UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LUÍS ANTÔNIO L. F. DA COSTA

Using Artificial Intelligence to Support Emerging Networks Management Approaches

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Edison Pignaton de Freitas
Coadvisor: Prof. Dr. Rafael Kunst

Porto Alegre
October 2020

CIP — CATALOGING-IN-PUBLICATION

L. F. da Costa, Luís Antônio

Using Artificial Intelligence to Support Emerging Networks Management Approaches / Luís Antônio L. F. da Costa. – Porto Alegre: PPGC da UFRGS, 2020.

74 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Advisor: Edison Pignaton de Freitas; Coadvisor: Rafael Kunst.

1. Routing Protocol. 2. Emergent Networks. 3. Artificial Intelligence. 4. Complexity Analysis. I. de Freitas, Edison Pignaton. II. Kunst, Rafael. III. Título.

"When you realize there is something you don't understand,
then you're generally on the right path to understanding all kinds of things."
— JOSTEIN GAARDER, *The Solitaire Mystery*

ACKNOLEDGMENTS

I would first like to thank my supervisors Edison Pignaton de Freitas and Rafael Kunst of the Institute of Informatics at the Federal University of Rio Grande do Sul and the University of Valley of Sinos, respectively. Their consistent help and thoughtful advice allowed this thesis to be my own work, but always steering our research in the right direction whenever they thought it was necessary.

Finally, I must express my very profound gratitude to my family, friends and colleagues for providing me with unfailing support and continuous encouragement throughout my times of study and research. This accomplishment would not have been possible without them.

ABSTRACT

In emergent networks such as Internet of Things (IoT) and 5G applications, network traffic estimation is of great importance to forecast impacts on resource allocation that can influence the quality of service. Besides, controlling the network delay caused with route selection is still a notable challenge, owing to the high mobility of the devices. To analyse the trade-off between traffic forecasting accuracy and the complexity of artificial intelligence models used in this scenario, this work first evaluates the behavior of several traffic load forecasting models in a resource sharing environment. Moreover, in order to alleviate the routing problem in highly dynamic ad-hoc networks, this work also proposes a machine-learning-based routing scheme to reduce network delay in the high-mobility scenarios of flying ad-hoc networks, entitled Q-FANET. The performance of this new algorithm is compared with other methods using the WSNet simulator. With the obtained complexity analysis and the performed simulations, on one hand the best traffic load forecast model can be chosen, and on the other, the proposed routing solution presents lower delay, higher packet delivery ratio and lower jitter in highly dynamic networks than existing state-of-art methods.

Keywords: Routing Protocol. Emergent Networks. Artificial Intelligence. Complexity Analysis.

LIST OF ABBREVIATIONS AND ACRONYMS

FANET	Flying Ad-Hoc Network
MANET	Mobile Ad-Hoc Network
VANET	Vehicular Ad-Hoc Network
MLRM	Multiple Linear Regression Model
RL	Reinforcement Learning
ML	Machine Learning
GA	Genetic Algorithm
QoS	Quality of Service

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The traffic generated by emerging network applications, such as those related to video surveillance (KUNST et al., 2018), and other Internet of Things (IoT) and 5G-supported applications (KLIKS et al., 2018) such as e-health and intelligent transportation systems, is constantly growing. With this growth, it can be expected an overload of the existing licensed spectrum to occur soon. Dealing with this constant growth without jeopardizing the quality of experience offered to the network clients is challenging and may lead to a resource scarcity problem. The traditional approach used by network operators to deal with this problem is to expand and upgrade their network infrastructure, which generally demands huge investments. Network resource sharing is becoming a method that can be used by operators to improve the quality of experience perceived by their clients at a significantly lower cost than in the traditional unilateral approach (Zhang et al., 2019).

To provide an environment where the sharing of network resources is available, methods based on Artificial Intelligence algorithms can be used to forecast the traffic load in these types of networks. Nevertheless, the network operator must take into account two important aspects when choosing a suitable model and maintain an acceptable QoS: accuracy and complexity. Considering this trade-off, the first part of this work analyses the complexity of the most common algorithms that deal with traffic load forecasting. The goal is to identify the ones that demand less computational power to operate, and at the same time, provide high levels of accuracy. For this purpose, the main contributions of this analysis are:

- **Identification of the models that demand less computational power and have high accuracy:** Simulations with different amounts and types of network traffic are performed with several Artificial Intelligence Models implemented in MatLab by forecasting the network traffic load;
- **Discussion of the trade-off between complexity and accuracy:** It must be considered how this trade-off of parameters affects the quality of service delivered by network operators to their customers.

Such contributions are relevant since a well-organized resource sharing initiative may lead to a reduction in the investments necessary for the network operators to deal with the increasing demand for network resources imposed by modern network applications

and services. Nevertheless, resource sharing in these types of emerging networks is not the only challenge to be overcome. Considering situations that have unique characteristics such as network nodes that are constantly moving in regions with artificial and natural obstacles, in addition to the most varied types of climate conditions in which they operate. To overcome these difficulties, robust routing protocols are needed.

The technological advances in the last decades, especially in the development and miniaturization of electronic components lead to the popularization and to the decrease of production costs of Unmanned Air Vehicles (UAV) (MERWADAY; GÜVENç, 2015). This allowed UAVs to be applied in many different applications in both military and civilian domains, such as to perform surveillance and monitoring tasks (SUN et al., 2011), provision of communications networks in natural disasters or in conflict regions, among others. Common application areas also include wildfire monitoring (BARRADO et al., 2010), search and rescue operations (MANATHARA; SUJIT; BEARD, 2011), reconnaissance operations, hazardous site inspection (MAZA et al., 2011), range extension and the agriculture field (XIANG; TIAN, 2011).

The use of a single UAV is already well understood and even considered "ordinary", but the use of multiple simultaneous UAVs, which can provide great advantage over the option of a single UAV, is still a research area with many possibilities to be explored. Despite its usefulness, this multiple UAVs setup scenario poses a challenge regarding to the communication among them, which is not a trivial task (SAHINGOZ, 2013). Observing the importance of networked UAV systems, the second part of this thesis focuses on Flying Ad-Hoc Networks (FANETs).

In FANETs composed of many UAVs (YASSEIN; DAMER; JORDAN, 2016), the high mobility of the nodes creates a highly dynamic network topology. To address it, an adaptive and autonomous routing protocol is needed, which means that the routing protocol for FANETs should have the ability to find a reliable neighbor to complete the transmission, by detecting the change in the environment. In this context, Q-learning is an adaptive machine learning with environmental feedback as input, which contributes to adaptive routing design and presents a promising approach to be used in a routing protocol scheme (WATKINS; DAYAN, 1992).

The routing protocol based on Q-learning relies on the local data of the neighboring nodes and it did not make any assumption about any other network aspect. Most of the Q-learning based routing protocols work by making the best choice among the neighbors at any moment to transmit a packet to the destination. Due to the requirement of real-time

data transmission in multiple UAVs systems, to support a number of applications, it is crucial for a routing protocol to have low delay.

Since UAV networks present a high dynamic, if Q-learning parameters such as the learning rate and discount factor are fixed, the accuracy of action selection declines, and the selected link may have low probability of connecting to a neighbor node. This is a strategy used by most of the existing Q-learning based routing protocols which may limit they performance.

Based on these limitations, the second part of this work presents a novel Q-learning that combines other approaches that use Reinforcement Learning to create an optimized routing protocol for FANETs, called Q-FANET. The main contributions of this proposed solution are:

- **Delay decrease:** Without a fixed routing table, Q-learning can be used with specific rules and mechanisms to choose the optimal routing path based on a low delay constraint;
- **Exploration of last episodes with different weights:** Standard Q-learning always consider the most recent last episode to update the Q-values, which may lead to imprecise decisions. Therefore, the proposed solution considers a finite amount of last episodes;
- **Enhanced protocol parametrization based on channel conditions:** The transmission quality is a important element that can impact directly on the delay of data transmission in a FANET, even when the optimal route is selected. The proposed solution also takes into account the channel conditions as a new metric to calculate the Q-values.

The remainder of this work is organized as follows. In Chapter 2, an overview of the concepts behind the main Artificial Intelligence models for traffic load forecasting in network are presented, as well a review of the aspects of routing protocols for FANETs. This chapter also presents the relevant related work developed in both areas. Chapter 3 presents the first part of the work, an analysis of the trade-off between complexity and accuracy of the models used for resource sharing in emergent networks. Chapter 4 details the second part of the work, a proposal of a routing protocol for FANETs, called Q-FANET, as well as the elaborated simulation scenario, performed tests and obtained results, which are deeply discussed. Finally, this work is concluded in 5 presenting final remarks and directions for future investigations.

2 BACKGROUND AND RELATED WORKS

This chapter presents the background on Artificial Intelligence models used to forecast the traffic load for resource sharing in networks, as well as the routing protocols for FANETs. The chapter also discusses relevant related work in both areas.

2.1 Artificial Intelligence for Resource Sharing in Networks

The first step towards resource sharing relies on the accurate estimation of network resource usage. Artificial intelligence algorithms often forecast the behavior of the network clients, considering the amount and frequency of network traffic generated by each network client. Approaches like Multiple Linear Regression (MLR), Regression Trees (RT), and Fourier Analysis (FA) have been widely used to this end in the past. A more recent approach of related work is to implement modern approaches such as Reinforcement Learning (RL), Deep Learning (DL) and Genetic Algorithms (GA) for traffic load forecasting. In (KUNST et al., 2018) is showed that these techniques, in general, are very accurate to estimate the resources occupation and consequently to allow efficient resources sharing.

2.1.1 Multiple Linear Regression Model

Multiple Linear Regression Model (MLRM) is a statistic model widely implemented for predictive analysis. MLRM explains the relationship between one continuous dependent variable and two or more independent variables. This model implements three main functions that impact on the complexity of the algorithm. First, the algorithm chooses predictors to use during the regression phase, which aims at finding the variance inflation factor (VIF). Then, VIF measures how much the variance of an estimated regression coefficient increases due to collinearity. Finally, the algorithm removes the variables with high VIFs to find the best forecasting solution.

Many pieces of research applied this model in the context of data communications. The work in (PAPADOPOULI; RAFTOPOULOS; SHEN, 2006) proposed a solution to allow short term traffic load forecasting for wireless networks. The authors evaluated several traffic forecasting algorithms that consider the recent traffic history and information

related to the current traffic flow. (LIU; LEE, 2014) simulated the behavior of MLRM and six other algorithms to evaluate whether these algorithms can predict the throughput of mobile data networks. The approach proposed by (NAIMI et al., 2014) is focused on predicting network metrics such as the number of retransmissions and the total time expected to transmit a data packet. The predictions are used to adjust the routing metrics in ad-hoc wireless networks. (NOULAS et al., 2012) analyzes the information of Foursquare social platform to predict user mobility. The user, global, and temporal feature sets are analyzed, aiming at predicting the next check-in state.

2.1.2 Regression Tree Model

Regression Tree Model (RTM) implements a decision tree to serve as a predictive model that observes an item to conclude its target value. This model mainly applies to the fields of machine learning, statistics, and data mining, among others. In the field of wireless networks, RTM is often used to model the correlation between resource consumption (*e.g.* throughput) and other factors, like the number of users in a given network, and the priority given to each user. This correlation is generally calculated based on a data set that generates a binary tree. This tree serves as a temporary data structure used to divide the data set into subsets of the data, which are as homogeneous as possible following the response variables.

According to (STROBL; MALLEY; TUTZ, 2009), a regression tree is a simple non-parametric regression approach, which has as main characteristic the featured space. In other words, space spanned by all predictor variables is recursively split into a set of rectangular areas. The authors advocate that the technique group observations with similar response values. After this grouping, the solution predicts a constant value within each resulting area. (XU et al., 2013) proposed a model to forecast network performance in a real-time mobile applications scenarios. The solution uses a machine learning framework that implements the concept of regression trees to identify the trend of the network performance over short, fine-grained time windows, using previously available observations.

2.1.3 Fourier Series Model

Fourier series can forecast the behavior of phenomena in various fields of research, such as engineering, seismology, and economics, among others. A contribution of this model is to capture the effects of seasonality over the data. The implementation of the Fourier Series Model follows the concept of curve fitting to find the solution of the heat equation.

The work of (LYE; YUAN; CAI, 2009) has important research that relates to this model. Their solution decomposes the given time series (*i.e.* historical data) into a linear combination of frequency components via an orthogonal transform method. This approach results in a new method for forecasting through the analysis of the frequency domain. In this sense, the authors use Fourier's approach for forecasting changes in electricity load, transportation, and prices.

2.1.4 Reinforcement Learning

Reinforcement Learning (RL) is an important paradigm of the Learning Process in the field of Artificial Intelligence studies as well as an area of Machine Learning (ML) (SUTTON; BARTO, 2018). A simple analogy that is possible to imagine is a person that does not know the flavor of an specific food and tries it for the first time. This individual may identify the food as something bad or good and this acquired knowledge may be used to decide next time if this individual should eat or do not eat that food.

In the context of Computer Science, RL may be applied to an algorithm that has some knowledge about the task that it should perform and can use it to make better choices in order to complete the task. As shown in (BITHAS et al., 2019), ML can deal with several challenges involving the communication in emergent networks (specially FANETs), as well as improving various design and functional aspects such as channel modeling, resource management, positioning, and security. The key components of a RL algorithm are: the agent, the environment, the state, the action and the reward.

The agent learns over time to behave optimally in an environment by interacting continuously. The agent, during its course of learning, experiences various scenarios in the environment, which are called states. While in a particular state, the agent may choose from a set of allowable actions and, depending on the result of each action, it receives rewards or penalties. The learning agent overtime learns to maximize these rewards to

behave optimally at any given state it is in.

Q-Learning is a basic form of RL which uses Q-values (also called action values) to iteratively improve the behavior of the learning agent. These action values are defined for states and actions. $Q(S, A)$ is an estimation of how good is it to take the action A at the state S.

In Q-Learning, an agent starts from a given state and makes several transitions from its current state to a next state. Every transition happens due to an action considering the environment the agent is interacting with. In every step of the transition, the agent takes action, receives a reward from the environment, and then transitions to another state until it reaches the goal. This situation is called the completion of an episode. This estimation of $Q(S, A)$ will be iteratively computed using a specific rule that estimates the value of Q at every interaction of an agent with the environment is expressed by (2.1):

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma maxQ(S', A') - Q(S, A)) \tag{2.1}$$

S is the current state of the agent, A represents the current action picked according to some policy, S' is the Next State where the agent ends up, A' is the next best action to pick using current Q-value estimation, and R is the reward received from the current action. Other important parameters of this update function are:

- **Discounting Factor for Future Rewards** (γ)**:** a value set between 0 and 1. Future rewards are less valuable than current ones. Therefore they must be discounted;
- **Learning rate** (α) **:** step length taken to update the estimation of $Q(S, A)$;
- ϵ**-greedy policy:** a simple method to select actions using the current Q-value estimations. The probability of choosing the action with the highest Q-value is given by (1-ϵ), while the probability of selecting a random action is (ϵ).

2.1.5 Deep Learning Model

The majority of modern deep learning models implement a multilayer artificial neural network (BENGIO; COURVILLE; VINCENT, 2013). In this type of model, each layer learns to transform input data into a more abstract and composite representation. The main difference between deep learning and the traditional approach of an artificial

neural network model is the number of layers that deal with the inputs. Deep learning architectures are often constructed with a greedy layer-by-layer method, choosing which features improve performance. For supervised learning tasks, deep learning methods translate data into compact intermediate representations and derive layered structures that remove redundancy in the representation. One can also use deep learning to implement unsupervised learning tasks, which is an important benefit since unlabeled data are more abundant than labeled data. Examples of deep structures that fit unsupervised learning are neural history compressors and deep belief networks (BENGIO; COURVILLE; VINCENT, 2013).

Deep learning models can deal with forecasting-related problems in diverse fields. For example, (KHATIB et al., 2012) proposed the evaluation for sunlight based radiation forecast to create precise models for anticipating hourly sun based radiation based on the number of daylight hours, day, month, temperature, dampness, and area position. They regarded the neural network-based models such as feed-forward backpropagation, cascade forward backpropagation, and Elman backpropagation for the hourly solar radiation prediction. The work of (CHEEPATI; PRASAD, 2016) modeled the load forecasting of electricity power plant using multilayer neural networks. In that case, the authors considered the historical data as input (average measurements of electrical load) from the last 24 hours, obtaining a mean average error of 2.9%.

2.1.6 Genetic Algorithm Model

In a genetic algorithm (GA), a population of candidate solutions (individuals) evolves to seek for the best solution for a problem. Each candidate solution has a set of properties (chromosomes or genotype) that suffer mutations (Srinivas; Patnaik, 1994). The evolution process begins with a population of random individuals that form a new generation in each iteration of the algorithm. For each generation, the fitness (value of the objective function in the optimization problem) of every individual in the population is evaluated. The more fit individuals are selected from the current population, and each individual's genome is modified (recombination, mutation) to form a new generation of candidate solutions to be used in the next iteration. The process terminates when either a maximum number of generations has been created, or the population reaches a satisfactory fitness level.

The GA approach applies to the forecasting of diverse variables. The work of

(Kampouropoulos et al., 2018) proposes a method for the energy optimization of multi-carrier energy systems. The method combines an adaptive neuro-fuzzy inference system, to model and forecast the power demand of a plant. Moreover, the authors implement a genetic algorithm to optimize the energy flow, taking into account the dynamics of the system and the equipment thermal inertia.

2.2 Related Works in Resource Sharing for Network Clients

Find the accurate estimation of network usage is of great importance to network operators, which may need to implement resource sharing to seek for quality of service to their customers. However, accuracy is not the only variable to consider in resources sharing, since a fast allocation decision is often necessary for realistic resources sharing scenarios. Thus, one also needs to consider the computational complexity of the algorithms that forecast the behavior of the customers.

Many related pieces of research have dealt with traffic load forecasting in the context of wireless network operators. In (AL-TURJMAN et al., 2017), a traffic model is presented for a new generation of sensor networks that support a wide range of communication-intensive real-time multimedia applications. The model is used to investigate the effects of multi-hop communication on Intelligent Transportation Systems (ITS) through the implementation of a Markov discrete-time queuing system, designed to simulate the network traffic. (QI; YAN; PENG, 2018) proposed a stochastic geometry tool to determine the coverage probability and spectral efficiency of Unmanned Aerial Vehicle networks. Although very relevant, none of these related works focuses on analyzing the computing power demanded to operate the proposed solutions. Consequently, the trade-off between accuracy and complexity, and its impacts on the implementation of resources sharing among network operators are not analyzed.

