP [4], in around 18 to 40 milliseconds. However, MPANN offers a lower percentage of packet losses than 3MRP in around 5-20%, see Fig. 7.25.

In the Berlin city area, which has a similar street layout as Barcelona city area, MPANN shows a similar average end-to-end packet delay for the four vehicles' densities, around 10 msec. 3MRP offers more variable delays, from 10 msec. for 100 and 200 vehicles/km^2 to 50 msec. for 150 vehicles/km^2. In the Rome scenario, MPANN shows average packet delays under 20 msec. for the four densities, whereas 3MRP offers a maximum delay around 60 msec for 200 vehicles/km^2.

Overall, we can conclude that our ANN-based proposal of a routing protocol for VANETs called MPANN, shows a good adaptation to the similar Berlin scenario. However, a low prediction power is presented in the Rome scenario. The reason is that the model does not have enough experience with a urban configuration so complex as the one of the Rome city. Although the model can face new scenarios, its prediction power fails when the city layout differs a lot from the city map used to create the dataset.

To tackle this issue we should include a larger amount of registers in our dataset, taken from diverse cities with different street layouts, and then train the ANN-based model again. This is one of the goals pointed out in the future work that will follow the research work of this thesis.

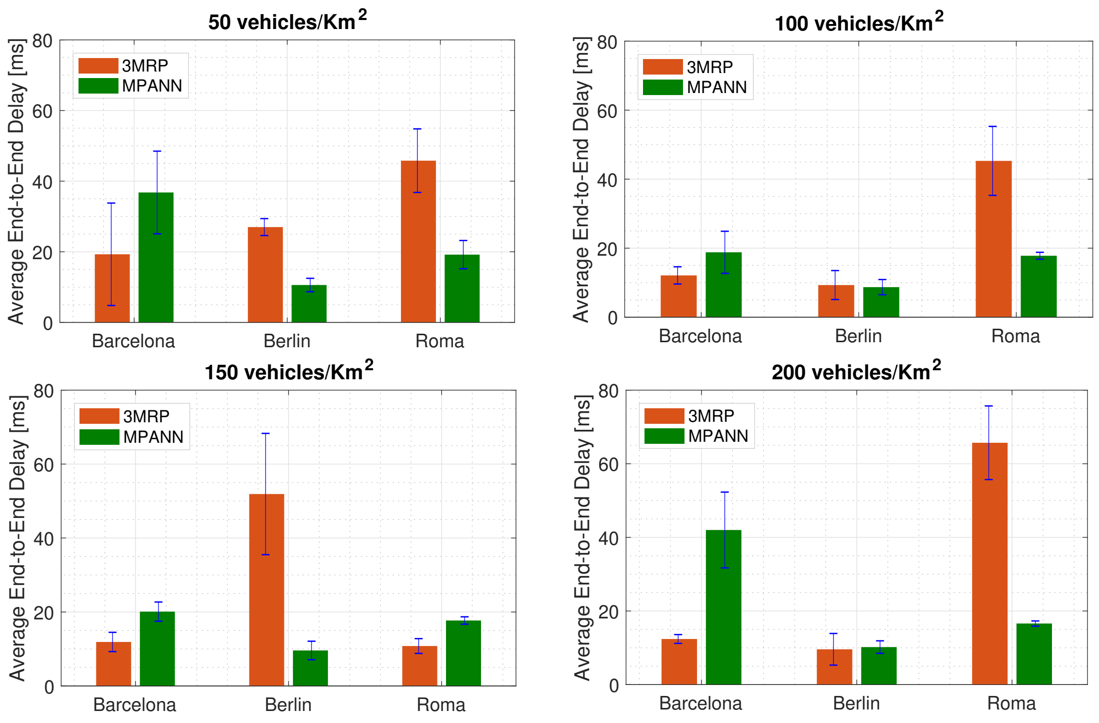


FIGURE 7.26: Average end-to-end packet delay for MPANN vs. 3MRP [4] in the 3 city maps considered. (5 rep./point, 95% CI).

7.8 Conclusions

In this chapter we have presented two novel ML-based multimetric routing protocols for VANETs that are based on machine learning decision trees and on artificial neural networks: (i) Multimetric predictive ML-based (MPML) routing protocol (see section 7.6.1) and (ii) multimetric predictive artificial neural network-based (MPANN) routing protocol (see section 7.6.2).

For this purpose, we have created a dataset from a high number of simulations regarding an urban scenario of a representative area of Barcelona city. In our simulations we have used a wide range of vehicles' densities, to gather data under sparse VANETs (50 vehicles/km^2) as well as under more dense scenarios (200 vehicles/km^2). Our dataset has a size of 6000 registers correspondent to the five routing metrics (distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses) computed from the data interchanged between nodes through periodic hello messages.

The dataset collected has been used to train the different learning algorithms that we have designed and evaluated. Besides, the different prediction models have been validated using traditional machine learning performance metrics (accuracy, ROC, AUC) and we have chosen those that showed a better performance, see section 7.4.

We have designed three learning models: (i) one simple decision tree with 6 levels of depth (ML_6), (ii) bagged decision tree with 20 levels of depth (ML_{20}), and (iii) a feed-forward artificial neural network (ANN-based forwarding model). Those ML-based forwarding models have been implemented in our previous routing protocols 3MRP over our simulation platform OMNeT++/Veins/SUMO/OSM as the routing protocol. The design of our ML-based forwarding algorithms has been presented in section 7.4.

To evaluate our proposals we have organized a performance evaluation using two tests (see section 7.7). On the one hand, we have used the same city map with which the dataset was created, i.e. a 2300m x 2100m area of Barcelona. Results of this Test 1 are shown in section 7.7.1. On the other hand, we have performed a second Test 2 using two different city maps: a similar area of Berlin and a larger and more complex area of Rome. Our goal was to measure the level of flexibility and adaptation attained by our designed ML-based forwarding models MPML and MPANN. For the experiments, we have configured four different vehicles' densities. We have compared our MPML and MPANN learning models with GPSR [3] as a well-known reference and with our previous proposal of multimetric routing protocol name 3MRP [4] protocols. We have carried out a performance evaluation whose results have been discussed in section 7.7.2.

In our performance evaluation of the diverse configurations of the ML-based forwarding algorithms (see Fig. 7.13 and Fig. 7.14), the decision tree configuration with a maximum tree depth of 20 (3MRP+ML_{20}) shown better results than the option with a simple decision tree with tree depth=6 (3MRP+ML_6). One could think that the deeper the tree, the higher the prediction accuracy. Nevertheless, this increase could lead to overfitting issues, which happens when the model shows high precision during the training phase but falls with low precision with the test or data not seen, see Fig. 7.5. For this reason we set a maximum tree depth = 20 to avoid that issue.

On the other hand, we have evaluated the importance of each routing metric (i.e., available bandwidth, distance to destination, nodes' density, MAC layer losses and nodes' trajectory) in the training dataset. The feature importance has been computed with the well-known *L1 regularization* technique [112]. From this analysis, the density and trajectory shown to be the most important and decisive metrics. Including the trajectory as tiebreaker metric (3MRP+ML_{20}+Tjr), improvements in results have been obtained either for low and medium-high densities. We have selected that configuration (3MRP+ML_{20}+Tjr) as our ML-based routing protocol named multimetric predictive machine learning (MPML). The performance evaluation and results have been discussed in section 7.6.1.1. Besides, MPML demands just a slightly higher runtime and computational cost compared to the basic 3MRP (see Figs. 7.21 and 7.22).

Additionally, in section 7.7.2 we have evaluated our best proposal (MPANN) in three different scenarios, and we have compared its performance to 3MRP [4] as a reference. The MPANN routing protocol includes our proposal of ANN-based forwarding algorithm. (i) In a first test we have done a simulation analysis with the same city map of Barcelona that we used to collect the data to create the dataset used to train the ML-based forwarding algorithms. In this case, the MPANN predicts the next-hop node more accurately than 3MRP, which leads to fewer packet losses about 21% with respect to 3MRP (see Fig. 7.19), but with a slight increment in delay about 40 milliseconds higher than with 3MRP (see Fig. 7.20). (ii) In a second test, we have evaluated the capacity of the MPANN prediction algorithm under two different city maps. That is, we have tested the prediction algorithm in new city environments. For that, we have used two additional city maps: an area of Berlin similar to the Barcelona map although with a little more complex streets layout, and a larger and much more complex area of Rome. In particular, the MPANN algorithm performs better than 3MRP in all vehicles' densities in the Berlin map achieving packet losses lower than 20% for a crowed traffic density, and lower than 8% in a sparse traffic density (see Fig. 7.25), which are good results for VANETs scenarios. In the much more complex city map of Rome, the algorithm had more difficulties to predict accurately, achieving losses in the range of 35% to 55%. However, it still performed better than 3MRP in all cases. The performance evaluation and results have been discussed in section 7.7.