2.3 Routing protocols for FANETs

This section aims to present background aspects on routing protocols for FANETs. Towards this aim, first different types of routing protocols are introduced, then, aspects related to each one of the major protocols are discussed. The existing routing protocols used in so called Mobile Ad Hoc Networks (MANETs) and Vehicular Ad Hoc Networks

(VANETs) are not completely suitable to be directly applied in UAVs networks, as they must be adapted to the higher degree of mobility that characterizes FANETs and the consequently more frequent changes in the topology and in the communication links (JIANG; HAN, 2018). Therefore, the routing protocols used in FANETs are classified in two categories: single-hop routing (FU et al., 2013) and multi-hop routing (Jean-Daniel Medjo Me Biomo, Thomas Kunz, 2014), each one containing specific types of protocols.

In single-hop routing protocols, a static routing table is used to define the transmission paths. This table is computed and loaded before the operation of the UAV nodes and cannot be changed until the network operation is over.

As for the multi-hop routing protocols, packets are forwarded hop by hop towards the destination. The selection of the proper hop node is the core issue of the routing discovery. Usually, these protocols are divided into two categories: topology-based and position-based routing (BOUKERCHE et al., 2011). Furthermore, the first category consists of three types of protocols: proactive, reactive, and hybrid.

2.3.1 Static Protocols

The static protocols are considered lightweight and are designed to be implemented in fixed topologies. Nevertheless, they are not fault tolerant, since it is necessary to wait until the end of the operation in case of failure to update the routing table, what makes these protocols not suitable for dynamic environments. To exemplify this class of protocols, some algorithms are presented, as follows.

Load-carry-and-deliver (LCAD): Also known as Load Carry and Deliver Routing (LCDR) (CHENG et al., 2007) is a classical static routing protocol. In this protocol, the UAVs carry and deliver packets from a ground node to the destination node. The process consists of three main phases: loading the packets from source node, carrying packets when flying, and delivering the packets to the destination node. LCAD provides high network throughput and it is recommended for delay-tolerant networks. Even so, increasing distances between source and destination can cause latency problems.

In (HEIMFARTH; ARAUJO; GIACOMIN, 2014), an adaptation of the LCAD protocol, is used to physically carry packets, as "data mule", across the disjoint partitions of a Wireless Sensor Network (WSN). The simulations show the effectiveness of the system for concentrated traffic. Also exploring the concepts of the LCAD protocol with UAVs serving as "data mules", the authors in (PALMA et al., 2017) proposed a field

experiment, using the obtained results to define and implement an emulator for intermittent links of the network. The results showed that trajectories at higher altitudes, despite increasing distance between nodes, improve communication performance.

Data Centric Routing (DCR): DRC is a routing protocol for FANETs in which the destination node (either a ground node or a UAV) sends queries with a specific goal to gather specific information from a particular zone (KO; MAHAJAN; SENGUPTA, 2002). This model involves minimum assistance when small numbers of UAVs are on the path.

2.3.2 Proactive Protocols

Proactive protocols, also know as active routing protocols, record and store the routing information in each UAV belonging to the network. Each node updates its routing table to meet changes in the network topology. Therefore, the routing paths can be selected to transmit packets with minimum waiting time (YANG; TIAN; YU, 2005). Although highly used due to its characteristics of serving high-mobility network scenarios, this type of protocols present several disadvantages, such as the amount of control packets necessary for route establishment, what generally increases the communication overhead. Some examples of algorithms that implement the concepts of proactive multi-hop protocol are presented as follows.

Destination Sequenced Distance Vector (DSDV): in this routing protocol, each node behaves like a router, therefore, each UAV can maintain its routing table and sequence number for each node of the network, triggering update mechanisms whenever there is a change in the topology (BANI; ALHUDA", 2016). Despite being highly used, this kind of protocols is generally not applied to dynamic networks because it is unable to deal with fast changes in the network topology and lacks support for multipath routing (WU; HARMS, 2001).

Optimized Link State Routing Protocol (OLSR): this well-known approach uses two types of messages to establish the routing information. The "hello" message finds the neighbor nodes and generates a list of neighbors and broadcasts it to the next hop. The "topology control message" is used to maintain the topology information, making each node to update its routing table when there is a change in the topology information (VASILIEV; ABILOV; KHVORENKOV, 2016). OLSR is able to deal with dynamic environments at the cost of a high overhead, due to the frequent control message exchange demanded by the algorithm. Because of the high overhead in the original implementation

of OLSF, alternative approaches have been proposed, such as Speed-aware Predictive-OLSR (POLSR) protocol (ROSATI et al., 2013), in which GPS information is used to assist the routing protocol. Based on GPS information, the relative speed between two nodes can be obtained, which is used to evaluate the communication link quality, allowing the algorithm to select the UAV with higher link quality for routing.

2.3.3 Reactive Protocols

Other common approach in the context of multi-hopping, are the reactive protocols, also known as passive routing or on-demand routing protocols. This class of routing protocols presents low overhead, since the routing information is created only when there is a communication between two nodes. However, the overhead reduction comes at the cost of increasing the end-to-end delay, due to the processing time required before a path is established (HABIB; SALEEM; SAQIB, 2013). Examples of algorithms that implement the concepts of reactive protocols are provided as follows.

Dynamic Source Routing (DSR): better suitable for high-mobility networks, this protocol establishes routing paths only when they are necessary, making it more adaptive to link dynamics and failures (JOHNSON; MALTZ, 1996). In this protocol, the sender node sends a unique request id to its neighbor nodes, which use their route caches to send the packet towards the destination. The main challenge in this protocol is to refresh the route caches in each UAV in highly dynamic environments.

Ad-hoc On-demand Distance Vector (AODV): this protocol is similar to DSR in a way that both operate on demand. The main difference between them is that AODV assigns dedicated time slots for packet transmission to avoid congestion and improve the packet delivery ratio. Every node in this protocol can store different entries in the table for every destination and establish the path for the packet transfer from source to destination (MAISTRENKO; ALEXEY; DANIL, 2016). If the next hop is unreachable, there are two options: drop the packet or resend it using the new established route. The protocol consists of three phases: routing discovery, packet transmission, and route maintenance.

2.3.4 Hybrid Protocols

The hybrid protocols are other important class, representing a combination of proactive and reactive routing protocols, used to overcome their limitations: time demanded to find routes and control messages overhead, respectively. In hybrid routing protocols, the network is divided into different regions. Proactive protocols are used to implement intra region routing, while reactive protocols are used for inter region routing (JAILTON et al., 2017). Research approaches that implement this concept are presented as follows.

Zone Routing Protocol (ZRP): the concept of zones is implemented, in which each node node belongs to a zone with a range called R. Proactive routing is used inside the zone, while reactive routing is used when information is needed to be sent outside the zone.

Temporarily Ordered Routing Algorithm (TORA) is a scalable, efficient and adaptive algorithm that finds several routes between source and destination, supporting link reversal and using an approach very similar to Multi-Path routing protocols (PARK; CORSON,). Unfortunately, TORA is a complex protocol, presenting high overhead when applied to large networks. A Rapid-reestablish Temporarily Ordered Routing Algorithm (RTORA), based on TORA, is proposed in (ZHAI; DU; REN, 2013). This algorithm adopts a reduced-overhead mechanism to overcome the adverse effects caused by link failure in TORA. Simulations results demonstrated that this approach has lower control overhead and better end-to-end delay performance in comparison to TORA.

2.3.5 Position-based Protocols

Position-based protocols are another common approach, designed to deal with the high speed of UAVs. This approach is justified because static routing tables maintained in proactive routing protocols are not always effective for dynamic networks. At the same time, in reactive protocols, it is costly to repeat routing establishment before each packet transmission (RAW; DAS, 2012). To solve these problems, researchers proposed a new kind of routing protocol based on geographic location information (LIN et al., 2012).

Greedy Perimeter Stateless Routing (GPSR): this protocol is the major example of this class. Simulation results demonstrate that GPSR (KARP; KUNG, 2000) outperforms most proactive and reactive protocols with respect to packet delivery ratio and

transmission delay. Since it relies on a greedy approach, this protocol fails when a packet arrives at a node that has no neighbor. However, recovery strategies have already been proposed to solve this problem, such as forwarding the packet to the furthest neighbor (BIOMO; KUNZ; ST-HILAIRE, 2014).

2.3.6 Hierarchical Protocols

The last class of routing protocols explored in this paper is the class of hierarchical protocols. The algorithms in this class are based on a clustered approach, in which the formation of the cluster poses as the main issue for these protocols.

Mobility prediction clustering (MPC): this is the first example of this class of protocols. Due to high mobility in FANETs, the formation of clusters demands fast updates, and this algorithm addresses this issue by predicting the updates regarding the network topology (ZANG; ZANG, 2011). It forecasts the structure of mobility of the UAVs by using structure prediction algorithm and a link expiration based mobility model, which allows the selection of the most suitable UAV to be the cluster head.

Clustering Algorithm of UAV Networking: another clustering algorithm is proposed by (KESHENG; JUN; TAO, 2008). In this approach, the clusters for multi-UAV systems are constructed on the ground and then updated during the flying operation. The clustering plan is calculated for selection of the cluster heads according to the geographical information. Results showed that this algorithm can be used to increase the stability and guarantee the ability of dynamic networking.

2.4 Related Works in Routing Protocols for FANETs

From all the existing routing protocols that are used in FANETs, the ones which are based on Q-Learning methods have proven to be a promising strategy to deal with the dynamic changes and the high mobility of this type of emergent network. Q-Grid (LI et al., 2014) is a routing protocol for VANETs that considers both macroscopic and microscopic aspects when making the routing decision, dividing the region into different grids. Q-Grid computes the Q-values of different movements between neighboring grids for a given destination using Q-learning. Simulation comparison among Q-Grid and other existing position-based routing protocols confirms its advantages.

Q-Learning based Adaptive Routing model (Q-LAR) (SERHANI; NAJA; JAMALI, 2016) detects the level of mobility at each node in the network and uses a new metric, called Q metric, which accounts for the static and dynamic routing metrics, and which are combined and updated to the changing network topologies. Both are deployed on OLSR protocol, with the simulations validating the effectiveness of Q-LAR over standard OLSR protocol.

Q-Learning based gegraphic routing (Q-Geo) (JUNG; YIM; KO, 2017) uses a scheme to reduce network overhead in high- mobility scenarios. The performance of Q-Geo is evaluated in comparison with other methods using the NS-3 simulator. It was found that Q-Geo has a higher packet delivery ratio and lower network overhead than existing methods.

Q-Fuzzy (YANG; JANG; YOO, 2020) uses fuzzy logic with link- and path-level parameters - which include transmission rate, energy state, and flight status between neighbor UAVs - to determine the optimal routing path to the destination. These parameters are dynamically updated by the RL method. The results show that, in comparison with distance vector routing based on Q-Values, Q-Fuzzy can maintain low hop count and energy consumption and extend the network lifetime.

Q-Learning Multi-Objective Routing (QMR) (LIU et al., 2020) is a routing protocol that uses adaptive parameters (as the learning rate and the mechanism of exploration) combined with link condition and specific constraints to provide low delay and low energy consumption. Compared to Q-Geo, QMR provides higher packet arrival ratio, lower delay and energy consumption.

In the context of MANETs, (HE et al., 2019) uses a Q-Learning based CSMA/MAS protocol. In this method, all nodes in network can be synchronized and then served in a round-robin way without contention collisions. At the network layer, it modifies Q-Geo and Q-Grid. The results show that this approach of the transmission protocol can provide higher packet arrival ratio and lower end-to-end delay than the existing transmission protocols.

QNGPSR (LYU et al., 2018) is a routing protocol based on well-known GPSR for the ad-hoc network. By using the reinforcement learning and neighbor topology information to make next-hop selection in multiple available paths, the probability of exploring perimeter forwarding mode that may induce network delay can be greatly reduced. The results show that QNGPSR obtains a higher packet delivery ratio and a lower end-to-end delay compared with the original GPSR.

In the context of cognitive sensor networks, Q-Noise+ (FAGANELLO et al., 2013) uses three improvements to the dynamic spectrum allocation algorithms based on reinforcement learning. Simulation results show that Q-Noise+ allows allocating channels with up to $6dB$ better quality and 4% higher efficiency than standard Q-Learning.

In this work, both techniques used in QMR and improvements of optimal channel selection made by Q-Noise+ are considered, with focus on providing low delay, using a finite number of last episodes to calculate the Q-Values.

3 COMPLEXITY ANALYSIS OF TRAFFIC LOAD FORECASTING MODELS

In this chapter, the complexity of the most common algorithms that deal with traffic load forecasting is analysed. The goal is to identify the ones that demand less computational power to operate, and at the same time, provide high levels of accuracy. The trade-off between complexity and accuracy is discussed, considering how it affects the quality of service delivered by network operators to their customers. The results obtained indicate the more suitable artificial intelligence algorithms that enable efficient resource sharing among network operators.

3.1 Multiple Linear Regression Model

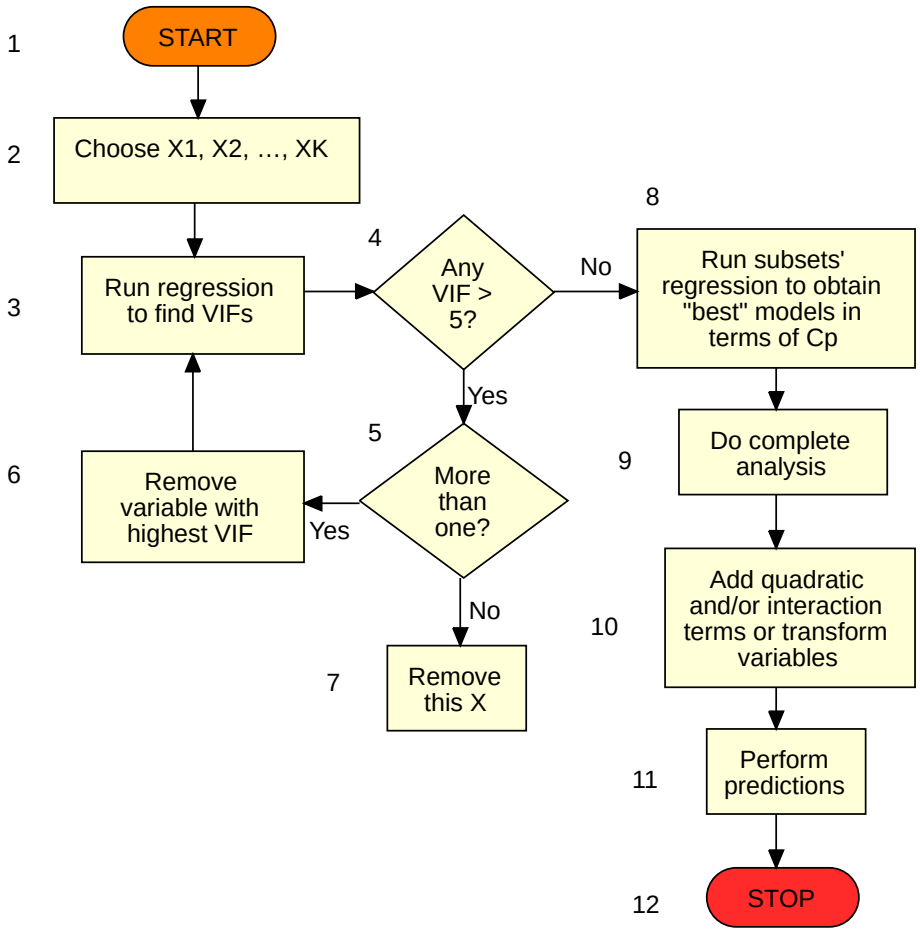
MLRM can be applied to perform the prediction of scenario variables based on historical information. The complexity analysis of this algorithm must take into account three main functions performed during its operation: (I) selection of predictors, (II) search for VIFs, and (III) removal of variables with high VIF. The computing cost of each function and the resulting order of complexity are presented in Table 3.1. The flowchart for the algorithm is presented in Figure 3.1.

The first observation regarding the complexity analysis of this algorithm is that step 2 has linear complexity, i.e., $O(n)$, where n is the number X_k to be chosen. In step 3, assuming that each X is a $(n \times m)$ matrix, the linear regression execution takes $(X'X)^{-}1X'y$ computing instructions, resulting in an order of complexity $O(k^2 * (n+m))$.

Continuing the complexity analysis, step 4 has constant complexity, i.e., $O(1)$, since VIF values were selected beforehand. Therefore, it is only necessary to check if there are more than one of these values. Step 5 also has linear complexity, $O(1)$, because this step only removes a value X from the system. Step 6, in turn, has linear complexity, $O(n)$, since it searches among all VIF values to check whether any of them is greater than 5. Step 7 also presents linear complexity, $O(1)$, because this step only removes a value X from the system.

The next function executed in MLRM is to find the best model in terms of C_p, the Mallow's C_p (step 8). To do that, the regression process is executed over subsets, leading to the same complexity observed in step 3. In step 9, the algorithm passes through the subsets to perform an analysis of the results, leading to a complexity of $O(n)$, where n is the number of subsets. Finally, step 10 presents linear complexity, $O(n)$, where n is the

Figure 3.1: Multiple Linear Regression algorithm flowchart



number of quadratic, interaction terms, or transform variables to be added.

Considering the complexity of each step, it is possible to conclude that the joint complexity of the algorithm is equal to $O(2 * (k^2(n + m) + 3n)$. Therefore, it can be considered that the general order of complexity for the Multiple Linear Regression Model is $O(n^2)$.