We have also evaluated the performance of our proposals in terms of (i) simulation run-time, (ii) computational cost (in terms of time) and (iii) overhead:

(i). In Fig. 7.21 we see that the run-time of MPANN, MPML and 3MRP are around 11h 30min, 12h and 11h, respectively.

(ii). ANN shows an increment around 0.02 msec in the computational time with respect to ML_{20} and ML_6, see Fig. 7.22. However, these values are very small compared to the end-to-end average packet delays shown by ANN in the range 20-40 ms, see Fig. 7.20.

(iii). Our proposals MPML and MPANN do not add any extra overhead compared to 3MRP, since both routing protocols require the same header fields of the beacon messages as 3MRP [4] to carry the five routing metrics (distance to destination, trajectory, vehicles' density, available bandwidth and MAC losses). The performance evaluation and results have been discussed in section 7.7.

Finally, we can conclude that our designed MPANN routing protocol adapts very well in scenarios with similar streets layout as the Barcelona scenario. Note that the data was collected using an area of Barcelona city to create the dataset used to train the ML-based forwarding models. We selected a general and representative area of Barcelona that includes wide and narrow streets as well as an avenue. However, we also conclude that it would be still advisable to learn more with new data taken from other different scenarios to further generalize it. This way, we would obtain a model able to adapt to most of the kind of city environments and street layouts. As it is pointed out in the next chapter, as future work we will extend our dataset with new values from new scenarios using heterogeneous city maps with very different street layouts and orography. Nevertheless, it would just be necessary to conduct a large amount of new simulations and process the data properly to include more registers in the dataset. Next, we just would follow the same procedure explained in section 7.4.2 to train and validate our improved version of the ANN-based forwarding algorithm. Finally, a performance evaluation would show the outcome with respect the current version of our ANN-based multimetric routing protocol over different kinds of cities.

Table 7.8 concludes this chapter with the main features of our two proposals MPML and MPANN machine learning-based routing proposals for VANETs, presented in sections 7.6.1 and 7.6.2, respectively. Benefits and costs of implementing both ML-based routing protocols are presented. On the one hand, the ML-based decision tree models ML_6 and ML_{20}, are simple and require a low computational cost. They showed good results with the machine learning performance metrics, but they were not so good in our network simulations. Results improved once we included the trajectory (most decisive metric) as tiebreaker metric in the forwarding decisions. On the other hand, the ANN-based forwarding model has a more complex architecture and demands a higher computational cost. Nevertheless, it achieves a very good performance for both machine learning performance metrics and network simulations. Besides, it has been able to adapt pretty well in different network scenarios of different city maps.

TABLE 7.8: Summary of the comparison between our two proposals MPML and MPANN machine learning-based routing proposals for VANETs.

Proposal	**Features**	**Benefits**	**Cost and issues**
MPML (Multi-metric predictive machine learning)	- Routing mechanism based on decision tree (DT) rules. - Goal: predict which is the best forwarding node among the candidate nodes (neighbouring vehicles of the node currently holding the packet) for each packet towards its destination. - It includes a tie-breaker metric to improve the forwarding decision among nodes.	- DTs are simple to understand and to interpret, and they can easily be visualised (see Appendix A.2). - The cost of using the tree (i.e., predicting data) takes less run-time simulation. MPML is built up with a set *if-else-then* comparisons, which do not require a significant load on the CPU (see Fig. 7.21). - High prediction power was obtained with machine learning performance metrics, (i.e., accuracy and ROC curve) (see sec. 7.4.1) - No additional control traffic overhead is needed by the routing mechanism (see section 7.7.1.4).	- The network performance does not reflect good results when it was evaluated through network simulations (see sec. 7.6.1.1). - The most important metric (according to a L1 regularization analysis) was included to improve the forwarding decisions.(see Fig. 7.12). Therefore, a tie-breaker metric was included in the forwarding routing decisions, which notably improved the performance (see. sec. 7.6.1).
MPANN (Multi-metric predictive artificial neural network)	- Routing mechanism based on deep learning methods. - Goal: predict which is the best forwarding node among the candidate nodes (neighbouring vehicles of the node currently holding the packet) for each packet towards its destination.	- The network performance reflects good results for both validation with machine learning performance metrics (see sec. 7.4.2.3) and network simulations (see section 7.7). - The network performance shows good results to new unseen data from different city maps (see Fig. 7.25). - Additional overhead control traffic is not added to the routing mechanism, it uses the the same hello message structure as in 3MRP (see section 7.7.1.4).	- ANNs are more complex to model, and several training runs were performed to obtain the final model (see section. 7.4.2.1). - The cost of using ANN (i.e., predicting data) takes more running simulation time (see Fig. 7.21). MPANN has a complex architecture, and it is composed of different layers, and each layer contains a set of neurons (see Fig. 7.9). Also, the computational cost is related to the number of neighbouring nodes that each forwarding vehicle needs to evaluate (see Fig. 7.22). Thus, this time cost increases slightly with high vehicle densities.

Chapter 8

Conclusions and future work

8.1 Conclusions

The main objective of this thesis was devoted to making contributions to design routing protocols for vehicular ad hoc networks (VANETs) in urban environments. The general scenarios that we have considered to carry out all the performance evaluations of our proposals are urban environments with realistic traffic patterns and real maps. In these urban scenarios, VANET routing protocols are responsible for forwarding the information generated by vehicles to the city monitoring center through multi-hop vehicle communications.

We started our research work by analyzing some important aspects of the forwarding routing mechanisms based on the nodes' evaluation using a multimetric score. To enhance the nodes' classification to be chosen as next forwarding node, we have made some adaptation to the prediction of the current nodes' positions, improving routing protocol's effectiveness in terms of lower packet losses. The corresponding results have been presented in chapter 4.

One of our main contributions to VANET geographic routing protocols is the modeling of metrics. Using the probabilistic distribution of the considered metrics, we have designed a probabilistic-based multi-metric routing protocol for VANETs to optimize the process of selecting candidate nodes to forward information. Using our proposal we have obtained good performance in terms of lower packet losses also a lower end-to-end packet delay, according to the results presented in chapter 5.

Besides, we have also developed machine learning models to improve our multi-metric routing protocols for VANETs. For this purpose, we have collected a dataset with traffic metrics values by performing an extensive simulation campaign process.

Then, we have generated prediction models based on machine learning and deep learning algorithms to be applied in the forwarding decision process of the VANET routing protocols. Results show notably improvements in the network performance in terms of lower average packet losses and computational cost, as chapter 7 shows.

Below, we list the main contributions made in the research work during the development of this thesis:

- Firstly, we studied the state of the art regarding geographical routing protocols in VANETs. These kind of routing protocols have shown to be suitable for VANETs since they take forwarding decision following a hop-by-hop scheme. Also, these protocols can adapt more easily to dynamic topologies and cope with the potentially low link lifetimes inherent in VANETs. To this end, they use local information and low signaling overhead. We took the proposal named multimedia multimetric map-aware routing protocol (3MRP) [4], previously developed by the tutors of this thesis [15], as a starting point to develop or proposals. In this sense, we have proposed two contributions to improve 3MRP. (i) On the one hand, a weighted power mean function (W-PMF) to improve the selection of the best forwarding candidate which was presented in [39]. (ii) On the other hand, an improvement to the 3MRP multimetric protocol (3MRP+) for VANETs which was proposed in [40]:

  (i) **Weighted power mean function (W-PMF)**. Different weighting strategies for the 3MRP metrics were analyzed. We have obtained a multi-metric score value to arrange the candidate nodes. Our analysis has shown that the best way to combine the diverse metrics we have considered, is to use the geometric mean function. With this strategy, we have obtained the lowest average packet losses by maintaining a low delay. We have presented analysis and results in chapter 4.

  (ii) **Improved multimedia multimetric map-aware routing protocol (3MRP+)**. This proposal selects more accurately the forwarding nodes, estimating the current position at the moment of forwarding the message. Therefore, our proposal is better adapted to network conditions, especially when the speed of the nodes increases. As a result, the percentage of packets successfully delivered at destination from nodes with high speed, increases. Furthermore, we have not generated an overload, since we have no modified the frequency of the beacons nor introduced new fields in the beacon header. We have presented analysis and results in chapter 4.

- Secondly, we have studied the most important approaches that include strategies to forward packets following a hop-by-hop scheme. Most of the current proposals of VANET outing protocols are based on a hop-by-hop forwarding operation, since it has shown to be a suitable strategy for VANETs. Those strategies usually evaluate one or several routing metrics to take the forwarding decision, i.e. to choose the next hop (among the vehicles in the neighbourhood of the node currently holding the packet) to forward the packet. In this sense, we claim that the packet receptions according to a hop-by-hop scheme represent a sequence of events, i.e., the event of successfully deliver or not a packet to the next forwarding node. Those events can be modeled with a probability distribution, like the one that we have presented in chapter .

  To attain our goal, we have performed a large number of simulations under different representative scenarios to collect a new dataset to develop our model. With this collected information, we have modeled a probability distribution to successfully deliver the packet based on three metrics: (i) available bandwidth, (ii) vehicles' density, and (iii) distance to destination. Thus, by the offline dataset analysis, a statistical model for each metric was obtained. Then, we used the metrics models to design a new forwarding algorithm included in our proposal named probability-based multi-metric routing protocol (ProMRP), presented in chapter 5. ProMRP selects as the next forwarding node the best neighbour that ensures the delivery of the packet with the highest probability, while keeping low the average packet delay. Further, we have also included an accurate estimation of the current position of the nodes, obtaining an improved version of the protocol called enhanced ProMRP (EProMRP). This new version improves the network performance showing lower losses and similar average end-to-end packet delays. Moreover, our algorithm could be adapted and implemented in any VANET routing protocol that uses a multi-metric algorithm to select the best next-hop node to forward information.

- In the last part of this Ph.D. dissertation, we have studied and analyzed how to design a novel forwarding decision scheme in VANETs based on learning models. This way, we use prior knowledge of VANET performance to dynamically adapt the routing operation. As starting point, a new dataset of collected results was obtained to assess the probability of successful packet delivery focusing on five metrics: (i) available bandwidth, (ii) vehicles' density, (iii) distance to destination, (iv) nodes' trajectory, and (v) packet losses in the MAC layer. We have used the dataset to obtain three machine-learning based models that work with high precision. Then, we have included them to two novel learning-based routing protocols: (i) a multimetric predictive ML-based routing protocol (MPML), and (ii) a multimetric predictive ANN-based routing protocol (MPANN). Both proposals have been resented in chapter 7.

  Different VANET scenarios with different network sizes and different city maps have been considered to evaluate each proposal. Besides, the learning-based routing algorithms have been evaluated in two different scenario conditions. On the one hand, the network performance of each model has been evaluated in the same data collection scenario. On the other hand, the learning model that obtained the best performance has been evaluated in the second testing phase, considering a scenario with different city maps. The objective here is to measure the ability of our model to adapt to new network conditions.e have presented our contributions about machine-learning based multimetric routing protocols for VANETs in urban scenarios:

  (i) We have developed two prediction models based on decision trees that have been presented in chapter 7. We named the models as ML_6 and ML_{20}, with a dimension or tree depth of 6 and 20, respectively. The higher the decision tree depth, the better the prediction power, but at the cost of a higher computation complexity. After doing several tests, we got an optimal depth of 20 above which the accuracy converged, showing a good trade-off between complexity and good results in terms of fewer losses.

  (ii) Furthermore, we have also analysed the influence of each metric to break the tie (in the output to score the neighbour nodes candidates to be chosen as the next forwarding hop) in case of draw in the decision of both algorithms ML_6 and ML_{20}. This way, we have seen that the use of the trajectory metric to break the tie can further improve the decision forwarding in the routing protocol. Numerical results in chapter 7 have shown that this scheme performs better, specially under high vehicles' densities.

  Our proposal MPML routing protocol, which includes the proposed ML based prediction model using the trajectory as the tie-break metric, has shown the lowest percentage of packet losses (around 25%) for all vehicles' densities. However, the end-to-end average packet delay shows a slightly increase in the delay. The reason is that the percentage of packet losses was higher for the other ML algorithms considered, thus a lower amount of packets arrived to destination. Those packets usually traveled lower distances so their packet delay is lower. Anyway, our proposal MPANN based on an artificial neural network has shown even a better performance.

  (iii) After an extensive performance evaluation using a representative Barcelona city map, we have concluded that our best proposal is the one based on neuronal networks, named multimetric predictive artificial neural network-based (MPANN) routing protocol. Moreover, we have also conducted another set of simulations to evaluate the performance of MPANN in other city maps: (i) A similar area in Berlin city and (ii) a more complex and bigger area (narrow and wide streets) in Rome city. We have also shown the outcome of MPANN in those scenarios compared to 3MRP as a reference, since this multimetric routing protocol was the starting point of our proposals. MPANN performs better than 3MRP in terms of average percentage of packet losses and average end-to-end packet delay.

  ∗ In the Berlin city map (area with a semi-Manhattan road structure) our proposal MPANN shows a 2-35% improvement in the percentage of packet losses and a 2-40% improvement in average end-to-end delay, compared to 3MRP.
  ∗ In a more complex area of Rome city, MPANN shows a 10-35% improvement in the percentage of packet losses and up to 50% improvement in average end-to-end delay, compared to 3MRP.

(iv) Finally, we have also evaluated other evaluation metrics to assess the trade-off between benefits and cost for of each proposal of VANET routing protocol. We have computed (a) the computational cost and (b) the incurred overhead.

- The neural network prediction algorithm presents the highest computational cost. Certainly, the number of operations done in the algorithms of the learning models consume quite CPU time. The number of operations grows with the number of nodes involved in the forwarding decision, so the computational cost increases with the vehicles' density. Anyway, computational cost varies in a range of microseconds with the ordinary computer we used, which is not a significant delay.
- Finally, in regard to the overhead cost, it is not increased in any of the ML algorithms proposed with respect to the reference 3MRP. The reason is that our routing algorithms based on ML prediction use the same information as the multimetric routing protocol 3MRP, being not necessary to add any other new field in the beacon headers.

In general, we think that our contributions presented during this research work may have an impact on the vehicular networks area and may help the work of future researchers. Some of the results obtained have been reviewed by experts in the area and have been published in articles of international prestige (see next section). On the other hand, the improvements achieved when applying prediction models for multi-hop routing of information in vehicular networks, reflect the efficiency of its propagation more adapted to the inherent conditions of the environment. In addition to this, having implemented decision mechanisms based on machine learning and deep learning models, we have taken vehicular routing under a smarter approach supported by big data. In summary, including ML-based models in the communication mechanisms between vehicles, at the end favors improving mobility in cities, thus reducing pollution levels and improving citizens' quality of life.

8.2 List of publications generated during this thesis

In this section we list the publications result of the research done during the thesis. Most of the contents of this dissertation have been published in the following journals and conferences:

JCR Journal Publications

- **Leticia Lemus Cárdenas**, Ahmad Mohamad Mezher, Pablo Andrés Barbecho Bautista, Mónica Aguilar Igartua, "A Probability-Based Multimetric Routing Protocol for Vehicular Ad Hoc Networks in Urban Scenarios", *IEEE ACCESS*, ISSN: 2169-3536, Vol.7, Iss.1, pp.178020-178032, 2019. **IF(2018): 4.098; JCR: Q1 (Telecommunications category)** DOI: 10.1109/ACCESS.2019.2958743. [54]

- Pablo Barbecho Bautista, **Leticia Lemus Cárdenas**, Luis Urquiza Aguiar, Mónica Aguilar Igartua, "A Traffic-Aware Electric Vehicle Charging Management System for Smart Cities", *Elsevier Vehicular Communications*, ISSN: 2214-2096, Vol. 20, 2019. **IF(2018): 3.530; JCR: Q2 (Telecommunications category)** DOI: 10.1016/j.vehcom.2019.100188. [113]

- **Leticia Lemus Cárdenas**, Ahmad Mohamad Mezher, Pablo Barbecho Bautista, Mónica Aguilar Igartua, "Multimetric predictive routing protocols for VANETs in urban scenarios", *In preparation.*

Conferences

- Ahmad Mohamad Mezher, **Leticia Lemus Cárdenas**, Julián Cárdenas-Barrera, Eduardo Castillo Guerra, Julian Meng, Mónica Aguilar Igartua, "Improved Selection of the Best Forwarding Candidate in 3MRP for VANETs", *IEEE Symposium on Computers and Communications (ISCC 2019)*, DOI: 10.1109/ISCC47284.2019.8969580, Barcelona, Spain. pp.1-6 2019. [39].

- **Leticia Lemus Cárdenas**, Ahmad Mohamad Mezher, Nely Patricia López Márquez, Pablo Barbecho Bautista, Julian Cárdenas-Barrera, Mónica Aguilar Igartua, "3MRP+: An improved multimetric geographical routing protocol for VANETs", *15th ACM PE-WASUN*, DOI: 10.1145/3243046.3243056, Montreal, Canada.pp. 33–39, 2018. [40].

- Cristhian Iza Paredes, José Antonio Uribe Ramírez, Nely P. López Márquez, **Leticia Lemus Cárdenas**, Ahmad M. Mezher, Mónica Aguilar Igartua, "Multimedia communications in vehicular ad hoc networks for several applications in the smart cities", XIII Jornadas de Ingeniería Telemática (JITEL 2017),DOI: 10.4995/JITEL2017.2017.6584, Valencia, Spain. pp.212–215, 2017. [114]

8.3 Future work

During the development of this Ph.D. thesis, several issues have arisen which we think deserve further research in the future. Some potential research directions and improvements are summarized below:

- Design of a dynamic beacon frequency based on some representative network parameters. In an urban environment, vehicles can move through a wide range of kind of roads, e.g. narrow streets or main avenues in the city. Such conditions may not require the same frequency to report hello messages, which could lead to a more efficient use of the network resources. Thus, it could be interesting to have a smart mechanism to make the beacon frequency vary dynamically according to the current network conditions. Our contribution to this goal would be to use our machine-learning models to apply derived knowledge to assess optimal beacon frequency values.