The pseudo-code for the Multiple Linear Regression Model is described in Algorithm 1:

Algorithm 1 MLRM

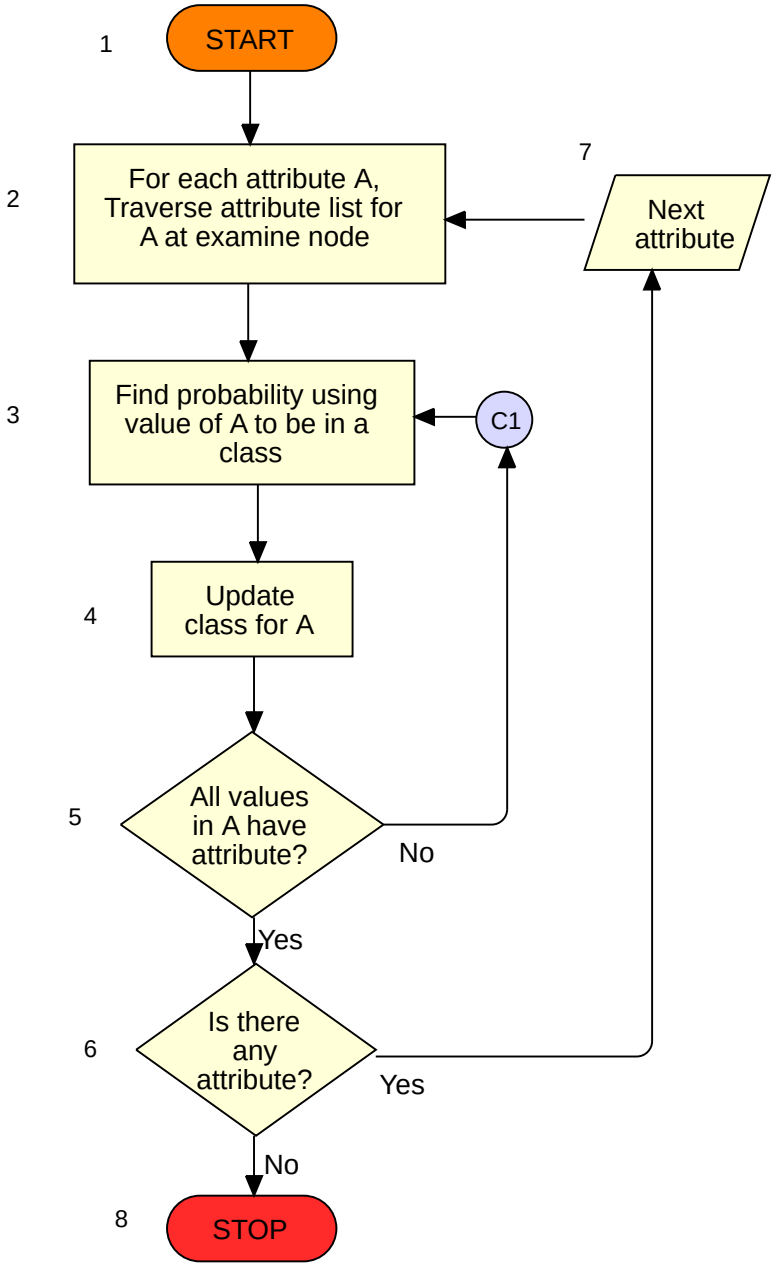
procedure CHOOSE $X, X_2, ..., X_K$
  procedure RUN REGRESSION TO FIND VIFS
    if Any VIF > 5 **then**
      if More than one **then**
        Remove variable with highest VIF
      else
        Remove this X
      end if
    else
      Run subsets regression to obtain "best" models in terms of C_p
    end if
    procedure DO COMPLETE ANALYSIS
      procedure ADD QUADRATIC AND/OR INTERACTION TERMS OF TRANSFORM VARIABLES
        procedure PERFORM PREDICTIONS

3.2 Regression Tree Model

The complexity of RTM relies on two main functions, which execute over a binary tree. First, the algorithm computes the best split point, using a function that creates subsets of the original data set. Next, the algorithm selects the best attributes aiming at implementing an accurate forecast. In this case, the computing cost of the algorithm is proportional to the size of the data set that feeds the model. The information related to both the computing cost and the overall order of complexity of TRM is summarized in Table 3.1 and the flowchart for the algorithm is presented in Figure 3.2.

The complexity of this algorithm mostly depends on the procedure for selecting

Figure 3.2: Regression Tree algorithm flowchart.



the best attribute to split and the split point. The two parameters that play a pivotal role in the analysis are the number of attributes and the number of training examples.

The part of the algorithm that demands more resources comprises the computation of the best split point for the continuous attribute (discretization), and the selection of the best attributes from among the set of candidate attributes to split on. One can assume that the complexity is quadratic in the number of attributes (denoted a) and linear in the number of examples (denoted n). Therefore, the total complexity is $O(n * a^2)$. With this result, it can be considered that the general order of complexity for the Regression Tree model is $O(n^2)$.

The pseudo-code for the Regression Tree is described in Algorithm 2:

Algorithm 2 Regression Tree

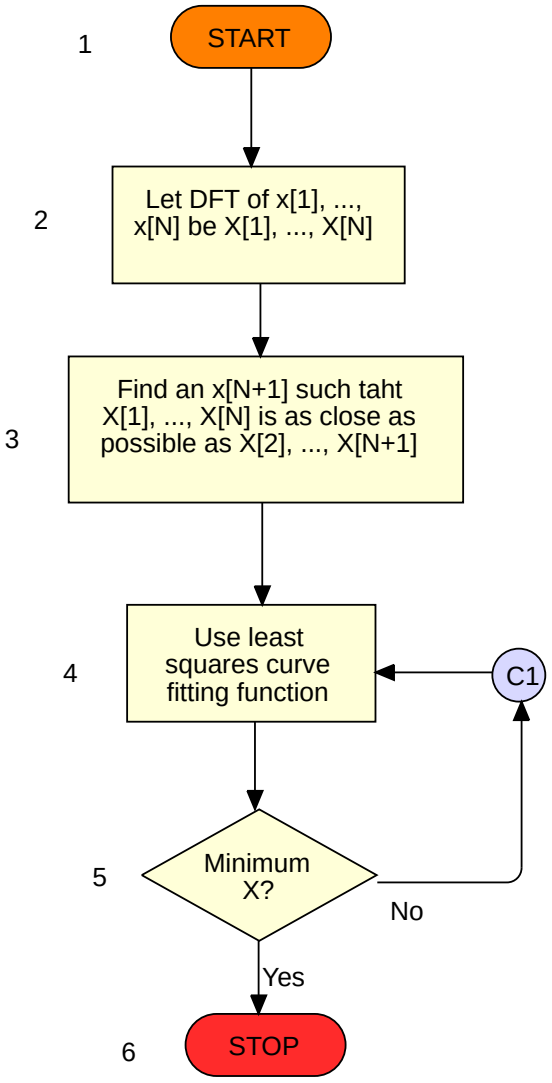
while Is there any attribute = TRUE **do**
    Get next attribute
    for Each attribute A **do**
        Traverse attribute list for A at examine node
    end for
    while All values in A have attribute = FALSE **do**
        Find probability using value of A to be in a class
        Update class for A
    end while
end while

3.3 Fourier Series Model

The Fourier Series Model can be adapted to apply it for network traffic forecasting by identifying traffic patterns based on previous measurements of the traffic. The flowchart for the Fourier series implementation is presented in Figure 3.3.

The step-by-step analysis of the algorithm's complexity allows one to observe that the DFT in step 2 is an important function for the implementation of the Fourier Series. This function sets a series $X[1]$, ..., $X[N]$ and, consequently has a complexity in the order of $O(2N^2)$. Steps 3 and 4 regard to the optimization problem solved by the algorithm using the least-squares curve fitting function. In this case, one can assume that for a least-squares regression with N training examples and C features, it takes $O(C^2N)$ to multiply X^T by X; $O(CN)$ to multiply X^T by X; and $O(C^3)$ to compute the LU factorization of

Figure 3.3: Fourier Series algorithm flowchart



X^TX and use it to compute the product $(X^TX) - 1(X^TY)$. Asymptotically, $O(C^2N)$ dominates $O(CN)$. We can also consider that $N > C$, which means that $O(C^2N)$ asymptotically dominates $O(C^3)$. This results in a total complexity of $O(2N^2 + C^3)$. Therefore, it can be considered that the general order of complexity for the Fourier Series model is $O(C^3)$.

The pseudo-code for the Fourier Series model is described in Algorithm 3:

Algorithm 3 Fourier Series

procedure LET DFT OF $x[1], ..., x[N]$ BE $X[1], ..., X[N]$

procedure FIND AN $x[N+1()$ SUCH THAT $X[1], ..., X[N]$ IS AS CLOSE AS POSSIBLE AS $X[2], ..., X[N+1]$

while Minimum X = FALSE **do**

Use least squares curve fitting function

end while

3.4 Reinforcement Learning

Q-Learning is the most well known algorithm that uses Reinforcement Learning and it can be easily adapted to be applied to network traffic forecasting. The flowchart for the Q-Learning algorithm is presented in Figure 3.4.

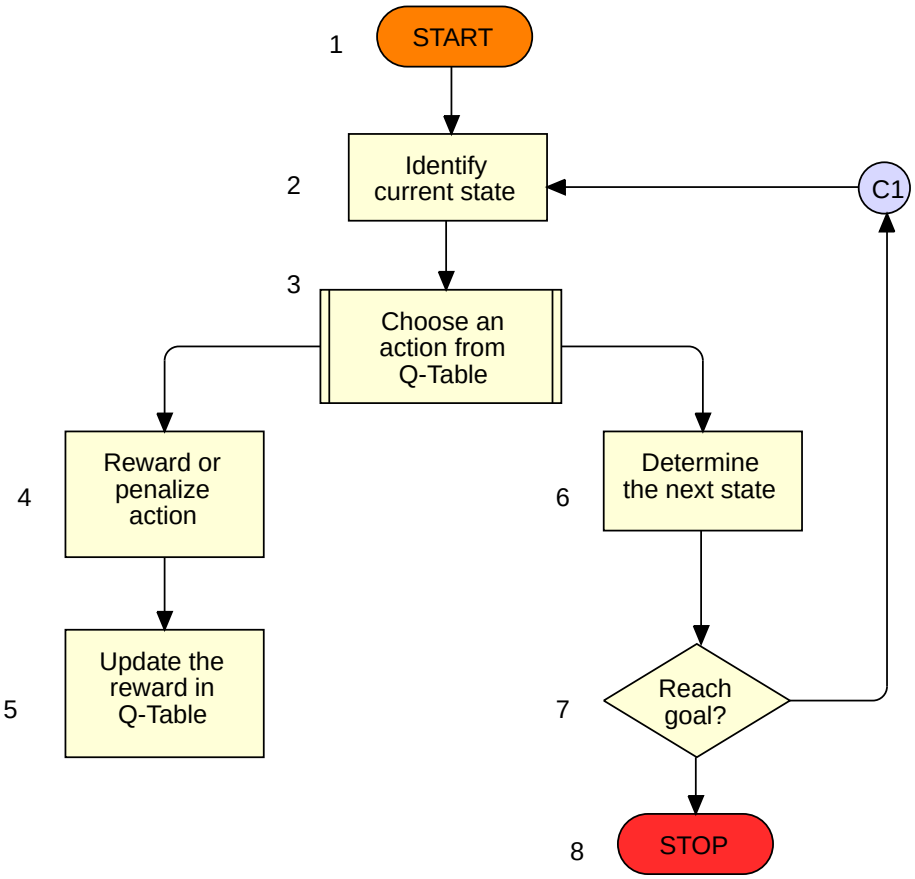
The analysis of the complexity of Q-learning shows that steps 2, 4, 6, and 7 have constant complexity, i.e., $O(1)$. Step 3 has linear complexity, $O(n)$ since it chooses one of the n actions from the Q-Table. Step 5 also presents linear complexity, in the order of $O(m)$, where m is the sum of updating the r rewards and the p penalties. The final complexity value for this model is given by $O(n \cdot m)$. If assumed that the number of actions is equal to the total of rewards and penalties of the model, then the total complexity for the Q-Learning model is $O(n^2)$.

The pseudo-code for the Q-Learning model is described in Algorithm 4:

3.5 Deep Learning Model

A deep learning model implementation usually follows three distinct phases. The first phase regards the selection of a data set for validation, training, and testing purposes. Next, occurs the adaptation of the parameters of this data for training the artificial neural

Figure 3.4: Q-Learning Algorithm flowchart



Algorithm 4 Q-Learning

while Reach goal = FALSE **do**
    Identify current state
    Choose an action from Q-Table
    Reward or penalize the action
    Update the reward in Q-Table
    Determine the next state
end while

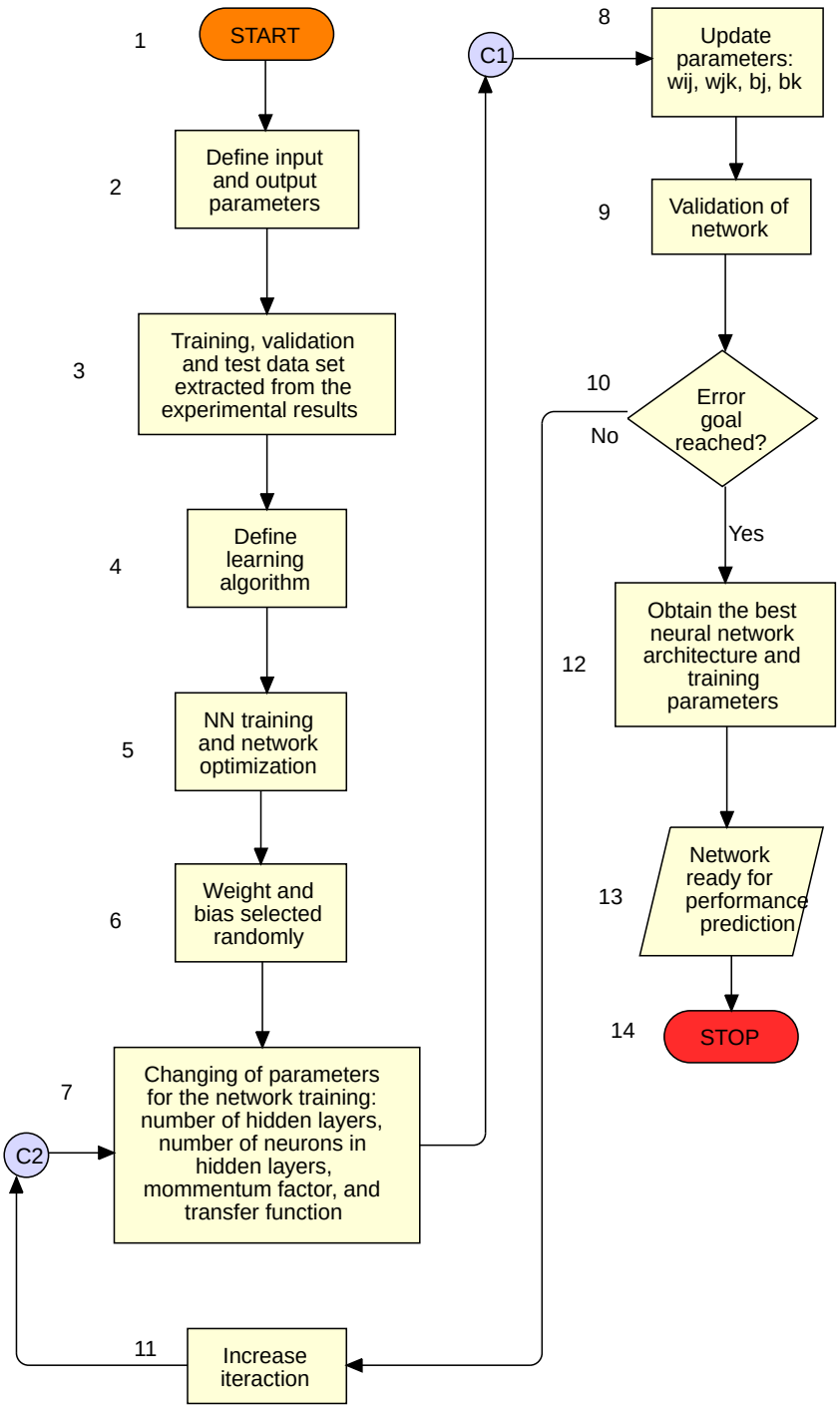
network. The final phase consists of obtaining the best neural network considering the available training data. The computing cost of each phase and the resulting order of complexity are available in Table 3.1, while Figure 3.5 presents the flowchart for this algorithm.

Steps 2, 7, and 13 of the flowchart present constant complexity, i.e., $O(1)$. Steps 4 and 9 have linear complexity $O(n)$, where n is the number of elements to split into the three datasets. The major steps that define the complexity of this model are the feed-forward and back-propagation processes that occur in step 5, the training in steps 6 and 8, and the network simulation in step 12. Therefore, we do not consider the complexity of the remaining functions on our analysis, since the complexity of the mentioned processes will dominate the overall evaluation. In the feed-forward approach, for each layer, matrix multiplication and an activation function are computed. It is known that naive matrix multiplication has an asymptotic runtime of $O(n^3)$, assuming that there is the same number of neurons in each layer and that the number of layers equals the number of neurons in each layer. For the back-propagation, by assuming that gradient descent runs for n iterations and that there are n layers each with n neurons, the total complexity of back-propagation is expressed as $O(n^6)$.

Assuming that only the computational times for the feed-forward and back-propagation are relevant to estimate the complexity of this model, the total complexity for the neural network model is given by $O(n^3 + n^6)$. Nevertheless, since the learning process is an NP-hard problem, this conclusion is not definitive (FELDMAN, 2007). Therefore, it can be considered that the general order of complexity for the Deep Learning model is, at least, equivalent to $O(n^2)$.

The pseudo-code for the Deep Learning model is described in Algorithm 5:

Figure 3.5: Deep Learning algorithm flowchart.



Algorithm 5 Deep Learning

procedure DEFINE INPUT AND OUTPUT PARAMETERS

procedure TRAINING, VALIDATION AND TEST DATA SET EXTRACTED FROM THE EXPERIMENTAL RESULTS

procedure DEFINE LEARNING ALGORITHM

procedure NN TRAINING AND NETWORK OPTIMIZATION

procedure WEIGHT AND BIAS ARE SELECTED RANDOMLY

while Error goal reached = FALSE **do**

Increase iteraction

procedure CHANING OF PARAMETERS FOR TRAINING OF THE NETWORK(Number of hidden layers, Number of neurons in hidden layers, Momentum factor, Transfer function)

Updated parameters (w_{ij}, w_{jk}, b_j, b_k)

Validation of network

procedure OBTAINING THE BEST NN ARCHITECTURE AND TRAINING PARAMETERS

Network is ready for performance prediction

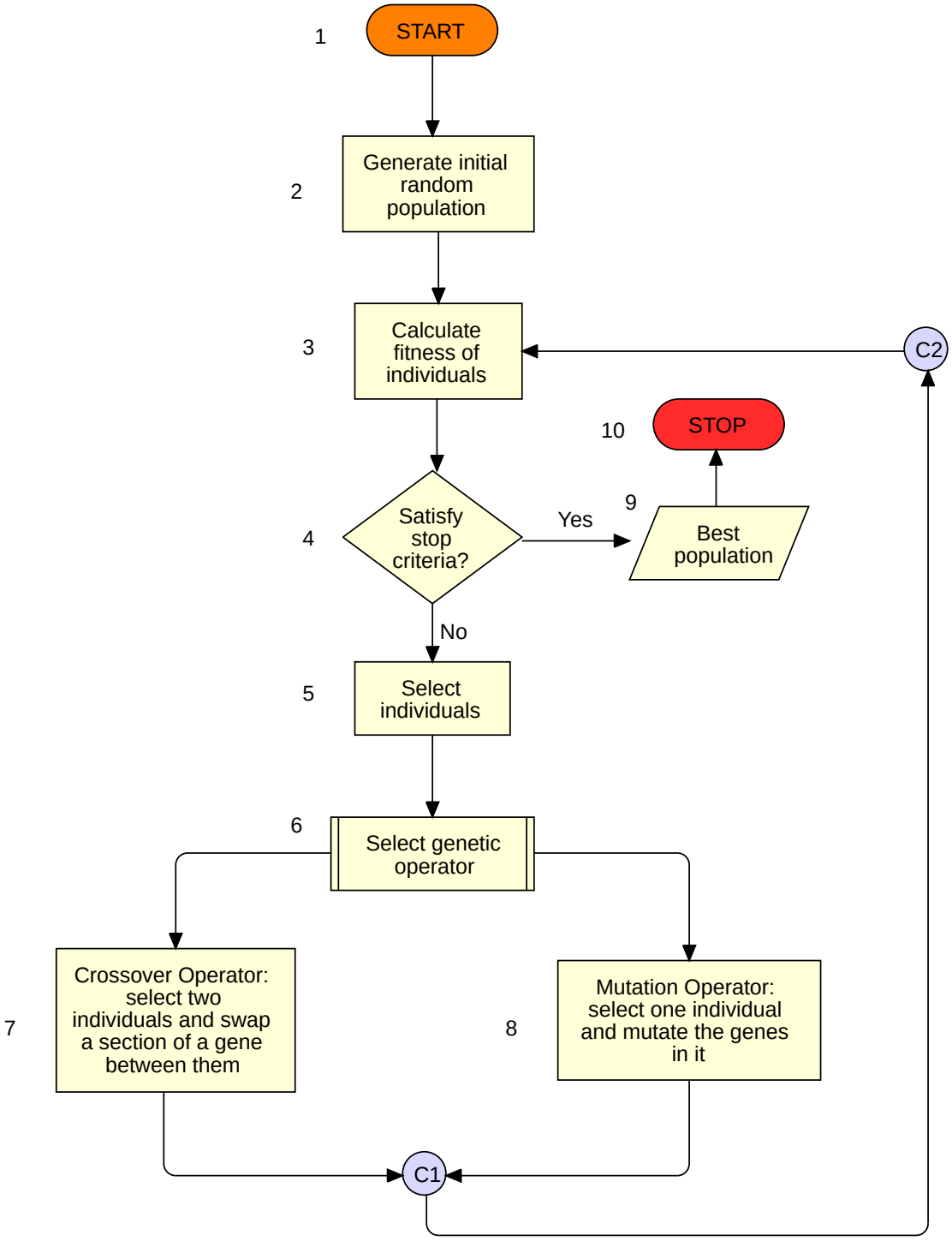
3.6 Genetic Algorithm Model

To be applied to network traffic forecasting, the GA model can be implemented considering three phases: (I) initial random population generation, (II) fitness calculation, and (III) crossover or mutation. Each phase has its own computing cost that contributes to an order of complexity. These values are presented in Table 3.1. The flowchart for this algorithm presented in Figure 3.6.