- Design of a new multimetric scoring model to arrange neighbouring nodes. This model will be derived from the analysis of the correlation between the several metrics and the results on the probability to successfully delivering packets to destination.
- Train and obtain an ML model from the collected data set considering which are the most decisive metrics according to generalization method analysis. With those results, we will design a novel efficient algorithm for VANETs.
- Obtain a global dataset that gathers results of all representative possible scenarios. Our goal is to generalize as much as possible the collected data set to generate a new deep learning-based prediction model. To accomplish our goal, we will need to do these tasks:
  - Classify the different and most common types of cities in the world, regarding size, vehicles' densities, shape of roads and streets and other city infrastructure.
  - For each type of city, we will generate a representative map. Then, we will adapt them to be compatible with the simulator.
  - Generate a large amount of simulations to collect new values to feed our dataset.
  - Train machine learning models to generate a new prediction algorithm to be used in our machine-learning routing protocols for VANETs.
  - Finally, evaluate the new routing ML algorithms under different VANET scenarios, and report the evaluation results.
- To facilitate the above process, we plan to generate a useful tool available to researchers who have to face the same type of tasks. This tool will automate in a user friendly framework each of the tasks using scripts, methods and mechanisms that we have used through the research work of this thesis.

Appendix A

Machine learning (ML) and artificial neural network (ANN) models designed in this thesis

In this appendix, various code fragments are provided. In particular, we present the Python code that we have used to generate the different machine learning models. The choice of Python as a programming environment reflects how simple and understandable it is to generate a learning model from scratch given a meaningful dataset.

A.1 ML-based forwarding model using decision tree

Figure A.1 shows the Python code used to generate the ML-based decision tree that we have designed (see section 7.4.1). For this purpose, we have used the scikit-learn project [115], which is a free software machine learning library for the Python programming language [116, 117]. As it can be observed, the imported dataset does not receive special processing, since all features and labels are already normalized (see Fig. 7.2). Therefore, we have only defined the columns of the dataset, which correspond to the inputs, and the binary output. Besides, we have selected the percentage of data that will be used for training and testing, respectively. Then, the gini criterion has been used to build the decision tree in conjunction with 20-fold cross-validation (see table 7.2). Finally, Figure A.2 shows the generated decision tree, while Figure A.3 shows one of the branches of this tree.

```
#Import the needed python modules
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn import tree
from sklearn.model_selection import cross_val_score

# Import the dataset into a pandas dataframe
df = pd.read_csv('./BD_5METRICAS.csv', header=None)

# A name is given to each column, where each one represets the features and the labeled class
df.columns = ['bw', 'den','traj', 'mloss','dist','pred']
# Selection of the features
df_sf = df.loc[:,['bw', 'den','traj', 'mloss','dist','pred']]

#Data is divided into a training dataset and testing dataset
X, y = df_sf.iloc[:, 0:5].values, df_sf.iloc[:, 5].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0, stratify=y)

# Cross-validation
clf = tree.DecisionTreeClassifier(max_depth=6, criterion = 'gini')
# Perform 20-fold cross validation
scores = cross_val_score(estimator=clf, X=X_train, y=y_train, cv=20, n_jobs=4)
print scores.mean() #print the obtained result

#Training the decision tree model
clf = clf.fit(X_train, y_train)
print('Training accuracy: ', clf.score(X_train,y_train))
print('Testing accuracy: ', clf.score(X_test,y_test))

#Save the model
from sklearn.externals import joblib
joblib.dump(clf, 'DT_6.pkl')
```

FIGURE A.1: Python code to generate our ML-based forwarding model using decision tree (see section 7.4.1).

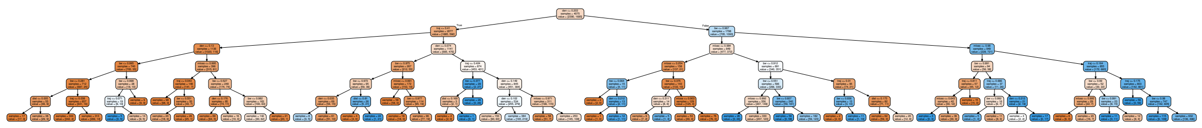


FIGURE A.2: ML-based decision tree (see section 7.4.1).

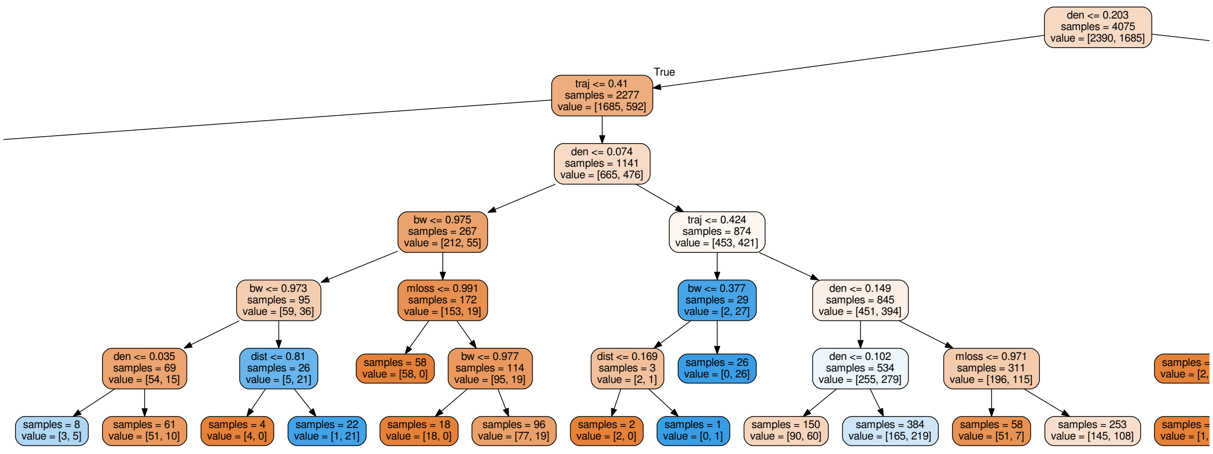


FIGURE A.3: Detail of our ML-based decision tree (see section 7.4.1).

A.2 ML-based forwarding model using bagged decision tree

On the other hand, Figure A.4 shows the ML-based bagged decision tree that we have designed (see section 7.4.1). As it can be seen, a similar methodology is used to train this ensemble tree, the only difference is that we have to provide the characteristics for every single tree. In this case, the bagged tree contain a combination of multiple single trees where each one will have a tree depth of 20 levels. Figures A.5 and A.6 show, in a graphical way, the resulting ML-based bagged decision tree.

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold

# Import the dataset into a pandas dataframe
df = pd.read_csv('./BD_5METRICAS.csv', header=None)

# A name is given to each column, where each one represets the features and the labeled class
df.columns = ['bw', 'den','traj', 'mloss','dist','pred']
# Selection of the features
df_sf = df.loc[:,['bw', 'den','traj', 'mloss','dist','pred']]

#Data is divided into a training dataset and testing dataset
X, y = df_sf.iloc[:, 0:5].values, df_sf.iloc[:, 5].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0, stratify=y)

#Create the Bagged tree
cart = DecisionTreeClassifier(max_depth=20) # Characteristics for each single tree
model = BaggingClassifier(base_estimator = cart, random_state = 0) # Bagged tree

# Cross-validation, Perform 20-fold cross validation
kfold = KFold(n_splits = 20, random_state = 0)
results = cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())

#Training the bagged decision tree model
model = model.fit(X_train, y_train)
print('Training accuracy: ', model.score(X_train,y_train))
print('Testing accuracy: ', model.score(X_test,y_test))

#Save the model
from sklearn.externals import joblib
joblib.dump(model, 'BT_20.pkl')
```

FIGURE A.4: Python code to generate the ML-based forwarding model using bagged decision tree (see section 7.4.1).

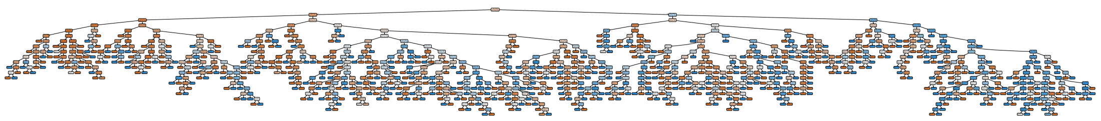


FIGURE A.5: ML-based bagged decision tree (see section 7.4.1).

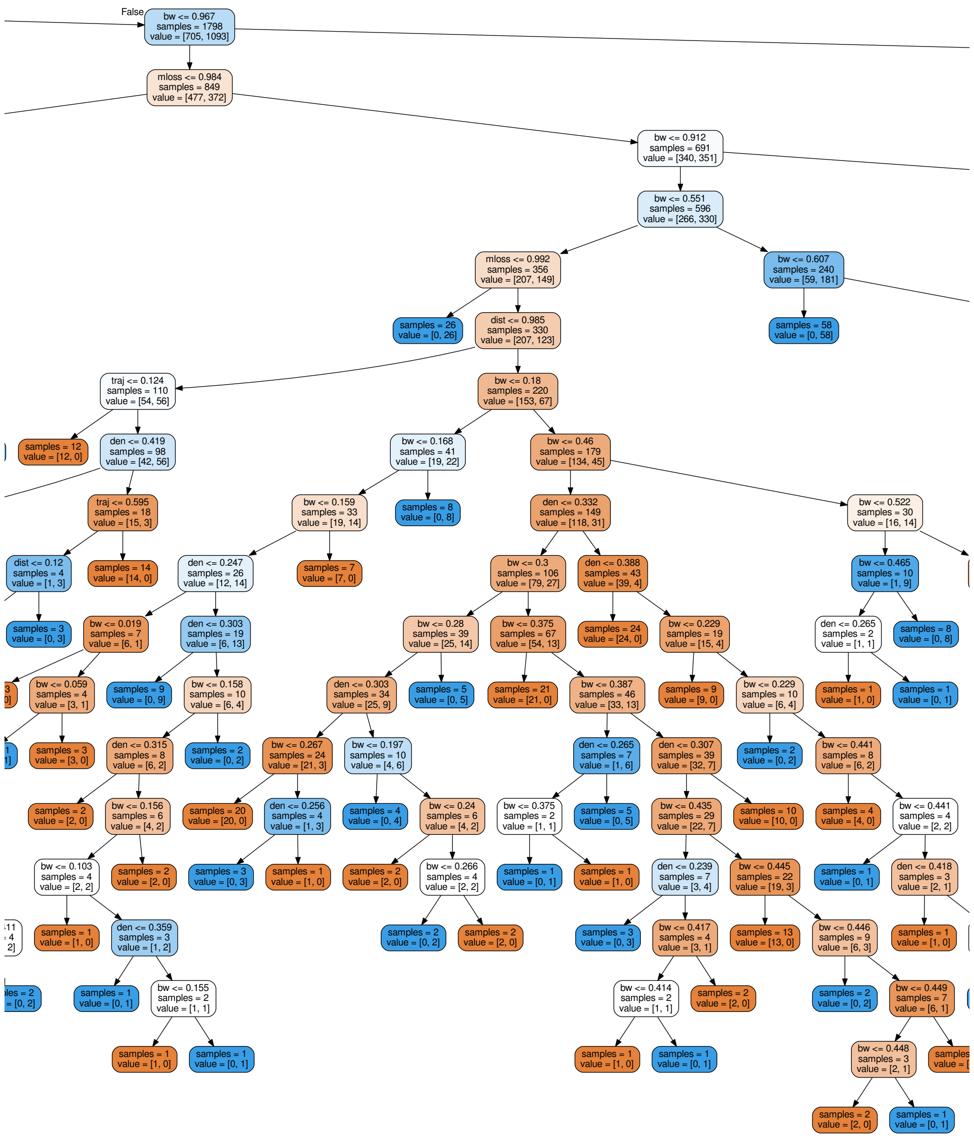
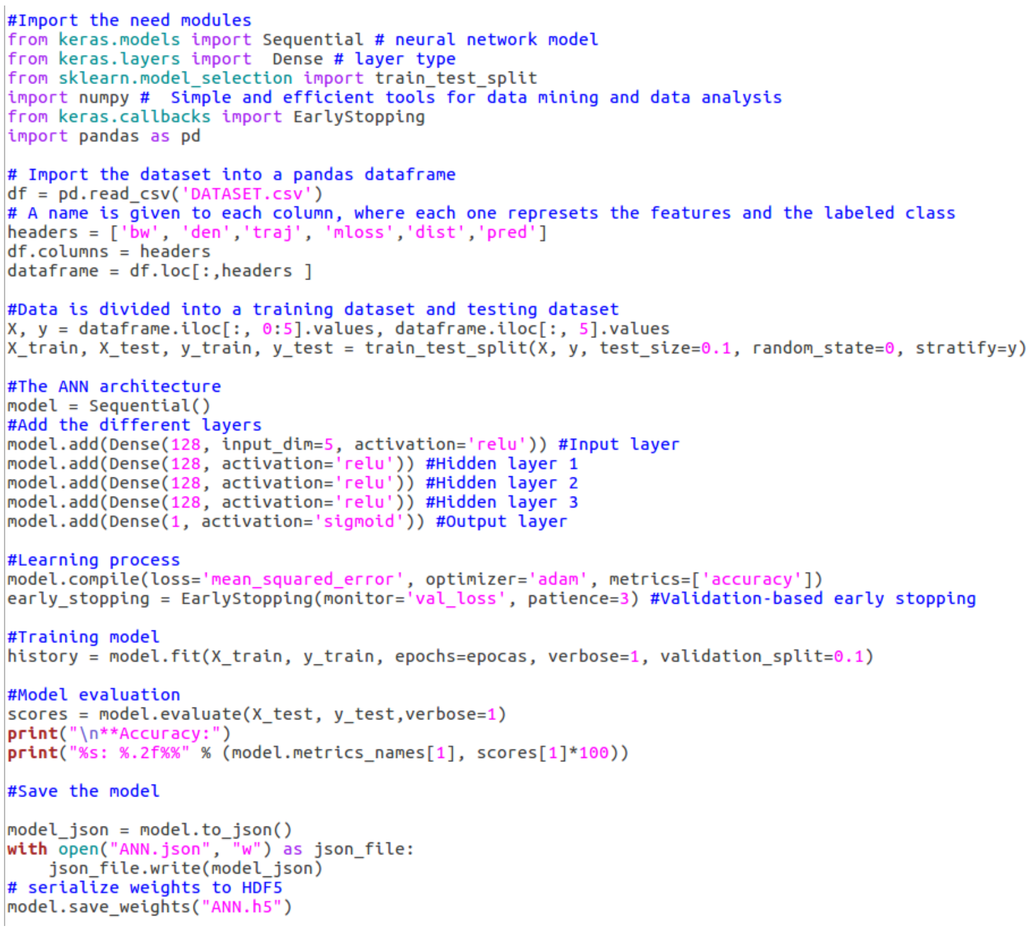


Figure A.6: Detail of our ML-based bagged decision tree (see section 7.4.1).

A.3 ANN-based forwarding model

Figure A.7 depicts the Python code used to generate the ANN-based forwarding model that we have designed (see section 7.4.2). This code was generated using the keras tool [111]. The procedure to import the dataset and its division into training and testing datasets is similar to the previous ML mechanisms. However, its construction differs in some aspects because it has a more complex architecture. First, the neural network architecture is built in terms of the number of layers, number of neurons, and the interconnection between the different layers. Second, in the learning process, a suitable optimization algorithm has to be selected in conjunction with the loss error function. Finally, in the training phase, we have selected the number of epochs, and the validation-based early stop technique to avoid overfitting.



```
#Import the need modules
from keras.models import Sequential # neural network model
from keras.layers import  Dense # layer type
from sklearn.model_selection import train_test_split
import numpy #  Simple and efficient tools for data mining and data analysis
from keras.callbacks import EarlyStopping
import pandas as pd

# Import the dataset into a pandas dataframe
df = pd.read_csv('DATASET.csv')
# A name is given to each column, where each one represets the features and the labeled class
headers = ['bw', 'den','traj', 'mloss','dist','pred']
df.columns = headers
dataframe = df.loc[:,headers ]

#Data is divided into a training dataset and testing dataset
X, y = dataframe.iloc[:, 0:5].values, dataframe.iloc[:, 5].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=0, stratify=y)

#The ANN architecture
model = Sequential()
#Add the different layers
model.add(Dense(128, input_dim=5, activation='relu')) #Input layer
model.add(Dense(128, activation='relu')) #Hidden layer 1
model.add(Dense(128, activation='relu')) #Hidden layer 2
model.add(Dense(128, activation='relu')) #Hidden layer 3
model.add(Dense(1, activation='sigmoid')) #Output layer

#Learning process
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=3) #Validation-based early stopping

#Training model
history = model.fit(X_train, y_train, epochs=epocas, verbose=1, validation_split=0.1)

#Model evaluation
scores = model.evaluate(X_test, y_test,verbose=1)
print("\n**Accuracy:")
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