The step-by-step complexity analysis of our implementation of GA shows that steps 1, 6, and 10 have constant complexity, i.e., $O(1)$. In Step 2, the complexity is linear, $O(P)$, where P represents the size of the population. Step 3 also presents a linear complexity, $O(P * G * O(Fitness)$, where G is the number of generations from the population that will have their fitness calculated. Both steps 7 and 8 have linear complexity, $O(Pc * O(Crossover))$ and $O(Pm * O(Mutation))$, respectively. Pc is the crossover probability and Pm is the mutation probability. The crossover type is a one-point random crossover. The mutation consists of randomly changing one gene. The stop criteria for this model is reached when the generation number 1000 is obtained.

The complexity of this model is highly dependent on the number of items, the

Figure 3.6: Genetic Algorithm flowchart



number of generations, and the computation time per generation. Therefore, considering the complexity analysis of the algorithm steps, the overall result is $O(P{*}G{*}O(Fitness){*}((Pc*O(Crossover))+(Pm*O(Mutation))))$. Since the complexity of the GA depends mainly of $O(Fitness)$, $O(Crossover)$ and $O(Mutation)$, it can be considered that the general order of complexity for the Genetic Algorithm model is linear, therefore equivalent to $O(n)$, where n is the size of the population.

The pseudo-code for the Genetic Algorithm model is described in Algorithm 6:

Algorithm 6 Genetic Algorithm

procedure GENERATE INITIAL RANDOM POPULATION

while (Crossover) OR (Mutation) = TRUE **do**

Calculate fitness of individuals

if Satisfy stop criteria = TRUE **then**

Best population

else

Select individuals

end if

procedure SELECT GENETIC OPERATOR

procedure CROSSOVER(Select two individuals and swap a section of a gene between them)

procedure MUTATION(Select one individual and mutate the its genes)

Table 3.1: Complexity Analysis

Model	Function	Computing Cost	Order of complexity
MLRM	Choose Predictors	n	O(n^2)
	Run regression to find VIFs	n^2	
	Remove variable	n	
Regression Tree	Compute best split point	n	O(n^2)
	Selection of best attributes	a^2	
Fourier Series	Discrete Fourier Transform	$2n^2$	O(c^3)
	Find the optimized value for fitting	c^3	
	Least squares fitting	c^{2n}	
Q-Learning	Identify current state	k	O(n^2)
	Choose an action from the Q-Table	n	
	Update the reward in Q-Table	m	
Deep Learning	Training, validation and test dataset	n	O(n^2)
	Changing of parameters for training the network	n^2	
	Obtaining best neural network	m	
Genetic Algorithm	Generate initial random population	n	O(n)
	Calculate fitness	$n*f$	
	Crossover/Mutation	$(pc*n)$+$(pm*n)$	

In the complexity analysis table (Table 3.1), *n* represents a variable corresponding to the size of the data set used on the model. For the Regression Tree model, *a* represents the number of attributes. In the Fourier Series model, *c* represents the number of features.

In the Q-Learning model, *k* represents the constant number of identifying the current state, *n* is the number of actions in the Q-Table, and *m* is the sum of the rewards and penalties. For the Deep Learning model, *m* represents the number of datasets obtained after the error goal is achieved. Finally, in the Genetic Algorithm Model, *pc* and *pm* represent the probability of choosing either the crossover or the mutation operator, respectively.

3.7 Artificial Intelligence Models' Analysis

The discussion on the trade-off between accuracy and complexity considers the results obtained from a Matlab simulation model that takes into account three kinds of network traffic. The traffic is aggregated to evaluate the performance of the system using different artificial intelligence approaches to forecast resource occupancy. The traffic models for simulation consider two aspects: the connection arrival and the amount of traffic per connection. The traffic generation follows the specification of the WiMAX Forum methodology (JAIN, 2006) for system evaluation. In the simulations, there are considered three different kinds of traffic: HTTP, VoIP, and video streaming, each one corresponding to $60\%, 20\%,$ and 20% of the total traffic, respectively. Considering the current increase of the amount of video streaming traffic, the WiMAX Forum methodology would need to change this aspect to be at least higher than the HTTP and VoIP traffic, corresponding to 50% or 60% of the total traffic.

The first kind of traffic models best-effort HTTP packets. In this case, the transmissions comprise the main page and various objects, such as images, scripts, and other types of files. This model processes the HTTP packets in the following manner: after requesting and receiving the files, the browser parses the page to produce a readable version for the user, then the user reads the page before making a new request. To model the number and size of the objects for this simulation, we use lognormal and Pareto distributions, respectively. The reading and parsing times follow an exponential distribution.

The model for VoIP transmissions follows the Adaptive Multi-Rate (AMR) codec, which presents an *ON/OFF* behavior. To model the duration of each period, it is applied an exponential distribution with a mean of $1026ms$ for the conversations, corresponding to the *ON* period, and $1171ms$ of silence, for the *OFF* period, with the generation of a Packet Data Unit (PDU) every $20ms$.

Finally, the streaming of video clips implements the MPEG-4 encoding. All the videos have variable lengths, from $15s$ to $60s$. For the display size of the video clip,

it is considered a resolution of $176x144$. This choice of pattern creates a frame size with a mean of $2725Kbytes$ after the video clip is fully compressed. It is worth mentioning that if the simulations considered the MPEG-10 encoding (which runs over TCP transmissions), there would be eventual packet loss and some parameters for the video transmissions should be changed to deal with these losses.

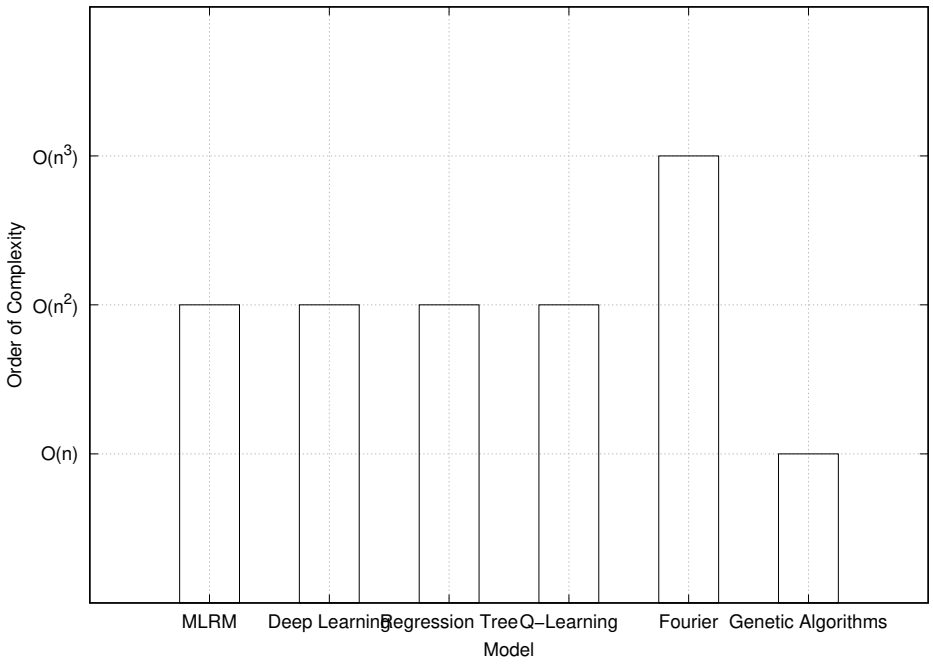
All simulations consider a frame duration of $10ms$ and a wireless channel bandwidth of $10MHz$. To evaluate the performance of the machine learning models for different traffic load scenarios, the number of active connections is varied. Considering the simulation results, the discussion focus in two main aspects: (I) the trade-off between accuracy and complexity, and (II) the number of connections supported with guaranteed QoS, which is an important metric for network operators to allocate network resource.

The analysis of the trade-off between accuracy and complexity takes into account the graphs in Figure 3.7. The first one, in Figure 3.7a, compares the order of complexity of each machine learning model. Next, in Figure 3.7b, it is shown the percentage of accuracy of each model in the evaluated scenario. In the complexity graph, a lower bar indicates a better result, since in this case, it means that less computational effort is necessary to execute the model. On the other hand, in the second graph, a higher bar is desirable and indicates better performance in terms of forecasting accuracy.

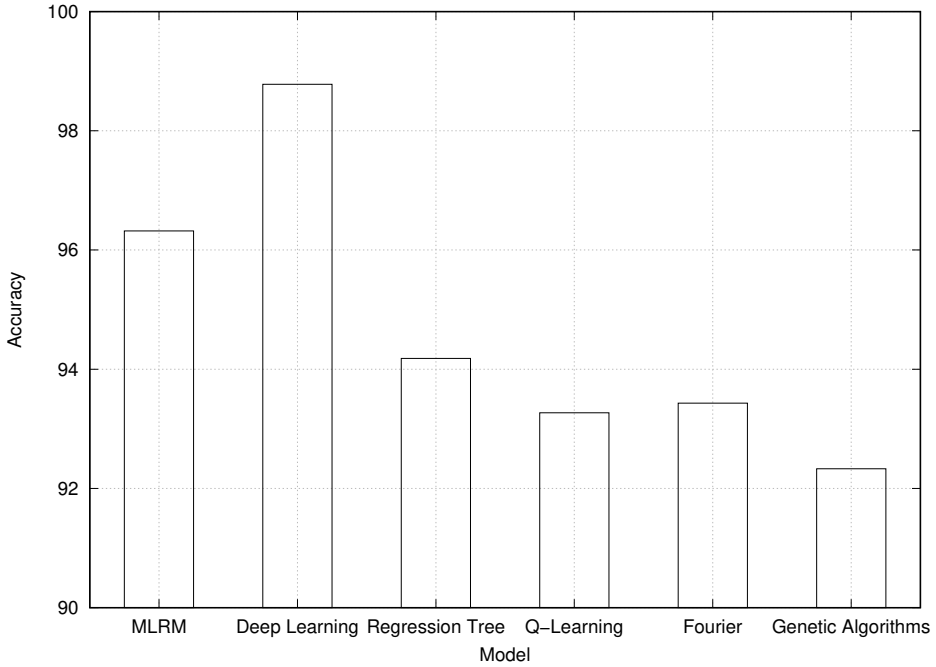
Analyzing the graphs, one can conclude that Genetic Algorithms demand relatively low computing power. On the other hand, GA presents the lowest accuracy among all models. This behavior indicates that this technique suits the real-time needs, where the response time is a key factor, and 93% is an acceptable level of accuracy. A network operator can, for example, implement genetic algorithms to forecast the behavior of a video conference transmission since this kind of traffic can deal with a certain level of packet loss. However, other parameters like delay and jitter demand more control.

Another interesting behavior regards the implementation of deep learning based on artificial neural networks. This technique is the best one in terms of accuracy, attaining a level of more than 98%. The complexity, in this case, is in the order of $O(n^2)$. However, analyzing Table 3.1, it is possible to conclude that this value is due to the training phase of the model, which occurs only sporadically. The main function of the model, which is the generation of the neural network, demands a cost in the order of $O(n)$, demanding considerably less computing power. Taking the trade-off between complexity and accuracy into consideration, for a more generic usage, a network operator should choose using deep learning as the main technique to forecast the behavior of traffic in its channels.

Figure 3.7: Complexity vs. Accuracy



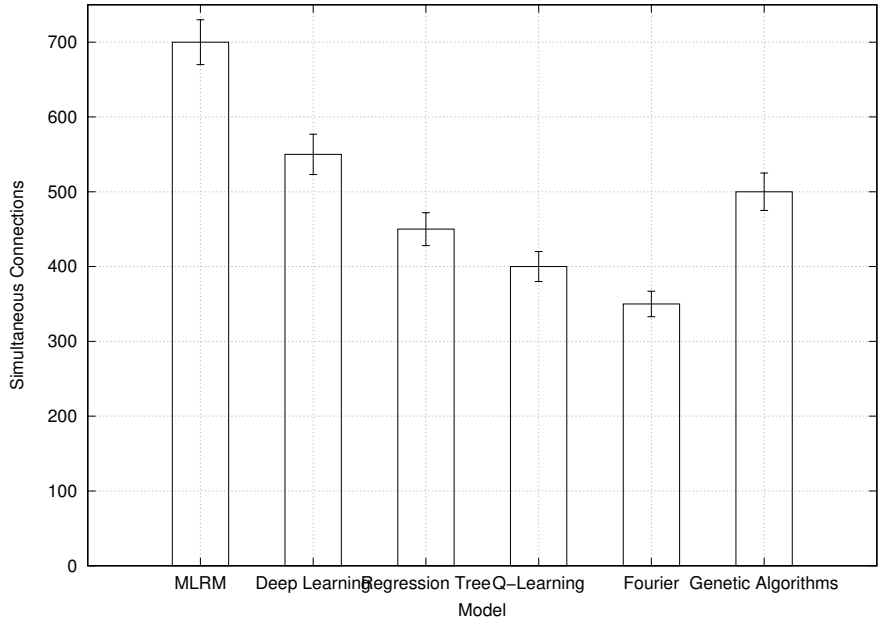
(a) Complexity Analysis



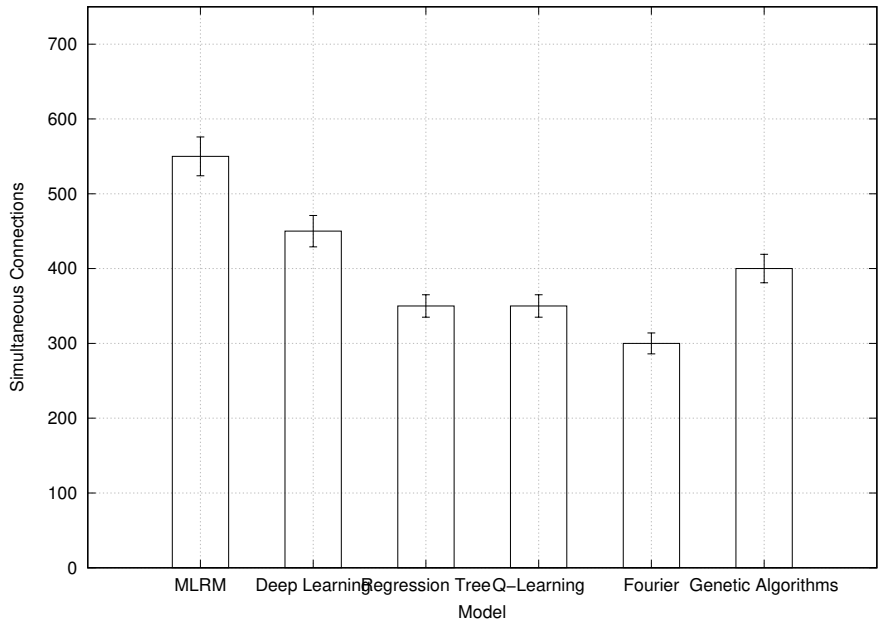
(b) Accuracy Analysis

The second group of results, presented in Figure 3.8, relates to the QoS parameters. In this case, the first graph (Fig. 3.8a) shows how many simultaneous connections are supported while the network delay is maintained under an acceptable limit, considering the requirements of VoIP and video applications. The second graph (Fig. 3.8b) considers the same kind of application requirements, but it cares for the network jitter QoS parameter. In both graphs, a higher bar is desirable, since it indicates that more simultaneous connections are supported.

Figure 3.8: QoS Parameters



(a) Delay Analysis



(b) Jitter Analysis

A visual comparison between the graphs allows one to conclude that MLRM is the model that accepts the highest amount of connections, while it also guarantees an acceptable level of both delay and jitter QoS parameters. This behavior clearly shows that the network operators - which need to provide QoS-enable transmissions to their customers - should consider the implementation of MLRM as a tool to forecast the behavior of the transmission channels since it would demand fewer investments to serve a high number of clients. Deep learning and genetic algorithms appear as other possibilities to address the transmissions of traffic generated by applications demanding more restrict QoS parameters. Considering all the results, MLRM, deep learning, and genetic algorithms are indicated to be used by network operators and the selection of one of them must rely on the transmission scenario required by each operator.

4 Q-FANET PROPOSAL AND EVALUATION

This chapter introduces and describes in details Q-FANET, an improved Q-learning based routing protocol for FANETs. Even considering that, in the context of resource sharing in networks the MLRM presents the best approach for traffic load forecasting in terms of low complexity and high levels of accuracy, the literature review demonstrates that routing protocols based on Q-learning methods are able to deal with the mobility issues and high dynamic of FANETs and provide promising results in terms of network performance.

Therefore, the proposed routing protocol is based on two Q-learning methods: Q-Noise+ and QMR. In Q-FANET, nodes use a reinforcement learning algorithm without having the knowledge of the entire network topology to make optimal routing decisions considering low-delay service.