#Save the model

model_json = model.to_json()
with open("ANN.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("ANN.h5")
```

FIGURE A.7: Python code to generate the ANN-based forwarding model (see section 7.4.2).

Appendix B

Guide with useful tips for future Ph.D. candidates

In this appendix we present a collection of different tools that are typically used to carry out experiments in the computer network research field. Also, we have included useful tips for researchers to help their work and save a lot of time in their research work.

B.1 Tools and tips to automate processes in simulation and data management tasks

There Ph.D. candidates of the SISCOM (Smart Services for Information Systems and Communication Networks)[1] research group, that belong to the Department of Networking Engineering at the Universitat Politècnica de Catalunya (UPC) in Barcelona, have worked in an article that collects the most important aspects to begin with the simulation work about their research work. Our goal is to write a guide document that collects useful tips to save time and effort of future researchers. The resulting document is in preparation to be submitted in the following article:

- **Leticia Lemus Cárdenas**, Pablo Barbecho, Juan Astudillo, *"Tools and tips to automate processes in simulation and data management tasks,"*, *In preparation.*

A preliminary draft version of our *Guide with Tips for future Ph.D. candidates* has been uploaded in [118]. In our guide we describe the main tools that we have used during our Ph.D. Thesis: network simulators, programming languages, LaTeX editors, among others. In the following subsection, the contents of this article are pointed out.

[1] https://siscom.upc.edu/en

B.1.1 Contents

Network simulators are commonly used to test different experiments to evaluate the performance of novel proposals developed by researchers. Simulation tools are widely used due to the infeasibility of deploying physical testbeds when complex scenarios are difficult to be evaluated. Besides, network simulators are also needed when researchers are proposing new techniques that must be tested quickly during the design phase, or when they are in an early stage of research. In our case, we have used two well-known network simulators widely used by the research community: (i) OMNeT++/Veins/SUMO and (ii) ns-3 discrete-event network simulator. In particular, we present a novel methodology to make efficient and faster simulations in both network simulators.

On the other hand, among the different programming languages, Python is widely used in different areas such as Web and Internet Development, scientific and numeric computing, education, desktop GUIs, business, and so on. Nowadays, it is one of the most prominent candidates to deploy different machine learning models. With this in mind, we present how to use Python to generate and validate a machine learning model from scratch. Besides, we present the Matlab toolbox for machine learning that is a good approach to start with in the machine learning research field. Also, we present how to calculate the computational cost and overhead of the most common algorithms. Finally, the most prevalent LaTex editors are presented.

The contents of the final version of the article will contain the following:

1. Introduction
2. Machine Learning. How to start with.
    - 2.1 Machine learning with Matlab
        - 2.1.1 Machine learning toolbox.
    - 2.2 Machine learning with Python.
        - 2.2.1 Building the machine learning models.
            - 2.2.1.1 Create machine learning models with skikit-learn project.
            - 2.2.1.2 Create deep learning learning model with keras.
        - 2.2.2 Export the learning model to the network simulators.
3. Trade-off between benefits and costs to evaluate our proposals
    - 3.1 Computational cost.
    - 3.2 Overhead
4. Offline and online LaTeX editors
    - 4.1 TeXstudio
        - 4.1.1 Installation
    - 4.2 Overleaf
5. How to make efficient and faster simulations
    - 5.1 OMNeT++

5.1.1 Prerequisite Packages.

5.1.2 SUMO

5.1.3 Simultaneous simulations.

5.2 ns-3

5.2.1 Prerequisite Packages.

5.2.2 Simultaneous simulations.

Bibliography

[1] João Almeida, Muhammad Alam, Joaquim Ferreira, and Arnaldo S.R. Oliveira. Mitigating adjacent channel interference in vehicular communication systems. *Digital Communications and Networks*, 2(2):57 – 64, 2016. ISSN 2352-8648. doi: https://doi.org/10.1016/j.dcan.2016.03.001. URL http://www.sciencedirect.com/science/article/pii/S2352864816300104.

[2] K. Flynn. V2x, Sep 2016. URL "https://www.3gpp.org/v2x".

[3] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 243–254, New York, NY, USA, 2000. ACM. ISBN 1-58113-197-6. doi: 10.1145/345910.345953. URL http://doi.acm.org/10.1145/345910.345953.

[4] Ahmad Mohamad Mezher and Mónica Aguilar Igartua. Multimedia multimetric map-aware routing protocol to send video-reporting messages over VANETs in smart cities. *IEEE Transactions on Vehicular Technology*, 66(12):10611–10625, Dec 2017. ISSN 0018-9545. doi: 10.1109/TVT.2017.2715719.

[5] Coast Steve. OpenStreetMap, February 2018. URL http://www.openstreetmap.org.

[6] S. Raschka and V. Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow.* Packt Publishing Ltd, 2017. ISBN 978-1-78712-593-3.

[7] Edureka. Edureka's data science training, May 2020. URL www.edureka.in/data-science.

[8] Sebastian Raschka. *Python machine learning.* Packt Publishing Ltd, 2015.

[9] MediaWiki.org. Simulation of urban mobility (SUMO), January 2008. URL https://sumo.dlr.de.

[10] United Nations. By 2050, 68% of world population will live in urban areas - 2018 world urbanization prospect, May 2018. URL https://www.youtube.com/watch?v=XN92srq5jwg.

[11] Juniper Research. Smart cities – what's in it for citizens?, May 2017. URL https://newsroom.intel.com/wp-content/uploads/sites/11/2018/03/smart-cities-whats-in-it-for-citizens.pdf.

[12] N. R. Moloisane, R. Malekian, and D. Capeska Bogatinoska. Wireless machine-to-machine communication for intelligent transportation systems: Internet of vehicles and vehicle to grid. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 411–415, May 2017. doi: 10.23919/MIPRO.2017.7973459.

[13] Asim Rasheed, Saira Gillani, Sana Ajmal, and Amir Qayyum. Vehicular ad hoc network (VANET): A survey, challenges, and applications. In Anis Laouiti, Amir Qayyum, and Mohamad Naufal Mohamad Saad, editors, *Vehicular Ad-Hoc Networks for Smart Cities*, pages 39–51, Singapore, 2017. Springer Singapore. ISBN 978-981-10-3503-6.

[14] Industry Europe. How data can build the smart cities of the future, July 2019. URL https://industryeurope.com/how-data-can-build-the-smart-cities-of-the-future/.

[15] Ahmad Mohamad Mezher, 2016.

[16] V. D. V, A. Chekima, F. Wong, and J. A. Dargham. A study on vehicular ad hoc networks. In *2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, pages 422–426, Dec 2015. doi: 10.1109/AIMS.2015.73.

[17] E. C. Eze, S. Zhang, and E. Liu. Vehicular ad hoc networks (VANETs): Current state, challenges, potentials and way forward. In *2014 20th International Conference on Automation and Computing*, pages 176–181, Sept 2014. doi: 10.1109/IConAC.2014.6935482.

[18] H. Seo, K. Lee, S. Yasukawa, Y. Peng, and P. Sartori. LTE evolution for vehicle-to-everything services. *IEEE Communications Magazine*, 54(6):22–28, June 2016. ISSN 0163-6804. doi: 10.1109/MCOM.2016.7497762.

[19] 3GPP. Lte. URL https://www.3gpp.org/technologies/keywords-acronyms/98-lte.

[20] Ramona Trestian and Gabriel-Miro Muntean. *Paving the way for 5G through the convergence of wireless systems*. IGI Global, 2018.

[21] Andreas Festag. Standards for vehicular communication—from ieee 802.11 p to 5g. *e & i Elektrotechnik und Informationstechnik*, 132(7):409–416, 2015.

[22] 3GPP TS 36.300 v14. 4.0. Evolved universal terrestrial radio access (e-utra) and evolved universal terrestrial radio access network (e-utran); overall description; stage 2 (release 14). 2017.

[23] Ruqayah Al-ani, Bo Zhou, Qi Shi, and Ali Sagheer. A survey on secure safety applications in VANET. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1485–1490. IEEE, 2018.

[24] Anna Maria Vegni, Mauro Biagi, Roberto Cusani, et al. Smart vehicles, technologies and main applications in vehicular ad hoc networks. *Vehicular technologies-deployment and applications*, pages 3–20, 2013.

[25] Asim Rasheed, Saira Gillani, Sana Ajmal, and Amir Qayyum. Vehicular ad hoc network (VANET): A survey, challenges, and applications. In *Vehicular Ad-Hoc Networks for Smart Cities*, pages 39–51. Springer, 2017.

[26] F. Dressler, C. Sommer, D. Eckhoff, and O. K. Tonguz. Toward realistic simulation of intervehicle communication. *IEEE Vehicular Technology Magazine*, 6(3):43–51, Sep. 2011. ISSN 1556-6080. doi: 10.1109/MVT.2011.941898.

[27] Christoph Sommer. Veins. the open source vehicular network simulator framework, June 2018. URL `https://veins.car2x.org`.

[28] German Aerospace Center. OMNeT++ discrete event simulator. June 2017. URL `http://www.dlr.de`.

[29] András Varga and Rudolf Hornig. An overview of the OMNeT++ simulation environment. In *Proceedings of the 1st international conference on simulation tools and techniques for communications, networks and systems and workshops*, page 60. ICST, 2008.

[30] Christoph Sommer, Jérôme Härri, Fatma Hrizi, Björn Schünemann, and Falko Dressler. Simulation tools and techniques for vehicular communications and applications. In *Vehicular ad hoc Networks*, pages 365–392. Springer, 2015.

[31] I. Stojmenovic. Position-based routing in ad hoc networks. *IEEE Communications Magazine*, 40(7):128–134, July 2002. ISSN 0163-6804. doi: 10.1109/MCOM.2002.1018018.

[32] Degui Xiao, Lixiang Peng, Clement Ogugua Asogwa, and Lei Huang. An improved GPSR routing protocol. *Int. J. Adv. Comput. Technol*, 3(5):132–139, 2011.

[33] S. Dahmane and P. Lorenz. Weighted probabilistic next-hop forwarder decision-making in VANET environments. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2016. doi: 10.1109/GLOCOM.2016.7842381.

[34] H. Tu, L. Peng, H. Li, and F. Liu. GPSR-MV: A routing protocol based on motion vector for VANET. In *2014 12th International Conference on Signal Processing (ICSP)*, pages 2354–2359, Oct 2014. doi: 10.1109/ICOSP.2014.7015415.

[35] X. Yang, M. Li, Z. Qian, and T. Di. Improvement of GPSR protocol in vehicular ad hoc network. *IEEE Access*, 6:39515–39524, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2853112.

[36] Ahmad Mohamad Mezher, Juan J. Oltra, Luis Urquiza-Aguiar, Cristhian Iza-Parades, Carolina Tripp-Barba, and Mónica Aguilar Igartua. Realistic environment for VANET simulations to detect the presence of obstacles in vehicular ad hoc networks. In *PE-WASUN'14*, volume 47, pages 77–84, November 2014.

[37] Cheikh Sarr, Claude Chaudet, Guillaume Chelius, and Isabelle Guérin Lassous. A node-based available bandwidth evaluation in ieee 802.11 ad hoc networks. *International Journal of Parallel, Emergent and Distributed Systems*, 21(6):423–440, 2006. doi: 10.1080/17445760600761403. URL `https://doi.org/10.1080/17445760600761403`.

[38] Cristiano Rezende, Azzedine Boukerche, Heitor S Ramos, and Antonio AF Loureiro. A reactive and scalable unicast solution for video streaming over VANETs. *IEEE Transactions on Computers*, 64(3):614–626, 2014.

[39] Ahmad Mohamad Mezher, Leticia Lemus Cárdenas, J. C. Barrera, E. C. Guerra, J. Meng, and Mónica Aguilar Igartua. Improved selection of the best forwarding candidate in 3MRP for VANETs. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, June 2019. doi: 10.1109/ISCC47284.2019.8969580.

[40] Leticia Lemus Cárdenas, Ahmad Mohamad Mezher, Nely Patricia López Márquez, Pablo Barbecho Bautista, Julián Cárdenas-Barrera, and Mónica Aguilar Igartua. 3MRP+: An improved multimetric geographical routing protocol for VANETs. In *Proceedings of the 15th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks*, PE-WASUN'18, pages 33–39, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5961-0. doi: 10.1145/3243046.3243056. URL http://doi.acm.org.recursos.biblioteca.upc.edu/10.1145/3243046.3243056.

[41] A. Ashtaiwi and A. Saoud. Performance comparison of position based routing protocols for VANETs. In *2017 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, pages 78–84, Oct 2017. doi: 10.1109/IINTEC.2017.8325917.

[42] R. K. Jaiswal and J. C. D. Edagf: Estimation amp;amp; direction aware greedy forwarding for urban scenario in vehicular ad-hoc network. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 814–821, Aug 2015. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.160.

[43] Ahmed Nazar Hassan, Abdul Hanan Abdullah, Omprakash Kaiwartya, Yue Cao, and Dalya Khalid Sheet. Multi-metric geographic routing for vehicular ad hoc networks. *Wireless Networks*, Apr 2017. ISSN 1572-8196. doi: 10.1007/s11276-017-1502-5. URL https://doi.org/10.1007/s11276-017-1502-5.

[44] K. N. Qureshi, F. Bashir, and A. H. Abdullah. Real time traffic density aware road based forwarding method for vehicular ad hoc networks. In *2017 10th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 1–6, Sept 2017. doi: 10.1109/WMNC.2017.8248850.

[45] Z. S. Houssaini, I. Zaimi, M. Oumsis, and S. El Alaoui Ouatik. Improvement of gpsr protocol by using future position estimation of participating nodes in vehicular ad-hoc networks. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 87–94, Oct 2016. doi: 10.1109/WINCOM.2016.7777196.

[46] N.V. Dharani Kumari and B.S. Shylaja. AMGRP: AHP-based multimetric geographical routing protocol for urban environment of VANETs. *Journal of King Saud University - Computer and Information Sciences*, 31(1):72 – 81, 2019. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2017.01.001. URL http://www.sciencedirect.com/science/article/pii/S1319157817300095.

[47] Peter S Bullen. *Handbook of means and their inequalities*, volume 560. Springer Science & Business Media, 2013.

[48] Ahmad Mohamad Mezher, Mónica Aguilar Igartua, Luis J. de la Cruz Llopis, Esteve Pallarès Segarra, Carolina Tripp-Barba, Luis Urquiza-Aguiar, Jordi Forné, and Emilio Sanvicente Gargallo. A multi-user game-theoretical multipath routing protocol to send video-warning messages over mobile ad hoc networks. *Sensors*, 15(4):9039, 2015. ISSN 1424-8220. doi: 10.3390/s150409039. URL http://www.mdpi.com/1424-8220/15/4/9039.

[49] András Varga. OMNeT++ discrete event simulator, June 2018. URL https://www.omnetp.org.

[50] Shylaja Banagiri Srikantaiah Dharani Kumari Nooji Venkatramana and Jayalakshmi Moodabidri. Cisrp: connectivity-aware intersection-based shortest path routing protocol for VANETs in urban environments. *IET Networks*, 7:152–161(9), May 2018. ISSN 2047-4954. URL http://digital-library.theiet.org/content/journals/10.1049/iet-net.2017.0012.

[51] Carolina Tripp-Barba, Luis Urquiza Aguiar, Mónica Aguilar-Igartua, David Rebollo-Monedero, Luis de la Cruz Llopis, Ahmad Mohamad-Mezher, and José Aguilar. A multimetric, map-aware routing protocol for VANETs in urban areas. 14:2199–224, 02 2014.

[52] German Aerospace Center. OMNeT++ discrete event simulator, June 2017. URL http://www.dlr.de.

[53] Christian Iza. Data dissemination in vehicular ad hoc networks, May 2017. URL https://github.com/7d5791/flooding.

[54] Leticia Lemus Cárdenas, Ahmad Mohamad Mezher, Pablo Andrés Bautista, and Mónica Aguilar Igartua. A probability-based multimetric routing protocol for vehicular ad hoc networks in urban scenarios. *IEEE Access*, 7:178020–178032, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2958743.

[55] A. Silva, N. Reza, and A. Oliveira. Improvement and performance evaluation of gpsr-based routing techniques for vehicular ad hoc networks. *IEEE Access*, 7:21722–21733, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2898776.

[56] N.V. Dharani Kumari and B.S. Shylaja. AMGRP: AHP-based multimetric geographical routing protocol for urban environment of VANETs. *Journal of King Saud University - Computer and Information Sciences*, 31(1):72 – 81, 2019. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2017.01.001. URL http://www.sciencedirect.com/science/article/pii/S1319157817300095.

[57] C. Wu, T. Yoshinaga, Y. Ji, T. Murase, and Y. Zhang. A reinforcement learning-based data storage scheme for vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 66(7):6336–6348, July 2017. ISSN 0018-9545. doi: 10.1109/TVT.2016.2643665.

[58] N. Li, J. Martínez-Ortega, V. H. Díaz, and J. A. S. Fernández. Probability prediction-based reliable and efficient opportunistic routing algorithm for VANETs, 2018, 26. *IEEE/ACM Transactions on Networking*, (4):1933–1947. ISSN 1063-6692. doi: 10.1109/TNET.2018.2852220.

[59] X. Zeng, M. Yu, and D. Wang. A new probabilistic multi-hop broadcast protocol for vehicular networks. volume 67, pages 12165–12176, Dec 2018. doi: 10.1109/TVT.2018.2872998.

[60] Saeed Shokrollahi Ramin Karimi. Pgrp: Predictive geographic routing protocol for VANETs. *Computer Networks*, 141:67–81, 2018. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2018.05.017.

[61] M. Ye, L. Guan, and M. Quddus. Mpbrp- mobility prediction based routing protocol in VANETs. In *2019 International Conference on Advanced Communication Technologies and Networking (CommNet)*, pages 1–7, April 2019. doi: 10.1109/COMMNET.2019.8742389.

[62] Zineb Squalli Houssaini, Imane Zaimi, Mohammed Oumsis, and Said El Alaoui Ouatik. Gpsr+predict: An enhancement for gpsr to make smart routing decision by anticipating movement of vehicles in VANETs. 2017.