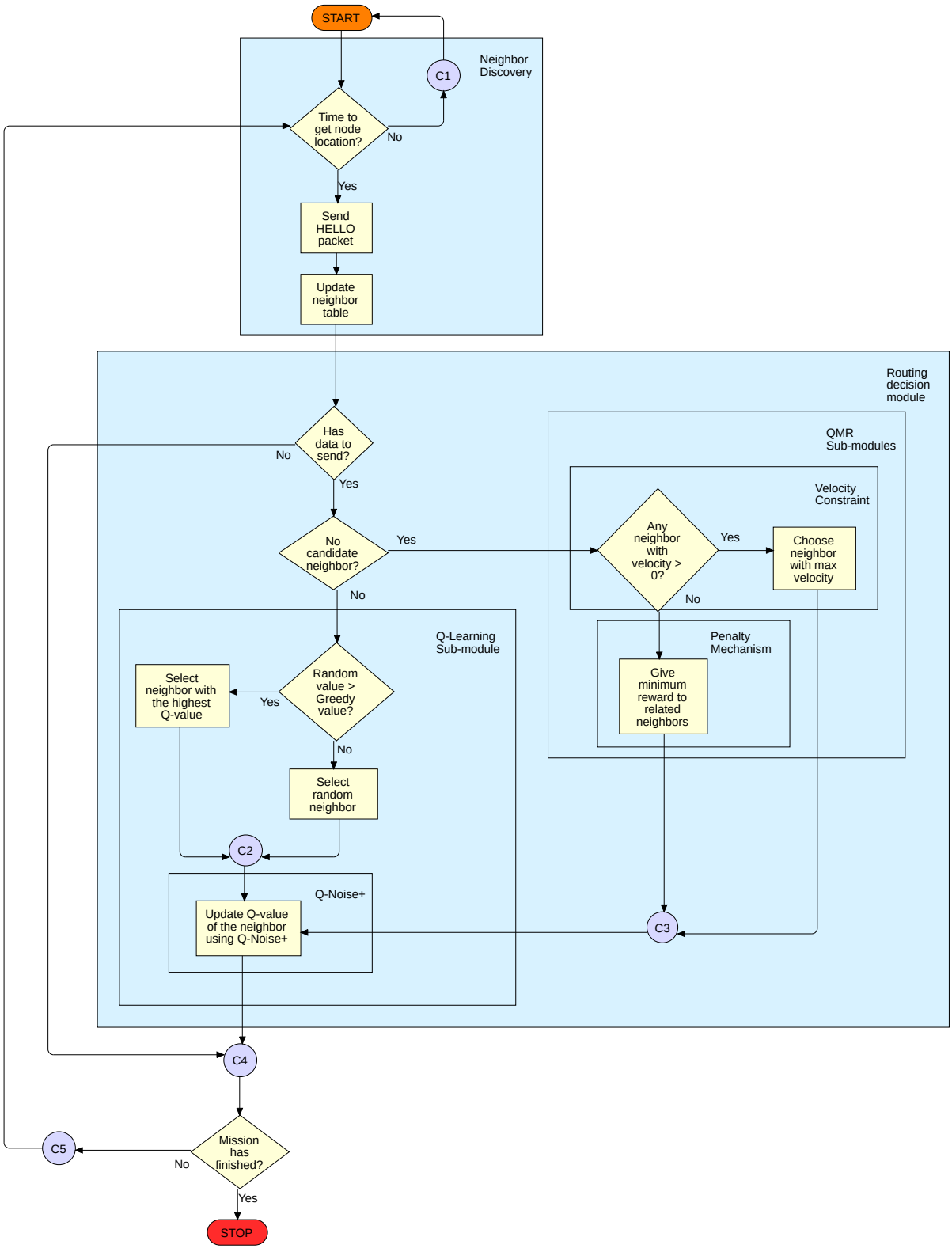
4.1 Q-FANET Design Overview

Q-FANET uses several components of QMR as basis to change and improve the algorithm, some simplifications, and appropriate adjustments to explore other techniques that are suitable with Q-learning. In QMR, nodes use a reinforcement learning algorithm without knowledge of entire networks to make optimal routing decisions considering low-delay and low-energy service. To solve the routing problem caused by the high mobility within FANETs, QMR also uses adaptive Q-learning parameters and a new exploration and exploitation mechanism to enhance the routing performance. The proposed Q-FANET consists of two major modules, the Neighbor Discovery and the Routing decision. On its turn, this last module contains a Q-learning sub-module and a QMR sub-module. This overview of Q-FANET is shown in Figure 4.1.

In Q-FANET, nodes acquire their geographic location information by GPS. Further, the routing neighbor discovery is implemented by sending HELLO packets. When a data packet generated from a source and directed to a destination, Q-learning is the key component of the routing decision. In this sense, if a routing hole problem happens (i.e., all the neighbors of a node are distant than the distance from this node to the destination), the penalty mechanism is triggered.

Algorithm 7 shows Q-FANET method and the details of the work at each phase.

Figure 4.1: Q-FANET flowchart exposing its internal modules.



Algorithm 7 Q-FANET

$\alpha := 0.6$

$\epsilon := 0.1$

$QValues := 0.5$

procedure ROUTING NEIGHBOR DISCOVERY

if $TimeToGetNodeLocation = True$ **then**

Send HELLO Packet

Update Neighbor Table

end if

procedure ROUTING DECISION

while $dataToSend = True$ **do**

if $neighborsList \neq Empty$ **then**

Q-learning

else if $neighborsWithVelocity > 0$ **then**

$routingNeighbor := neighborWithTopVelocity$

else

Penalty Mechanism

end if

end while

procedure Q-LEARNING

if $randomEpsilon > \epsilon$ **then**

Select neighbor with highest Q-value

else

Select random neighbor

end if

Update Q-value using Q-Noise+

procedure PENALTY MECHANISM

Related neighbors receive minimum reward

Update Q-value of related neighbors using Q-Noise+

4.1.1 Routing Neighbor Discovery

The Neighboring Discovery module is a control structure used to maintain the routing information updated. Q-FANET updates the location of the nodes regularly. The updating frequency is one of the parameters of the proposal, and its default value is $100ms$. In cases where a node does not inform its location within a specific expiration time of $300ms$, its neighbors remove the specific route from their routing tables.

This updating process in Q-FANET relies on the exchange of *HELLO* packets. In this case, a given network node broadcasts these packets aiming to discover its neighbors. This packet exchange happens periodically and carries the node's geographic location, energy, mobility model, queuing delay, learning rate, and Q-value. When a node receives *HELLO* packets, it uses the information to establish and maintain its neighbor table.

The idea behind this module is to keep the network ready for transmission at any time. Therefore, its functionalities are always running, despite the existence of a current transmission session. Whenever a transmission is necessary, the Neighbor Discovery module communicates with the Routing Decision, the other module of Q-FANET, to provide information regarding the best routes.

4.1.2 Routing Decision Module

The Routing Decision Model receives information about the available routes and selects the one for a given node to transmit data. To do that, it counts with two sub-modules: (I) QMR and (II) Q-learning that consider the background of existing approaches but modify them to improve the routing response.

4.1.2.1 Q-learning Sub-module

The standard Q-learning algorithm applies a reward-based approach which takes into account two criteria: (I) the transmission successful rate in the last episode, and (II) the sum of the successful rates of all past episodes. However, except for the most recent episode, the remaining episodes are considered to have the same weight in the decision process. This approach can lead to imprecise decisions, especially in scenarios where a high number of epochs are used, as for example, in military, surveillance and rescue mission application scenarios where dynamism of the situation implies in high mobility

of the nodes.

Taking the mobility problem into consideration, an extension of Q-learning, called Q-learning+, was proposed in (FAGANELLO et al., 2013). This extension of the algorithm considers a finite amount of past episodes. In this approach, the newer the epoch, the higher its weight is. A look back value (l) is defined to designate the amount of episodes to be considered.

Considering this new information, the Q-value at the instant of time $t+1$ is calculated as expressed in (4.1), where w_i represents the weight of the last l instants of time and r_i is the reward calculated based on $l+1$ actions.

$$Q_{t+1}(a_t) = (1-\alpha)\sum_{n=1}^{l}[w_{t-i}r_{t-i}](a_t) + \alpha r_t(a_t) \tag{4.1}$$

Although Q-learning+ is designed to improve the efficiency of the original approach, it keeps only considering the amount of successful transmissions to perform its decision, ignoring the propagation conditions of the channel. In order to consider also the channel conditions, another algorithm called Q-Noise can be used. This algorithm considers the transmission quality as a secondary metric, which is calculated according to the Signal-to-Interference-plus-Noise Ratio (SINR) measured in a given channel. This approach tries to avoid the selection of a channel when it is available but noisy. In this case, Q-learning+ would elect this channel as a transmission candidate, but the transmission quality should be unacceptable.

Q-Noise deals with two criteria to take the decision: (I) the learning rate considering the reward obtained in an episode T, (II) a quality criteria that considers both the SINR level of the channel, and the importance of the SINR for a given transmission, calculating a weighted reward in the last episode with respect to the trade-off between channel availability and transmission quality. These adaptations to the Q-learning+ algorithm create the Q-Noise+ approach and are expressed in (4.2).

$$Q_{t+1}(a_t) = (1-\alpha)\sum_{n=1}^{l}[w_{t-i}r_{t-i}](a_t) + \alpha r_t(a_t) + (S_w + \eta) \tag{4.2}$$

In (4.2), two new terms are included, S_w and η. S_w ($0 \leq S_w \leq 1$) represents the weight of the SINR in the calculated reward. The η value corresponds to the intervals of SINR, as described in Table 4.1. The values defined for η have been chosen to change the Q-value according to the channel conditions based on (RAPPAPORT, 1996). In good propagation conditions, η will be higher, increasing the Q-value of the channel. On the

other hand, as the channel conditions get worse, η decreases, unchanging the Q-value.

Table 4.1: Noise level correspondence.

SINR value	η
$SINR < 15dB$	0
$15dB \leq SINR < 17dB$	0.25
$17dB \leq SINR < 20dB$	0.5
$20dB \leq SINR < 25dB$	0.75
$SINR \geq 25dB$	1

Q-FANET also makes use of an ϵ-greedy policy for exploration and exploitation (WUNDER; LITTMAN; BABES, 2010). The exploration consists in searching for unknown actions (i.e. obtain new knowledge). Nevertheless, exploration in excess makes it difficult to retain some better actions. On the other hand, the exploitation is to take advantage of explored actions which may generate high rewards. Although, too much exploitation makes it difficult to select some undiscovered potential optimal actions.

In order to balance the trade-off between exploration and exploitation, the ϵ-greedy policy instructs the Q-learning Sub-module to explore by choosing a random path with probability ϵ (usually 10%) and exploit by choosing the option which offers the highest Q-value.

Q-FANET benefits from these approaches as building blocks of the Q-learning Sub-module. One crucial adaptation proposed in Q-FANET regards the reward function, which is discussed in the next sub-section.

4.1.2.2 Reward function

In Q-FANET, a data structure called R-Table (Reward Table) is proposed to store reward cells. The initial value of the reward cell values is zero. After each forwarded data from node i to node j, the R-table values are updated according to the logic expressed in (4.3):

$$R(s,a) = \begin{cases} r_{max} = 100, \text{ if link } (i,j) \text{ leads to destination} \\ r_{min} = -100, \text{ if link } (i,j) \text{ is local minimum} \\ r = 50, \text{ otherwise} \end{cases} \quad (4.3)$$

When the next hop j is the destination node, it means that the destination node

is the neighbor of node i, so the link from node i to node j gets the maximum reward value r_max. Node i is a local minimum when all its neighbors are farther away from the destination than itself, therefore the minimum reward value r_min is applied. In any other case, such as node j being a relay node in the path to the destination, the value of 50 will be given as reward.

4.1.2.3 QMR Sub-module

The QMR sub-module is responsible for the penalty mechanism and to control the constraint regarding the velocity of the nodes to support the better decision.

$Penalty\ Mechanism$:

Routing holes will increase the delay of the data packet. Q-FANET proposes a modification to the penalty mechanism of QMR, aiming to reduce the existence of routing holes. This mechanism applies to the following cases:

- **Routing hole:** if a node j discovers that all of its neighbors are further than itself from the destination, then it sends a feedback to the previous node i.
- **Not-ACK:** if a node i does not receive an ACK packet from next-hop node j.

In both scenarios, the action taken by the penalty mechanism will be that node i will give the r_min for the link i, j and update the corresponding Q-value of the link.

$Velocity\ Constraint$:

To always obtain the minimum delay between the hops, QMR defines a velocity constraint, which was here adapted to a more simplified approach in Q-FANET. The velocity is a given to a link i, j and it is defined in (4.4), observing (4.5):

$$Velocity(i,j) = \frac{d_(i,D) - d_(j,D)}{delay_(i,j)} \tag{4.4}$$

$$\begin{cases} Velocity(i,j) < 0, \text{ if } d_(j,D) > d_(i,D) \\ Velocity(i,j) > 0, \text{ if } d_(j,D) < d_(i,D) \end{cases} \tag{4.5}$$

These equations show that higher delays will lead to lower velocity constraint values. Moreover, velocities below zero indicate that the distance between node j and the destination is bigger than the one from i to the destination. Q-FANET will obtain this

velocity information during the routing neighbor discovery process, where each node will create its routing table.

4.2 Q-FANET Working Example

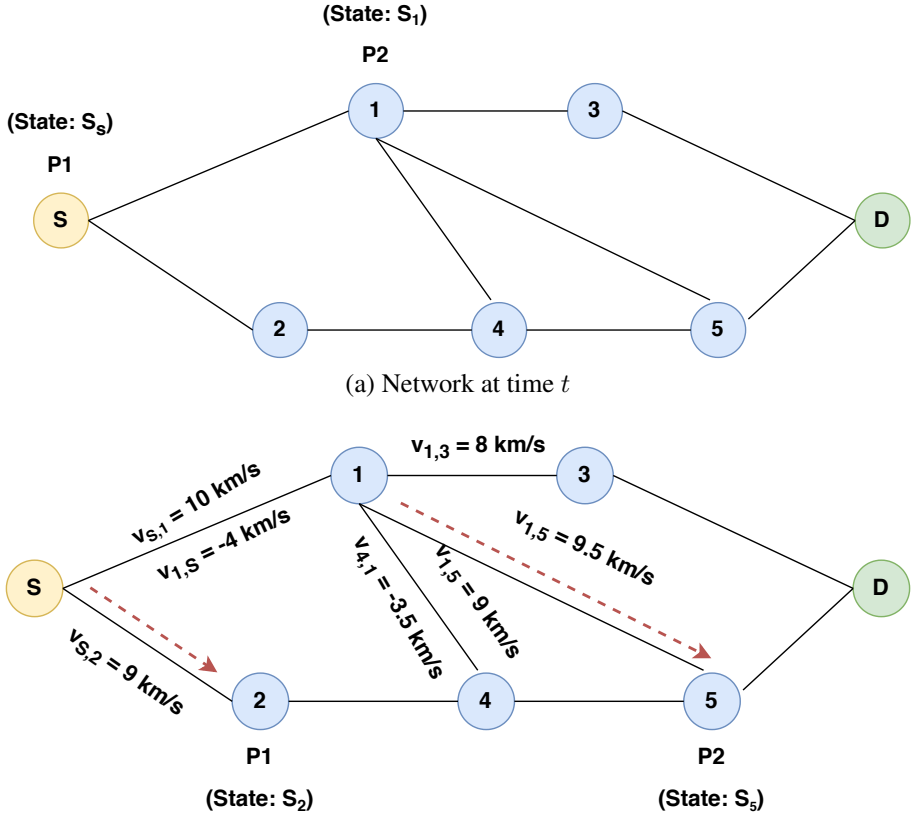
For a better understanding of how Q-FANET system works in FANETs, Figure 4.2 shows a simple network topology as an example scenario for the application of the proposed Q-FANET. In this network there are the source node (S), destination node (D) and relay nodes (1, 2, 3, 4, 5). In the scenario represented in 4.2a, it is assumed that at a current time t, there are two data packets in the network, i.e,, two agents. Packet 1 (P1) and Packet 2 (P2) are in nodes S and 1, and their states are S_S and S_1, respectively. In 4.2a it is possible to determine that the set of neighbors of node S is node 1, node 2 and of node 1 is node S, node 3, node 4, node 5. Node S and node 4 need to select one of their neighbors as the next hop to forward the packet until it reaches the destination node.

By using the Velocity Constraint sub-module from QMR, it is possible to determine the velocity of the links described in 4.2b. According to the Q-FANET algorithm, that determines that the velocity of the candidate neighbors must be greater than zero, the set of candidate neighbors is then obtained. Therefore, the set of neighbors of node S and node 1 are node 1, node 2 and node 3, node 4, node 5. Suppose that at time t, the Q-value using only the Q-learning+ approach (without considering the channel conditions), SINR and the Q-value when using the Q-Noise+ approach for the links in the network are shown in Table 4.2. It is important to note that the SINR information was obtained in a time $t-1$, since channel conditions may vary through time.

For Packet 1 (agent 1), although the Q-value of the link (S,1) is bigger than the Q-learning+ Value of the link (S,2), the link (S,2) presented a greater SINR than the link (S,1), therefore the updated Q-value by Q-Noise+ of the link (S,2) will be higher than the value of the link (S,1) and the link (S,2) will be selected to forward the packet. It is also possible to observe that the Q-learning+ Value of the link (S,2) is equal to the initial value of 0.5, which implies that now Packet 1 (agent 1) will explore a new link, meaning that previously undiscovered links might be explored.

As for Packet 2 (agent 2), the link (1,5) presents the higher Q-learning+ Value between the possible forwarding neighbor candidates. Even so, notice that this link had at time $t-1$ the highest SINR value amongst all possible links and it was the best channel to forward the packet. Therefore, the updated Q-value by Q-Noise+ of the link (1,5)

Figure 4.2: Network topology example



(a) Network at time t

(b) Network at time $t+1$

shows that this is the most suitable link to forward Packet 2. It is also possible to observe that the link (1,5) was the best forwarding link in the past (even considering the channel conditions, since its high SINR). Since Packet 1 (agent 1) is choosing the bast link in the past to forward data, Q-FANET is exploiting the knowledge that has been previously learned.

Table 4.2: Network link information

Link	(S,1)	(S,2)	(1,3)	(1,4)	(1,5)
Q-learing+ Value	0.63	0.5	0.52	0.61	0.7
SINR	16.2dB	17.5dB	17.8dB	15.5dB	20.3dB
Q-Noise+ Value	0.8	0.85	0.87	0.78	1.22

From this example, it is possible to state that the adaptations to the Q-learning algorithm provided by the Q-Noise+ method will allow that Q-FANET uses the channel quality condition to explore new links that would normally not be used in the standard routing approach.

4.3 Experiment and Results

To validate the proposed Q-FANET, experiments were performed by means of simulation that compared its performance with other existing approaches, namely, QGeo, Q-Noise+, and QMR, using an event-driven wireless networks simulator WSNet (HAMIDA, 2009). The WSNET simulator is generally used to simulate the behavior of large scale sensor networks, but it can be extended to FANET scenarios as well.