[63] Saifullah Khan, Muhammad Alam, Martin Fränzle, Nils Müllner, and Yuanfang Chen. A traffic aware segment-based routing protocol for VANETs in urban scenarios. *Computers Electrical Engineering*, 68:447 – 462, 2018. ISSN 0045-7906. doi: https://doi.org/10.1016/j.compeleceng.2018.04.017. URL http://www.sciencedirect.com/science/article/pii/S0045790617337047.

[64] K. Krishnamoorthy. In *Handbook of statistical distributions with applications*. University of Louisiana at Lafayette U.S.A, 2016.

[65] C. Moler N. Little and S. Bangert. The mathworks, inc., February 2018. URL https://es.mathworks.com.

[66] Christoph Sommer and Falko Dressler. *Vehicular Networking*. Cambridge University Press, 2014. doi: 10.1017/CBO9781107110649.

[67] Carolina Tripp-Barba, Luis Urquiza-Aguiar, Mónica Aguilar Igartua, David Rebollo-Monedero, Luis J. De la Cruz Llopis, Ahmad Mohamad Mezher, and José Alfonso Aguilar-Calderón. A multimetric, map-aware routing protocol for VANETs in urban areas. *Sensors*, 14(2):2199–2224, 2014. ISSN 1424-8220. doi: 10.3390/s140202199. URL https://www.mdpi.com/1424-8220/14/2/2199.

[68] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, January 2011. doi: 10.1109/TMC.2010.133.

[69] Andras Varga. OMNeT++ discrete event simulator, 2019.

[70] B. Pattberg. Dlr - institute of transportation systems - eclipse SUMO – simulation of urban mobility, Jan 2008. URL "https://sumo.dlr.de".

[71] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16, 2018.

[72] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[73] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[74] Nicolaos Karayiannis and Anastasios N Venetsanopoulos. *Artificial neural networks: learning algorithms, performance evaluation, and applications*, volume 209. Springer Science & Business Media, 2013.

[75] Andreas C Müller, Sarah Guido, et al. *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[76] Harry Henderson. *Encyclopedia of computer science and technology*. Infobase Publishing, 2009.

[77] Rodrigo C Barros, André CPLF De Carvalho, Alex A Freitas, et al. *Automatic design of decision-tree induction algorithms*, volume 10. Springer, 2015.

[78] Jehad Ali, Rehanullah Khan, Nasir Ahmad, and Imran Maqsood. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, 9(5):272, 2012.

[79] Donald J Norris. Machine learning with the raspberry pi.

[80] Lior Rokach and Oded Z Maimon. *Data mining with decision trees: theory and applications*, volume 69. World scientific, 2008.

[81] Jason Bell. *Machine learning: hands-on for developers and technical professionals*. John Wiley & Sons, 2020.

[82] Sarkar Dipayan and Natarajan Vijayalakshmi. *Ensemble Machine Learning Cookbook: Over 35 practical recipies to explore ensemble machine learning techniques using Python*. Packt Publishing Ltd, 2019.

[83] Mohamad H Hassoun et al. *Fundamentals of artificial neural networks*. MIT press, 1995.

[84] SN Sivanandam and SN Deepa. *Introduction to neural networks using Matlab 6.0*. Tata McGraw-Hill Education, 2006.

[85] Giuseppe Ciaburro and Balaji Venkateswaran. *Neural Networks with R: Smart models using CNN, RNN, deep learning, and artificial intelligence principles*. Packt Publishing Ltd, 2017.

[86] Farshad Firouzi. *Intelligent Internet of Things: From Device to Fog and Cloud*. Springer, 2020.

[87] Hamed Habibi Aghdam and Elnaz Jahani Heravi. Guide to convolutional neural networks. *New York, NY: Springer. doi*, 10:978–3, 2017.

[88] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7, 2017.

[89] Igor Aizenberg. *Complex-valued neural networks with multi-valued neurons*, volume 353. Springer, 2011.

[90] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[91] James Loy. *Neural Network Projects with Python: The ultimate guide to using Python to explore the true power of neural networks through six projects*. Packt Publishing Ltd, 2019.

[92] Kenji Suzuki. *Artificial neural networks: methodological advances and biomedical applications*. BoD–Books on Demand, 2011.

[93] Kishan Maladkar. 6 types of neural network currently being used in machine learning, June 2018. URL `https://analyticsindiamag.com/6-types-of-artificial-neural-networks-currently-being-used-in-todays-technology`.

[94] Tonny J Oyana, Luke EK Achenie, Ernesto Cuadros-Vargas, Patrick A Rivers, and Kara E Scott. A mathematical improvement of the self-organizing map algorithm. In *Proceedings from the International Conference on Advances in Engineering and Technology*, pages 522–531. Elsevier, 2006.

[95] Rodolfo Bonnin. *Machine Learning for Developers: Uplift your regular applications with the power of statistics, analytics, and machine learning.* Packt Publishing Ltd, 2017.

[96] Anand J Kulkarni and Suresh Chandra Satapathy. Optimization in machine learning and applications.

[97] Ali Tizghadam, Hamzeh Khazaei, Mohammad Moghaddam, and Yasser Hassan. Machine learning in transportation. *Journal of Advanced Transportation*, 2019:1–3, 06 2019. doi: 10.1155/2019/4359785.

[98] Sangmin Lee, Younghoon Kim, Hyungu Kahng, Soon-Kyo Lee, Seokhyun Chung, Taesu Cheong, Keeyong Shin, Jeehyuk Park, and Seoung Bum Kim. Intelligent traffic control for autonomous vehicle systems based on machine learning. *Expert Systems with Applications*, 144:113074, 2020.

[99] Çağrı Kaymak and Ayşegül Uçar. *A Brief Survey and an Application of Semantic Image Segmentation for Autonomous Driving*, pages 161–200. Springer International Publishing, Cham, 2019. ISBN 978-3-030-11479-4. doi: 10.1007/978-3-030-11479-4_9. URL https://doi.org/10.1007/978-3-030-11479-4_9.

[100] Y. Tang, N. Cheng, W. Wu, M. Wang, Y. Dai, and X. Shen. Delay-minimization routing for heterogeneous VANETs with machine learning based mobility prediction. *IEEE Transactions on Vehicular Technology*, 68(4):3967–3979, April 2019. ISSN 0018-9545. doi: 10.1109/TVT.2019.2899627.

[101] C. Wu, T. Yoshinaga, Y. Ji, and Y. Zhang. Computational intelligence inspired data delivery for vehicle-to-roadside communications. *IEEE Transactions on Vehicular Technology*, 67(12):12038–12048, Dec 2018. ISSN 0018-9545. doi: 10.1109/TVT.2018.2871606.

[102] M. Laroui, A. Sellami, B. Nour, H. Moungla, H. Afifi, and S. B. Hacene. Driving path stability in VANETs. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2018. doi: 10.1109/GLOCOM.2018.8647450.

[103] Degan Zhang, Ting Zhang, and Xiaohuan Liu. Novel self-adaptive routing service algorithm for application in VANET. *Applied Intelligence*, 49(5):1866–1879, 2019.

[104] J. Wu, M. Fang, H. Li, and X. Li. RSU-assisted traffic-aware routing based on reinforcement learning for urban VANETs. *IEEE Access*, 8:5733–5748, 2020.

[105] D. E. O'Leary. Artificial intelligence and big data. *IEEE Intelligent Systems*, 28(2):96–99, March 2013. ISSN 1941-1294. doi: 10.1109/MIS.2013.39.

[106] The MathWorks, Inc. MATLAB. "https://es.mathworks.com/products/matlab.html?s_tid=hp_ff_p_matlab".

[107] Diederik P. Kingma and Jimmy Lei Ba. Adam. A method for stochastic optimization. *arXiv.org*, December 2014. doi: arXiv:1412.6980.

[108] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016.

[109] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[110] Garvesh Raskutti, Martin J Wainwright, and Bin Yu. Early stopping and non-parametric regression: an optimal data-dependent stopping rule. *The Journal of Machine Learning Research*, 15(1):335–366, 2014.

[111] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[112] Jacolien Lokhorst. The lasso and generalised linear models. 1999.

[113] Pablo Barbecho, Leticia Lemus, Luis Urquiza, and Mónica Aguilar Igartua. A traffic-aware electric vehicle charging management system for smart cities. *Vehicular Communication*, 20:100188:1–100188:14, Dec 2019. doi: 10.1016/j.vehcom.2019.100188. URL http://hdl.handle.net/2117/172770;https://www.sciencedirect.com/science/article/abs/pii/S2214209619302359.

[114] Christian Iza Paredes, José Antonio Uribe, Nely López Márquez, Leticia Lemus Cárdenas, Ahmad Mohamad Mezher, and Mónica Aguilar Igartua. Multimedia communications in vehicular adhoc networks for several applications in the smart cities. In *XIII Jornadas de Ingeniería Telemática (JITEL 2017)*, pages 212–215, September 2017. doi: DOI: 10.4995/JITEL2017.2017.6584.

[115] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[116] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[117] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[118] Leticia Lemus, Pablo Barbecho, and Juan Astudillo. Guide with tools and tips to automate processes in simulation and data management tasks, June 2020. URL https://drive.google.com/file/d/1EvVOLCHlIXpQiBoXZnSeUyQkInEyWrkX/view?usp=sharing.