WSNet is an event-driven simulator for large scale wireless networks that has main features such as node, environment and radio medium simulation, as well as dealing with network scalability.

In WSNet (HAMIDA, 2009), the simulated nodes are built as an arbitrary assembly of blocks which represent either a hardware component, a software component or a behavior/resource of the node. There is no restriction in the number of blocks or the relation between the blocks. As an consequence, MIMO (multiple-input and multiple-output) systems or nodes with multiple radio interfaces may easily be implemented. The blocks may model the following components/ behaviors such as mobility, energy source, application, routing protocols, mac protocols, applications, radio interface and antenna settings. An example of node architecture is depicted in the Figure 4.3:

In a simulation, the nodes birth time can be specified and does not necessarily correspond to the the start of the simulation. Nodes can also die during a simulation due to external physical environments or to a lack of energy. Finally, nodes can read physical measures in their environment and impact on these measures. This feature gives the opportunity to simulate sensor-actuator networks.

For the simulation scenario, 25 nodes (representing the UAVs) are randomly distributed in an area of 500m $\times$ 500m, with the source node being randomly selected and coordinates of destination node set to (500, 500). The source is set to transmit a periodic flow of data packets whose data interval is set differently for comparison. The parameters for the scenario are shown in Table 4.3.

In the simulations, all mobile nodes move according to the Random Waypoint Mobility Model (CAMP; BOLENG; DAVIES, 2002), and follow the study of (Orfanus; de Freitas, 2014). A mobile node moves from its current location to a new random location by choosing a direction and speed (in *[minspeed, maxspeed]*). In this mobility model, after the mobile node moves to the new destination, it pauses for a specified period, and then starts to move to another new location, set to 0 for all the simulations. Also, this

Figure 4.3: WSNet simulator Node Architecture.

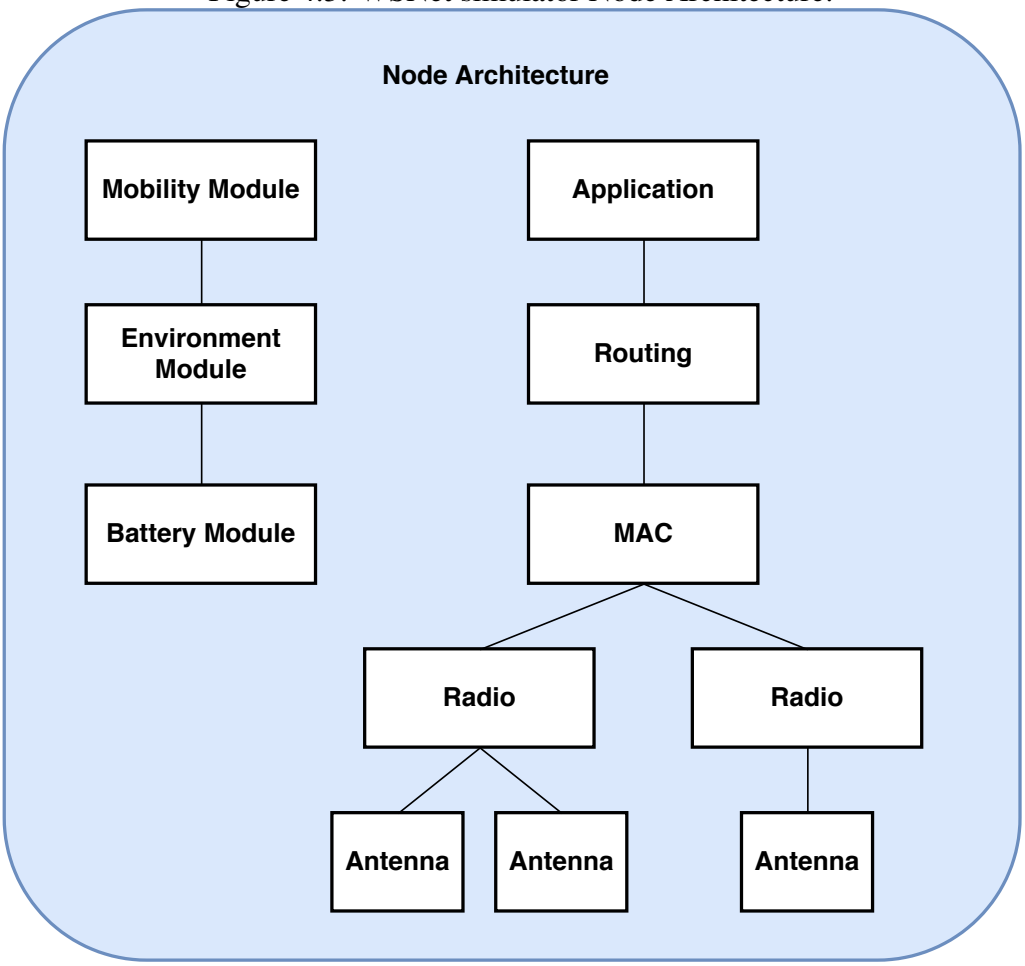


Table 4.3: Simulation Parameters Setup.

Parameters	Settings
Area size	500m $\times$ 500m
Number of nodes	25
Radio propagation	propagation range, rang = 180m
Interferences	interferences orthogonal
Modulation	modulation bpsk
Antenna	antenna omnidirectionnal
Battery	energy linear
HELLO Interval	100ms
Expire Time	300ms
Initial Q-values	0.5
minspeed	0 m/s
maxspeed	15 m/s
Data packet	127 Bytes
Look back for Q-Noise+ (l)	10
SINR weight	0.7
w	$0 < w < 1$
α	0.6
ϵ	0.1

approach considers a lookback of ten episodes to parametrize Q-Noise+. The simulation tool generates random weights for each episode at the beginning of the simulation. The simulation tool generates random weights for each episode at the beginning of the simulation. Different from the original Q-Noise+ approach, the most recent episode does not receive the higher weight. Therefore, in this manner, Q-FANET can assign random weights to each episode, not prioritizing a specific one.

Two sets of simulations were performed for each algorithm. In the first one, all 25 nodes work correctly, while the second one simulates the existence of ten faulty nodes. This last set evaluates the protocols' capability to overcome the unfavorable conditions of a network with faulty nodes. The data transmission intervals vary between $10ms$ an $50ms$, with an increasing pace of 10 ms (LIU et al., 2020). This approach allows comparing the results of QMR and Q-FANET in the same conditions. For each transmission interval, initially 100 simulations runs were performed, with the final values been represented as the results' average. Then, the confidence interval was calculated using the t-student distribution and performed additional runs, if necessary, to reach a confidence interval of 95%. The evaluation considers the following metrics.

- **Maximum end-to-end delay:** The maximum delay of the data packet from the source node to the destination node.
- **Jitter:** The average delay of the data packet from the source node to the destination node.
- **Packet delivery ratio:** The ratio of the number of data packets received by the destination node to the number of data packets transmitted by the source node.

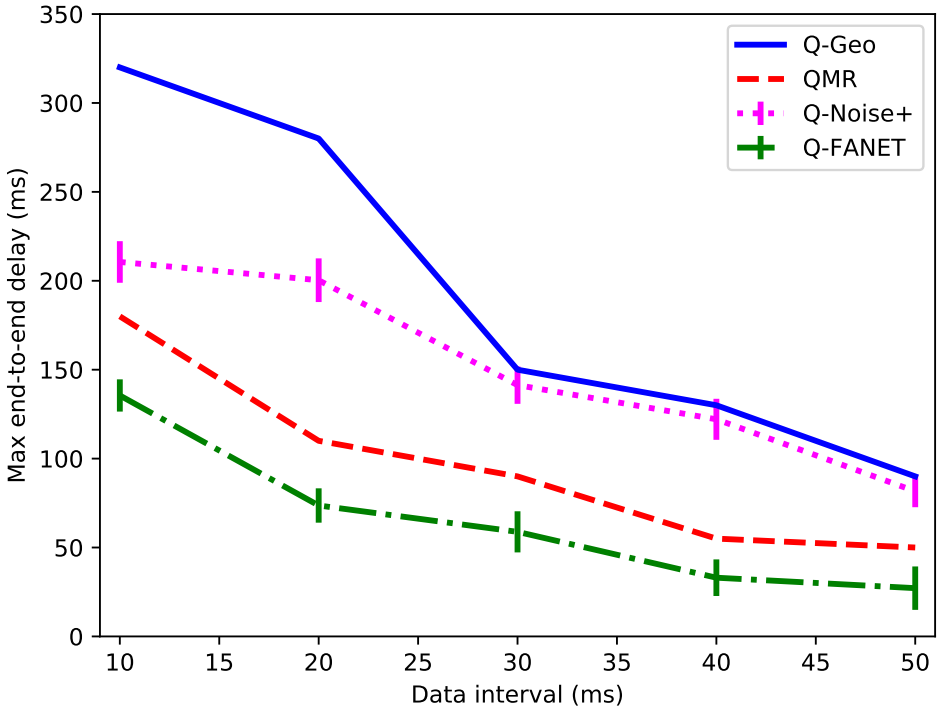
As most of the reinforcement learning techniques, it is recommended that Q-learning uses a training time or number of cases to compose a training set, which would be executed until a convergence of results was obtained. Nevertheless, since 100 simulations are executed for each time interval - with the final average representing the result - the training aspect of this algorithm is not used in the simulation.

This simulation parameters were chosen according to the ones stated in (LIU et al., 2020) in order to compare the results obtained with Q-Noise+ and Q-FANET in a equivalent test environment.

4.3.1 Tests for a scenario without faulty nodes

In this simulation, the source node sends one thousand data packets at different intervals, as explained above. In Figures 4.4 to 4.6, the performance of Q-FANET for the different tested data intervals is compared with QGeo, Q-Noise+ and QMR considering the selected performance metrics.

Figure 4.4: Max end-to-end delay for the first scenario with all nodes working properly.



In Figures 4.4 and 4.5 it is possible to observe that Q-FANET presents a lower max end-to-end delay, as well as a lower jitter than Q-Geo, Q-Noise+ and QMR. There are two reasons for this better performance: the first is the use of the velocity constraint adapted from QMR, and the second is the channel selection from Q-Noise+. The velocity constraint always select the routing path with the lowest delay from source to destination. Furthermore, the use of Q-Noise+ features gives a higher weight to the channels with a good SINR value. Exploring advantageous features of both algorithms, the new proposal surpass them two. Besides, the standard deviation error bar shows that the results of Q-Noise+ and Q-FANET are inside an acceptable error margin.

Figure 4.6 shows that Q-FANET increases the packet delivery ratio compared to the other algorithms. This improvement mainly occurs because the weighted last ten episodes change the learning rate and discount factor in the Q-learning sub-module of

Figure 4.5: Jitter for the first scenario with all nodes working properly.

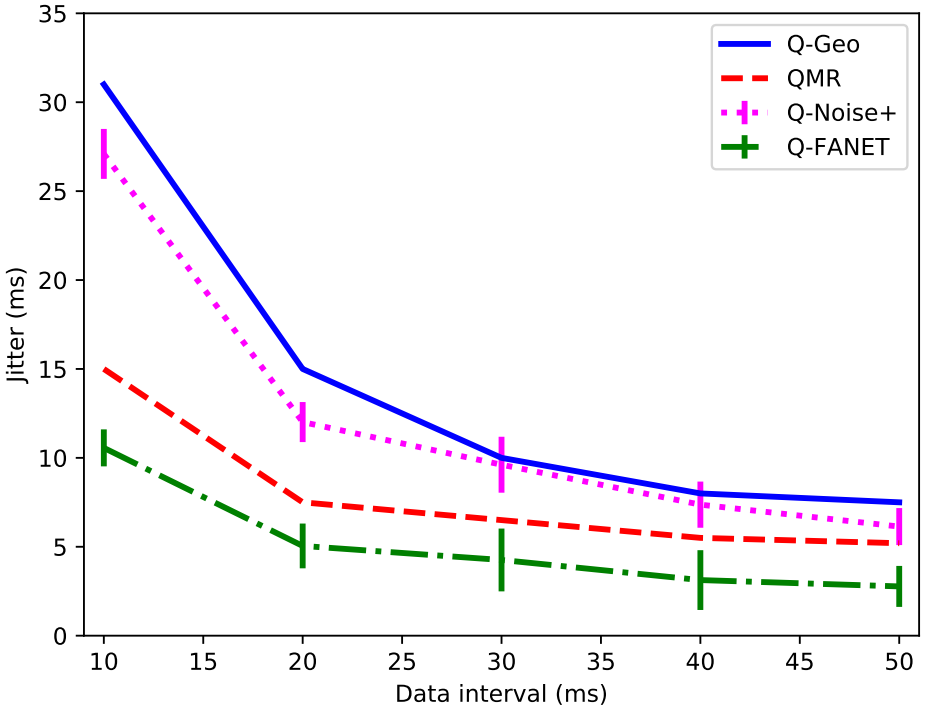
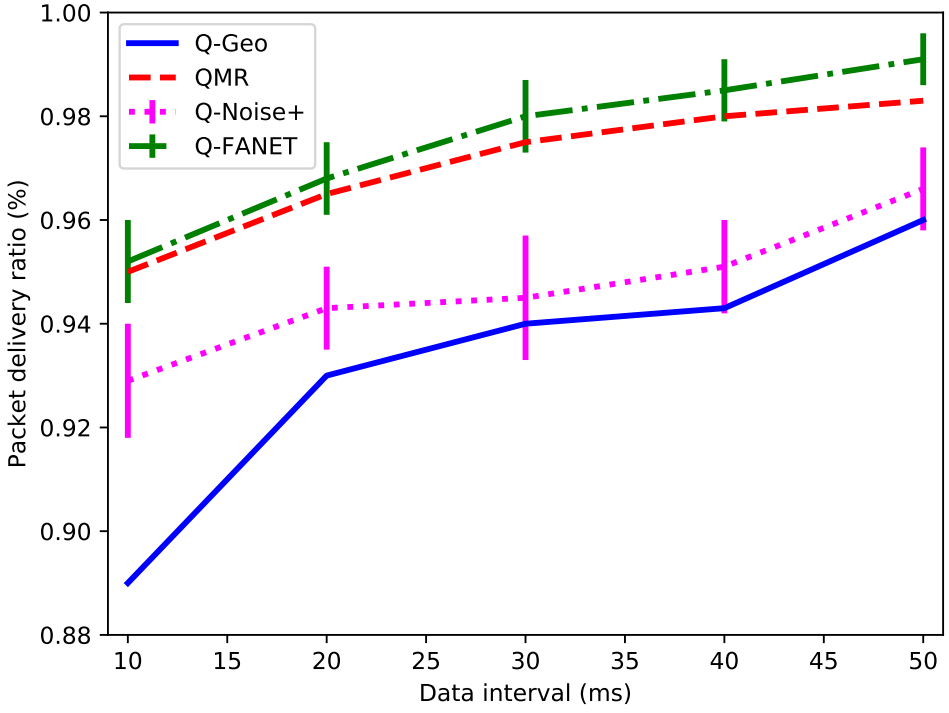


Figure 4.6: Packet delivery ratio for the first scenario with all nodes working properly.

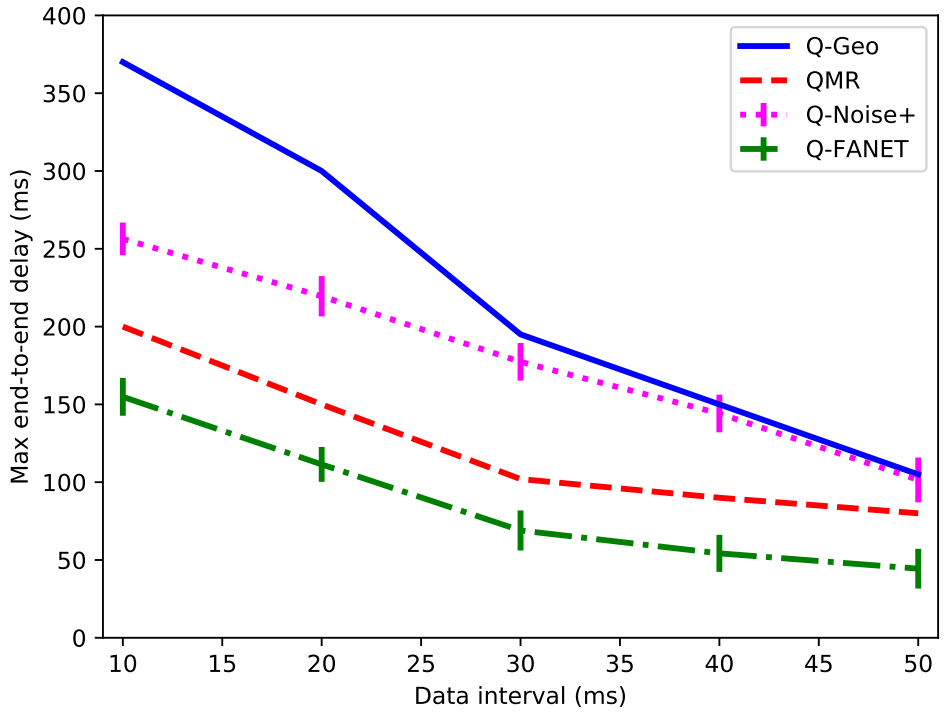


Q-FANET. The SINR-based selection of the best channel, in the QMR sub-module, also collaborates for this result.

4.3.2 Tests for a scenario with faulty nodes

In this second set of simulations, 10 out of the 25 nodes were randomly selected to stop working by powering them off at 1 second after starting the simulation. The performance of the proposed Q-FANET and the existing Q-Geo, Q-Noise+ and QMR under different data interval are compared.

Figure 4.7: Max end-to-end delay for the scenario with faulty relay nodes.



From Figures 4.7 to 4.9, it is still possible to observe that Q-FANET presents a better performance in all the evaluation metrics. As observed in Figures 4.7 and 4.8, Q-Geo, and Q-Noise+ are greatly affected by the presence of the faulty nodes while both QMR and Q-FANET present a better adaptive behavior, and can overcome the problem by selecting better routes to transmit. The packet delivery is also more affected in Q-Geo and Q-Noise+ compared to Q-FANET and QMR. The difference between the latter ones is smaller in this situation with faulty nodes, but still significant, particularly considering applications such as video streaming, which are very sensitive to the Quality of Service

Figure 4.8: Jitter for a scenario with faulty relay nodes

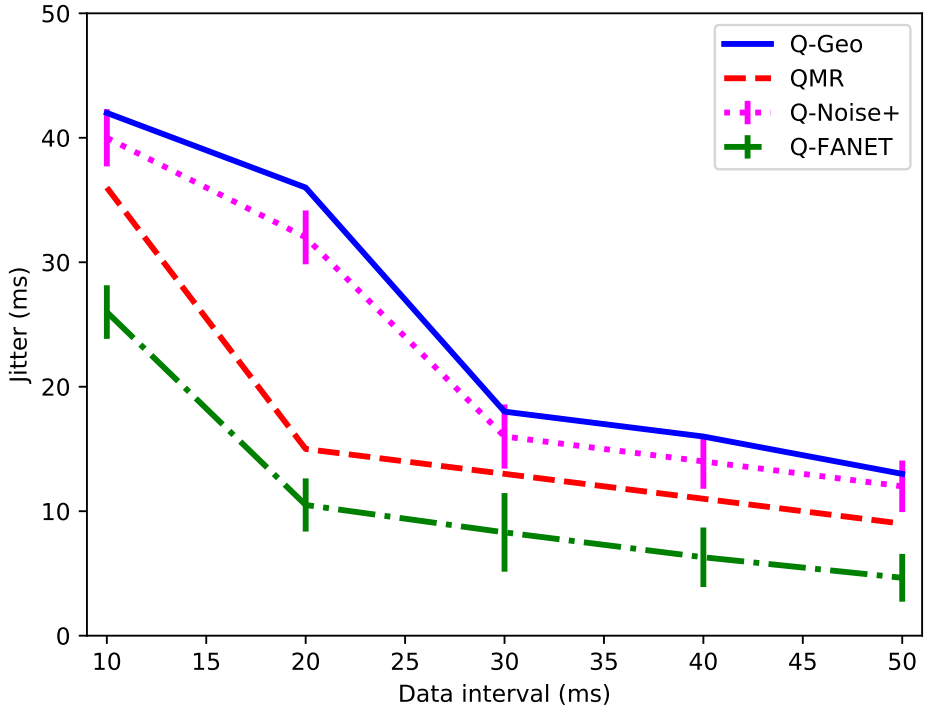
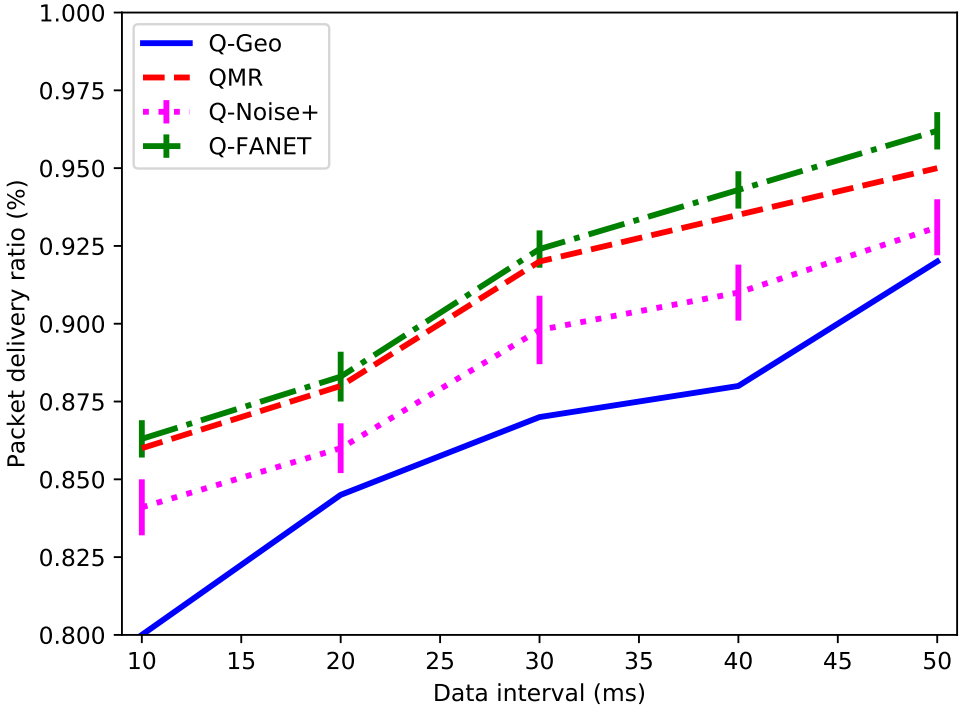


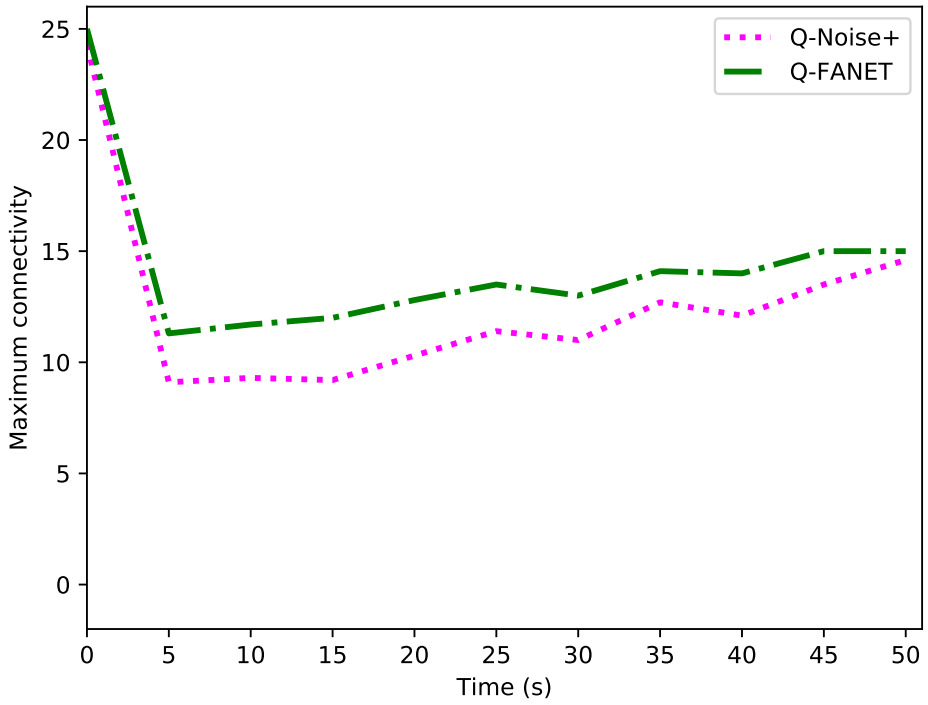
Figure 4.9: Packet delivery ratio for the scenario with faulty relay nodes.



(QoS) degradation, as discussed by (ZACARIAS et al., 2018).

It is also worth mentioning the behavior of the network after 1 s when the 10 nodes are turned off in regards to its level of connectivity. As shown in (ORFANUS; FREITAS; ELIASSEN, 2016) this is an important measure to analyse when performing experiments and simulations in FANETs. Since Q-FANET algorithm needs to readjust the neighbor table of each node, the level of connectivity in the network is changed. Nevertheless, Figure 4.10 shows that in comparison with Q-Noise+, Q-FANET presents a better performance in this aspect, converging faster to establish the connections between the remaining 15 nodes in the test scenario where the data transmission interval is of 10 ms and the simulation runtime has an average of 50 s.

Figure 4.10: Maximum connectivity in the scenario with faulty nodes and data interval of 10 ms.



4.3.3 Discussing the improvements provided by Q-FANET

Figures 4.11 and 4.12 show that Q-FANET can enhance routing performance and presents a significant improvement over QMR, which is the best among the other three protocols. Q-FANET presents an increasing improvement in terms of maximum delay and jitter over QMR as the time intervals between the data transmission increase, achieving

between 45.71% and 46.75%, respectively, of better performance than QMR for the 50 ms data sending time interval. Even in the scenario with the faulty nodes, Q-FANET shows improvements of 44.49% and 48.28% for the maximum delay and jitter.

It is worth mentioning that Q-FANET has a minor improvement over QMR in terms of packet delivery ratio, showing a better performance of 0.81% and 1.26%, for the scenario with all the nodes working and the scenario with the faulty nodes, respectively. However, considering the possible video streaming applications, this small improvement can represent a significant result for the final user, considering Quality of Experince (QoE) evaluations, as discussed in (ZHAO et al., 2019). Analyzing these results in light of the discussions provided by (ZHAO et al., 2019) and (ZACARIAS et al., 2018), it is possible to assess that the improvements provided by Q-FANET can benefit end-users of video streaming applications through lowering the number and length of stalls in the videos, as these QoE metrics are directly affected by the QoS delay and jitter metrics.

Figure 4.11: Improvement percentage of Q-FANET over QMR.

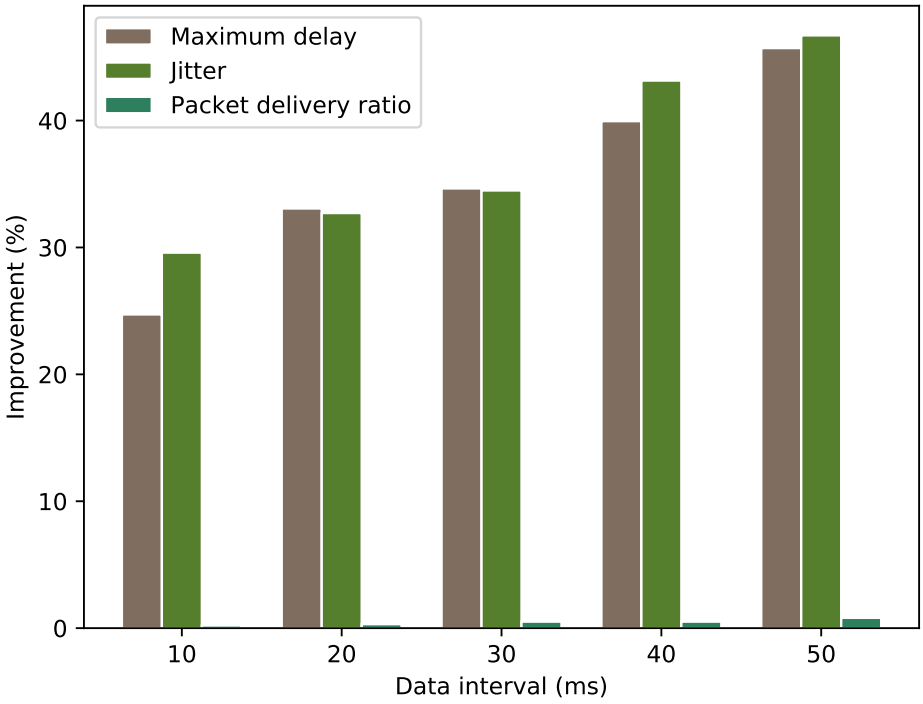
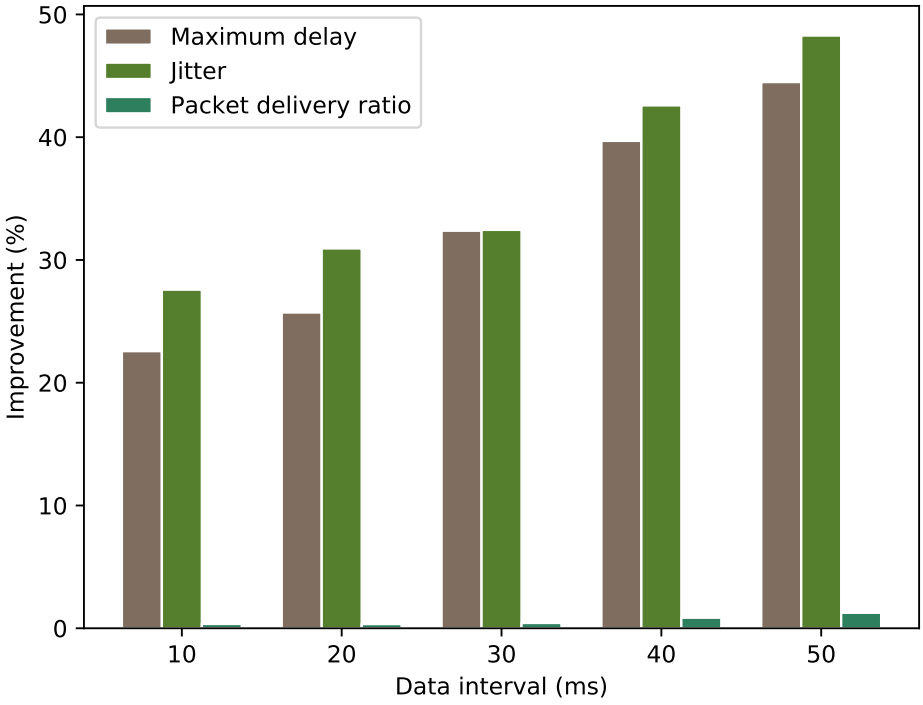


Figure 4.12: Improvement percentage of Q-FANET over QMR in the faulty nodes simulation scenario.



5 CONCLUSION

This work addressed two important issues in emergent networks: (I) the trade-off between accuracy and complexity of several traffic load forecasting models and (II) an improvement to the Q-learning method for routing data information in FANETs. In the context of the Artificial Intelligence algorithm models used in resource sharing, the review of the main concepts of each analyzed heuristic provided a comprehensive complexity analysis that can be used as background knowledge to support decisions about which model is more appropriate to network operators.

The analysis results have shown that the MLRM provides the best trade-off between high accuracy and low complexity to forecast load traffic in emergent networks such Internet of Things (IoT) and 5G applications. The results obtained also demonstrate that Deep Learning and Genetic Algorithms present suitable options to be used by networks operators. The selection of one of them will often vary depending of the QoS requirements for a specific transmission scenario.

In this sense, directions for future work include the execution of the machine learning models in realistic testbeds. A few existing testbeds are compatible with the approach proposed by this work. The most important one is the Cognitive Radio Trial Environment (CORE) from VTT Technical Research Center of Finland (Matinmikko et al., 2013). Moreover, after validating the models in a testbed, the goal is to implement the artificial intelligence models in a real network operator.

Nevertheless, it would be interesting to analyse the complexity and performance of Autoregressive integrated moving average models (ARIMA) (ZHANG, 2003) in the forecast of network traffic load. ARIMA is one of the popular linear models in time series forecasting and is quite flexible in that they can represent several different types of time series.

Also, this work addressed the routing protocols for FANETs and its main challenges, proposing Q-FANET, an improved Q-Learning based routing protocol that is able to deal with the high dynamic and mobility of this type of network. The proposed approach has brought together the main techniques and elements used in two different routing protocols that make use of Reinforcement Learning: QMR and Q-Noise+. By combining and adapting elements of these base protocols, the goal was to propose a protocol that better suites for the dynamic behavior of FANETs, improving the network reliability and performance.

The proposed routing solution was evaluated and compared with QMR, Q-Noise+ and Q-Geo protocols. This evaluation was performed in a scenario in which all the nodes were active and running in normal conditions and in another scenario in which there were a number of selected faulty nodes. The obtained results from the simulations have shown that Q-FANET possesses significant lower maximum end-to-end delay and jitter than the competitors in both scenarios. There was also a minor increase in the packet delivery ratio.

For Q-FANET, future works can address other issues regarding energy consumption, which is an important concern regarding small UAVs with constrained energy resources. Online inner parameters adaptation is a possible direction to further enhance the proposed solution. Moreover, particular movement patterns, use of multiple IA algorithms in the same experiment, simulated LTE network environments and other wireless ad-hoc network simulators such as NS3 (RILEY; HENDERSON, 2010) can also be used in additional exploratory experiments.

REFERENCES

AL-TURJMAN, F. et al. Mobile traffic modelling for wireless multimedia sensor networks in iot. v. 112, 09 2017.

BANI, M.; ALHUDA", . **International Journal of Advanced Computer Science and Applications**, v. 7, n. 6, p. 162–168, 2016. ISSN 21565570. Available from Internet: <https://goo.gl/Tu95TD>.

BARRADO, C. et al. Wildfire monitoring using a mixed air-ground mobile network. **IEEE Pervasive Computing**, v. 9, p. 24–32, 2010.

BENGIO, Y.; COURVILLE, A. C.; VINCENT, P. Representation learning: A review and new perspectives. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 35, p. 1798–1828, 2013.

BIOMO, J.-D. M. M.; KUNZ, T.; ST-HILAIRE, M. Routing in unmanned aerial ad hoc networks: A recovery strategy for greedy geographic forwarding failure. **2014 IEEE Wireless Communications and Networking Conference (WCNC)**, p. 2236–2241, 2014.

BITHAS, P. S. et al. A survey on machine-learning techniques for uav-based communications. **Sensors (Basel, Switzerland)**, v. 19, 2019.

BOUKERCHE, A. et al. Routing protocols in ad hoc networks: A survey. **Comput. Networks**, v. 55, p. 3032–3080, 2011.

CAMP, T.; BOLENG, J.; DAVIES, V. A survey of mobility models for ad hoc network research. **Wireless Communications and Mobile Computing**, v. 2, p. 483–502, 2002.

CHEEPATI, K. R.; PRASAD, T. N. Performance comparison of short term load forecasting techniques. **International Journal of Grid and Distributed Computing**, v. 9, n. 4, p. 287–302, 2016.

CHENG, C. M. et al. Maximizing throughput of UAV-relaying networks with the load-carry-and-deliver paradigm. **IEEE Wireless Communications and Networking Conference, WCNC**, p. 4420–4427, 2007. ISSN 15253511.

FAGANELLO, L. R. et al. Improving reinforcement learning algorithms for dynamic spectrum allocation in cognitive sensor networks. **2013 IEEE Wireless Communications and Networking Conference (WCNC)**, p. 35–40, 2013.

FELDMAN, V. Hardness of proper learning (1988 ; pitt , valiant). In: . [S.l.: s.n.], 2007.

FU, Y. et al. Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 43, n. 6, p. 1451–1465, 2013. ISSN 10834427.

HABIB, S.; SALEEM, S.; SAQIB, K. M. Review on manet routing protocols and challenges. **2013 IEEE Student Conference on Research and Developement**, p. 529–533, 2013.

HAMIDA, E. B. **WSNet: An event-driven simulator for large scale wireless sensor networks**. [S.l.], 2009.

HE, C. et al. A q-learning based cross-layer transmission protocol for manets. **2019 IEEE International Conferences on Ubiquitous Computing Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)**, p. 580–585, 2019.

HEIMFARTH, T.; ARAUJO, J. P. de; GIACOMIN, J. C. Unmanned aerial vehicle as data mule for connecting disjoint segments of wireless sensor network with unbalanced traffic. **2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing**, p. 246–252, 2014.

JAILTON, J. et al. Relay positioning strategy for traffic data collection of multiple unmanned aerial vehicles using hybrid optimization systems: A fanet-based case study. **Wireless Communications and Mobile Computing**, v. 2017, p. 1–11, 12 2017.

JAIN, R. Wimax system modeling methodology. In: . [S.l.: s.n.], 2006.

Jean-Daniel Medjo Me Biomo, Thomas Kunz, M. S.-H. Routing in Unmanned Aerial Ad hoc Networks: A Recovery Strategy for Greedy Geographic Forwarding Failure. v. 3, p. 2236–2241, 2014. ISSN 15253511. Available from Internet: <https://goo.gl/aBgvgQ>.

JIANG, J.; HAN, G. Routing Protocols for Unmanned Aerial Vehicles. **IEEE Communications Magazine**, v. 56, p. 58–63, 2018.

JOHNSON, D. B.; MALTZ, D. A. Dynamic source routing in ad hoc wireless networks. In: . [S.l.: s.n.], 1996.

JUNG, W.-S.; YIM, J.; KO, Y.-B. Qgeo: Q-learning-based geographic ad hoc routing protocol for unmanned robotic networks. **IEEE Communications Letters**, v. 21, p. 2258–2261, 2017.

Kampouropoulos, K. et al. Multiobjective optimization of multi-carrier energy system using a combination of anfis and genetic algorithms. **IEEE Transactions on Smart Grid**, v. 9, n. 3, p. 2276–2283, May 2018. ISSN 1949-3053.

KARP, B.; KUNG, H. T. Gpsr: greedy perimeter stateless routing for wireless networks. In: **MobiCom**. [S.l.: s.n.], 2000.

KESHENG, L.; JUN, Z.; TAO, Z. The clustering algorithm of uav networking in near-space. **2008 8th International Symposium on Antennas, Propagation and EM Theory**, p. 1550–1553, 2008.

KHATIB, T. et al. Assessment of artificial neural networks for hourly solar radiation prediction. **International journal of Photoenergy**, Hindawi Publishing Corporation, v. 2012, 2012.

KLIKS, A. et al. Perspectives for resource sharing in 5g networks. **Telecommunication Systems**, v. 68, p. 605 – 619, 2018. ISSN 1572-9451. Available from Internet: <https://link.springer.com/article/10.1007/s11235-017-0411-3#citeas>.

KO, J.; MAHAJAN, A.; SENGUPTA, R. A network-centric UAV organization for search and pursuit operations. **IEEE Aerospace Conference Proceedings**, v. 6, p. 2697–2713, 2002. ISSN 1095323X.

KUNST, R. et al. Improving network resources allocation in smart cities video surveillance. **Computer Networks**, v. 134, p. 228 – 244, 2018. ISSN 1389-1286. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S1389128618300513>.

LI, R. et al. Qgrid: Q-learning based routing protocol for vehicular ad hoc networks. **2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)**, p. 1–8, 2014.

LIN, L. et al. A novel geographic position mobility oriented routing strategy for uavs. **Journal of Computational Information Systems**, v. 8, p. 709–716, 02 2012.

LIU, J. et al. Qmr: Q-learning based multi-objective optimization routing protocol for flying ad hoc networks. **Comput. Commun.**, v. 150, p. 304–316, 2020.

LIU, Y.; LEE, J. Y. B. An empirical study of throughput prediction in mobile data networks. **2015 IEEE Global Communications Conference (GLOBECOM)**, p. 1–6, 2014.

LYE, K. W.; YUAN, X. M.; CAI, T. X. A spectrum comparison method for demand forecasting. In: . [S.l.: s.n.], 2009.

LYU, N. et al. Qngpsr: A q-network enhanced geographic ad-hoc routing protocol based on gpsr. **2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)**, 2018.

MAISTRENKO, V. A.; ALEXEY, L. V.; DANIL, V. A. Experimental estimate of using the ant colony optimization algorithm to solve the routing problem in fanet. **2016 International Siberian Conference on Control and Communications (SIBCON)**, p. 1–10, 2016.

MANATHARA, J. G.; SUJIT, P.; BEARD, R. Multiple uav coalitions for a search and prosecute mission. **Journal of Intelligent Robotic Systems**, v. 62, p. 125–158, 2011.

Matinmikko, M. et al. Cognitive radio trial environment: First live authorized shared access-based spectrum-sharing demonstration. **IEEE Vehicular Technology Magazine**, v. 8, n. 3, p. 30–37, 2013.

MAZA, I. et al. Experimental results in multi-uav coordination for disaster management and civil security applications. **Journal of Intelligent Robotic Systems**, v. 61, p. 563–585, 2011.

MERWADAY, A.; GÜVENç, I. Uav assisted heterogeneous networks for public safety communications. **2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)**, p. 329–334, 2015.

NAIMI, S. et al. Anticipation of etx metric to manage mobility in ad hoc wireless networks. In: **ADHOC-NOW**. [S.l.: s.n.], 2014.

NOULAS, A. et al. Mining user mobility features for next place prediction in location-based services. **2012 IEEE 12th International Conference on Data Mining**, p. 1038–1043, 2012.

Orfanus, D.; de Freitas, E. P. Comparison of uav-based reconnaissance systems performance using realistic mobility models. In: **2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)**. [S.l.: s.n.], 2014. p. 248–253.

ORFANUS, D.; FREITAS, E. P.; ELIASSEN, F. Self-organization as a supporting paradigm for military uav relay networks. **IEEE Communications Letters**, v. 20, p. 804–807, 2016.

PALMA, D. et al. Unmanned aerial vehicles as data mules: An experimental assessment. **IEEE Access**, v. 5, p. 24716–24726, 2017.

PAPADOPOULI, M.; RAFTOPOULOS, E.; SHEN, H. Evaluation of short-term traffic forecasting algorithms in wireless networks. **2006 2nd Conference on Next Generation Internet Design and Engineering, 2006. NGI '06.**, p. 8 pp.–109, 2006.

PARK, V.; CORSON, S. **Temporally-Ordered Routing Algorithm (TORA) Version 1**. IETFMANET working group. Available from Internet: <https: //tools.ietf.org/html/draft-ietf-manet-tora-spec-00>.

QI, L.; YAN, S.; PENG, M. Modeling and performance analysis in uav assisted ultra dense networks. **2018 IEEE International Conference on Communications Workshops (ICC Workshops)**, p. 1–6, 2018.

RAPPAPORT, T. S. Wireless communications - principles and practice. In: . [S.l.: s.n.], 1996.

RAW, D. K. L. R. S.; DAS, S. An analytical approach to position-based routing protocol for vehicular ad hoc network. p. 9, 2012.

RILEY, G. F.; HENDERSON, T. R. The ns-3 network simulator. In: **Modeling and Tools for Network Simulation**. [S.l.: s.n.], 2010.

ROSATI, S. et al. Speed-aware routing for uav ad-hoc networks. **2013 IEEE Globecom Workshops (GC Wkshps)**, p. 1367–1373, 2013.

SAHINGOZ, O. K. Mobile networking with uavs: Opportunities and challenges. **2013 International Conference on Unmanned Aircraft Systems (ICUAS)**, p. 933–941, 2013.

SERHANI, A.; NAJA, N.; JAMALI, A. Qlar: A q-learning based adaptive routing for manets. **2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)**, p. 1–7, 2016.

Srinivas, M.; Patnaik, L. M. Genetic algorithms: a survey. **Computer**, v. 27, n. 6, p. 17–26, June 1994. ISSN 0018-9162.

STROBL, C.; MALLEY, J.; TUTZ, G. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. **Psychological methods**, v. 14 4, p. 323–48, 2009.

SUN, Z. et al. Bordersense: Border patrol through advanced wireless sensor networks. **Ad Hoc Networks**, v. 9, p. 468–477, 2011.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. [S.l.]: MIT press, 2018.

VASILIEV, D. S.; ABILOV, A.; KHVORENKOV, V. V. Peer selection algorithm in flying ad hoc networks. **2016 International Siberian Conference on Control and Communications (SIBCON)**, p. 1–4, 2016.

WATKINS, C. J.; DAYAN, P. Q-learning. **Machine learning**, Springer, v. 8, n. 3-4, p. 279–292, 1992.

WU, K.; HARMS, J. Performance study of a multipath routing method for wireless mobile ad hoc networks. In: **MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems**. [S.l.: s.n.], 2001. p. 99–107. ISSN 1526-7639.

WUNDER, M.; LITTMAN, M. L.; BABES, M. Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In: CITESEER. **Proceedings of the 27th International Conference on Machine Learning (ICML-10)**. [S.l.], 2010. p. 1167–1174.

XIANG, H.; TIAN, L. Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav). **Biosystems Engineering**, v. 108, p. 174–190, 2011.

XU, Q. et al. Proteus: network performance forecast for real-time, interactive mobile applications. In: **MobiSys**. [S.l.: s.n.], 2013.

YANG, P. long; TIAN, C.; YU, Y. H. Analysis on optimizing model for proactive ad hoc routing protocol. **MILCOM 2005 - 2005 IEEE Military Communications Conference**, p. 2960–2966 Vol. 5, 2005.

YANG, Q.; JANG, S.; YOO, S. Q-learning-based fuzzy logic for multi-objective routing algorithm in flying ad hoc networks. **Wireless Personal Communications**, p. 1–24, 2020.

YASSEIN, M. O. B.; DAMER; JORDAN. Flying ad-hoc networks : Routing protocols , mobility models , issues. In: . [S.l.: s.n.], 2016.

ZACARIAS, I. et al. Enhancing mobile military surveillance based on video streaming by employing software defined networks. **Wireless Communications and Mobile Computing**, v. 2018, p. 1–12, 2018.

ZANG, C.; ZANG, S. Mobility prediction clustering algorithm for uav networking. **2011 IEEE GLOBECOM Workshops (GC Wkshps)**, p. 1158–1161, 2011.

ZHAI, Z.; DU, J.; REN, Y. The Application and Improvement of Temporally Ordered Routing Algorithm in Swarm Network with Unmanned Aerial Vehicle Nodes. **9th ICWMC**, n. c, p. 7–12, 2013.

ZHANG, G. Time series forecasting using a hybrid arima and neural network model. **Neurocomputing**, v. 50, p. 159 – 175, 2003. ISSN 0925-2312. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0925231201007020>.

Zhang, Y. et al. Toward efficient network resource sharing: From one-sided market to two-sided market. **IEEE Wireless Communications**, p. 1–7, 2019.

ZHAO, Z. et al. Software-defined unmanned aerial vehicles networking for video dissemination services. **Ad Hoc Networks**, v. 83, p. 68 – 77, 2019. ISSN 1570-8705. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S1570870518306231>.

APPENDIX A — RESUMO EXPANDIDO

Devido ao desenvolvimento de redes emergentes como Internet of Things, aplicações em 5G e redes compostas por múltiplos UAVs (FANETs), existem dois desafios principais a serem resolvidos de forma a garantir condições aceitáveis de qualidade de serviço nesse tipo de rede: a predição da quantidade de tráfego e o roteamento eficiente de dados.

A predição da quantidade de tráfego em redes emergentes é um assunto de grande importância para garantir o compartilhamento igualitário de recursos e lidar com a eventual sobrecarga do espectro de frequências. Realizar altos investimentos para ampliar a infra-estrutura desse tipo de rede é uma opção inviável e muito custosa. Dessa forma, métodos baseados em abordagens que utilizam algoritmos de inteligência artificial podem ser usados para criar modelos de predição de tráfego e garantia de um QoS aceitável. Nesse tipo de modelo, dois aspectos devem ser levados em conta pelo operador de rede ao escolher a forma de predição: precisão e complexidade. Considerando um balanceamento justo desses dois aspectos, a análise da complexidade desses algoritmos é importante.

A popularização e a diminuição dos custos de produção de UAVs levou ao seu amplo uso em diferentes tipos de aplicações tanto militares quanto civis. Dentre as principais características das FANETs estão a alta mobilidade dos nodos e alta dinamismo da topologia da rede. Sendo assim, protocolos de roteamento adaptativos e autonômos são necessários para lidar com os desafios da transmissão eficiente de dados. Tais protocolos precisam ser robustos e confiáveis, e, nesse aspecto, o Q-learning é um algoritmo adaptável de aprendizado por reforço que apresenta uma abordagem promissora para ser utilizado como um esquema de protocolo de roteamento em uma FANET.

Sendo assim, as principais contribuições desse trabalho se dividem em dois aspectos diferentes. Primeiramente, na análise de complexidade dos modelos de predição de carga de tráfego que utilizam inteligência artificial. Nesse caso, se busca identificar os modelos que necessitam de menor poder computacional e possuem o maior nível de precisão. Além disso, é realizada uma dicussão sobre a troca justa entre os fatores de complexidade e precisão. Já em relação ao algoritmo de roteamento proposto Q-FANET, deseja se obter uma diminuição no delay da rede, a exploração dos últimos episódios com pesos diferentes e o melhoramento da parametrização do protocolo de transmissão em FANETs baseado nas condições do canal.

Dentre os algoritmos beaseados em inteligência artificial mais utilizados para mod-

elos de predição de carga de tráfego em redes estão o Modelo de Regressão Linear Múltipla (MLRM), a Árvore de Regressão, a Série de Fourier, o Aprendizado por Reforço (com foco especialmente no Q-learning), o Aprendizado Profundo e os Algoritmos Genéticos. Diversos trabalhos demonstram o uso de um ou mais modelos combinados para desenvolver padrões de carga e comportamento do tráfego. Além disso, tais abordagens com inteligência artificial também se mostram uteis até mesmo na predição da quantidade de incidência de radiação solar com base em informações de hora do dia, quantidade de luz, temperatura, área, etc, utilizando diferentes modelos de redes neurais.

No âmbito de protocolos de roteamento para FANETs, eles podem ser divididos em duas categorias principais: single-hop e multi-hop. Os protocolos do tipo single-hop, por sua vez, são categorizados como estáticos (LCAD e DCR). Já os protocolos do tipo multi-hop são divididos entre os baseados em topologia, posição (GPSR) e hierárquicos (MPC e CAUAV). Por fim, os protocolos baseados em topologia são divididos em próativos (DSDV e OLSR), reativos (DSR e AODV) e híbridos (ZRP e TORA). No entanto, a categoria de protocolos de roteamento para FANETs que se mostra mais promissora em termos de performance são aqueles baseados em técnicas de aprendizado por reforço - nesse caso o algoritmo de Q-learning. Diversos trabalhos apresentam as vantagens de tal abordagem, exibindo resultados que mostram delays menores, maior taxa de entrega de pacotes, entre outros benefícios, quando comparados a soluções similares no "estado-da-arte".

De forma a se verificar o aspecto de troca favorável entre os aspectos de complexidade e precisão, executou-se uma série de simulações com os modelos baseados em inteligência artificial para a predição de carga de tráfego em rede. Os experimentos consistiram na geração de tráfego de rede composto por 60% HTTP, 20% VoIP e 20% de video streaming. Apesar de as primeiras análises identificarem que o modelo de série de Fourier e o modelo de Deep Learning se mostrarem o mais preciso e o mais complexo dentre todos, respectivamente, os resultados da simulações apresentaram diferentes resultados. Satisfazendo padrões de QoS como delay e jitter, o MLRM se mostrou a melhor abordagem para ser utilizada por operadores de rede, além de oferecer a melhor troca entre os aspectos de complexidade e precisão.

Com tais resultados obtidos, desenvolveu-se a implementação da arquitetura do protocolo de roteamento proposto para FANETs, o Q-FANET. O algoritmo consiste de vários modelos que trabalham em conjunto para fornecer a rota que possua o menor delay e melhores condições de canal possível. Primeiramente, o módulo de descoberta de viz-

inhos estabelece a tabela de vizinhos de cada nodo através da troca de pacotes *HELLO*. Após isso, um módulo de decisão de roteamento utiliza das informações obtidas pela função de recompensa, o mecanismo de penalidade (que pune com recompensas mínimas os links em que problemas sejam detectados) e a limitação de velocidade (que garante a escolha de links com os menores delays possíveis para o encaminhamento de pacotes) para computar os Q-values. Tais valores são obtidos através do uso de uma versão modificada do algoritmo Q-Noise+, uma abordagem do algoritmo tradicional do Q-Learning que substitui o fator de desconto por uma quantidade limitada de episódios passados com pesos diferentes e um fator que leva em conta as condições do canal.

Para testar as vantagens do Q-FANET, um experimento com diferentes simulações foi executado para comparar a performance da solução proposta com dois algoritmos de roteamento de dados: Q-Geo, Q-Noise+ e QMR. O cenário de simulações proposto foi desenvolvido no simulador WSNet com 25 nodos em uma área de 500x500 m. Nas simulações, 1000 pacotes de $127Bytes$ são transmitidos com diferentes intervalos de tempo ($10, 20, 30, 40$ e $50ms$). Para cada intervalo de tempo, são executadas 100 rodadas, onde o resultado final é representado pela média dos valores coletados. As simulações avaliam três aspectos diferetes: delay máximo, jitter e taxa de entrega de pacotes. São realizados dois conjuntos de testes: um em que todos os nodos funcionam normalmente, e outro em que 10 nodos são desligados após $1s$ de início da simulação.

Analisando-se os resultados obtidos, pode-se afirmar que o Q-FANET apresenta ganhos significativos em relação ao seu principal concorrente em performance, o QMR. Em relação ao delay máximo, o Q-FANET mostrou melhoras de 45.71% (cenário normal) and 46.75% (cenário com nodos falhando), o que se deve principalmente por causa do uso do módulo de limitação de velocidade que escolhe o link com o menor delay possível. Já em em relação ao jitter, o Q-FANET teve melhoras de 44.49% (cenário normal) and 48.28% (cenário com nodos falhando), uma consequência direta da diminuição do delay. Por outro lado, o Q-FANET apresentou apenas uma baixa melhora em relação à taxa de entrega de pacotes, com ganhos de 0.81% (cenário normal) e 1.26% (cenário com nodos falhando), que foram ocasionados pelo uso dos 10 episódios mais recentes a escolha do canal com melhor SINR no no algoritmo do Q-Noise+.

Sumarizando, esse trabalho forneceu contribuições em dois tópicos diferentes. Em relação à predição de carga de tráfego em redes emergentes, se obteve que a revisão dos principais conceitos de cada heurística analisada proporcionou uma análise extensa de suas complexidades, informação que pode ser utilizada como conhecimento importante

para auxiliar as decisões de operadores de rede. Além disso, o MLRM mostrou-se o melhor modelo de predição de carga de tráfego ao balancear de forma favorável a alta precisão e baixa complexidade em cenários de redes emergentes como Internet of Things (IoT) e aplicações 5G.

Já o protocolo de roteamento proposto com o Q-FANET exibiu resultados que evidenciam a alta performance da abordagem em apresentar baixos delay máximo e jitter, quando comparado a outros protocolos competidores. Nos cenários de simulação, o Q-FANET também apresentou um pequeno aumento na taxa de entrega de pacotes.

Com relação a trabalhos futuros, no âmbito da predição de carga de tráfego em redes espera-se executar cenários de simulação mais realistas utilizando algoritmos de aprendizagem de máquina, além da implementação dos modelos de predição que utilizam inteligência artificial em ambientes reais de operação de redes emergentes. Para o Q-FANET, se almeja tratar de outros problemas importantes como o consumo de energia - um fator de grande importante quando tratamos de pequenos UAVs com recursos energéticos limitados. A adoção de parâmetros online também é uma possível direção futura para melhorar a abordagem proposta pelo Q-FANET. Além do mais, seria interessante explorar diferentes padrões de movimento, ambientes simulando redes LTE e outros simuladores de redes sem fio como o NS3 para simular experimentos exploratórios adicionais.