Machine Learning for Performance Aware Virtual Network Function Placement

Dimitrios Michael Manias, *The University of Western Ontario*

Supervisor: Shami, Abdallah, *The University of Western Ontario*

Abstract

With the growing demand for data connectivity, network service providers are faced with the task of reducing their capital and operational expenses while simultaneously improving network performance and addressing the increased connectivity demand. Although Network Function Virtualization has been identified as a potential solution, several challenges must be addressed to ensure its feasibility. The work presented in this thesis addresses the Virtual Network Function (VNF) placement problem through the development of a machine learning-based Delay-Aware Tree (DAT) which learns from the previous placement of VNF instances forming a Service Function Chain. The DAT is able to predict VNF instance placements with an average 34µs of additional delay when compared to the near-optimal BACON heuristic VNF placement algorithm. The DAT's max depth hyperparameter is then optimized using Particle Swarm Optimization (PSO) and its performance is improved by an average of 44µs through the introduction of the Depth-Optimized Delay-Aware Tree (DO-DAT).

Keywords: Network Function Virtualization, NFV, Virtual Network Functions, VNF, Service Function Chain, SFC, VNF Placement, Machine Learning, Optimization, Decision Tree, Particle Swarm Optimization, PSO

Summary for Lay Audience

The past two decades have seen an incredible increase in the number of devices producing network traffic including: smartphones, tablets, wearable smart devices, and smart home accessories. This surge in network traffic puts a major burden on service providers world-wide. In order to keep up with the demand, these providers must make significant upgrades on their network infrastructure. This proves to be a challenging task as there are several functionalities (e.g. firewalls) offered by the network which require infrastructure specific to those functions. Upgrading these specific infrastructural components and increasing the number of each present in the network is an extremely costly endeavour which results in significant capital expenditures incurred by the service providers. As a way to mitigate these costs as well as improve system performance and network strength, Network Function Virtualization (NFV) has been proposed as a solution. This technology essentially isolates the functionality of each of these specific infrastructural components and turns them into software applications which can be run on generic infrastructure such as datacenter servers. There are several issues which arise from this technology which must be addressed to ensure its feasibility. One of these includes the placement of these software applications in the network. While traditionally this placement task has been achieved through optimization models and approximate solutions, these can often require significant time and resources to solve. This is inadequate for network systems as there are several time critical applications using the network. As an alternative, this work outlines the use of machine learning to make a model which predicts the placement of each of these software applications based on previous placements. Quantitative results show that the machine learning model can predict a placement which produces a slightly higher delay between applications compared to a current approximate solution. To mitigate this, a domain based optimization model is presented to optimize the parameters of the previous machine learning model such that the placement results in reduced delay between applications. Results show a significant improvement once optimized and confirm that the work presented is a significant step towards a fully automated placement strategy.

Statement of Co-Authorship

This thesis contains the following manuscript that has been accepted for publication.

- D. M. Manias, M. Jammal, H. Hawilo, A. Shami, P. Heidari, A. Larabi, R. Brunner, "Machine Learning for Performance-Aware Virtual Network Function Placement," in GlobeCom, 2019. (accepted)

This thesis constains the following manuscript which will be submitted for review.

- D.M. Manias, H. Hawilo, M. Jammal, A. Shami, "Depth-Optimized Delay-Aware Tree (DO-DAT) for Virtual Network Function Placement" (pending submission)

The following coauthors provided experimental and technical support for the studies listed above:

- A. Shami contributed to the work done in Chapters 3 and 4 through his experiece as a professor, his technical expertise in the field, and his opinion and perspective.

- H. Hawilo contributed to the work done in Capters 3 and 4 through his technical expertise in the field and his opinion and perspective.

- M. Jammal contributed to the work done in Chapters 3 and 4 through her technical expertise in the field and her opinion and perspective.

- P. Heidari contributed to the work done in Chapter 3 through her experience in the field and research at Edgegravity by Ericsson.

- A. Larabi contributed to the work done in Chapter 3 through his experience in the field and research at Edgegravity by Ericsson.

- R. Brunner contributed to the work done in Chapter 3 through his experience in the field and research at Edgegravity by Ericsson.

Epigraph

«ὁ δὲ ἀνεξέταστος βίος οὐ βιωτὸς ἀνθρώπῳ»

- Πλάτων, Απολογία Σωκράτους 38a 1-6

Dedication

I would like to dedicate this thesis to my parents Vicky and Nickolaos Manias.

Acknowledgements

The completion of the work presented in this thesis would not have been possible without the constant support I received from my friends, family, and colleagues. I would like to take this moment to personally thank them for all their contributions.

I would like to thank my supervisor Prof. Abdallah Shami for his constant support throughout the completion of this degree. While being my supervisor, Dr. Shami constantly encouraged me to think about innovative solutions and to continually produce higher quality work. I am grateful to Dr. Shami for helping me become the researcher I am today through his guidance and mentoring, and for introducing me to the research field which I have grown to love. I consider Dr. Shami a friend as he is always approachable and willing to advise me on any problems I face. I would once again like to thank Dr. Shami for everything he has done and look forward to our future collaboration.

I would like to take this opportunity to thank the staff in the Faculty of Engineering and specifically our ECE graduate coordinator Stephanie Tigert. The amount of effort Stephanie puts into coordinating all of the ECE graduate programs is truly incredible. She is always able to answer any program-related question and always goes out of her way to make our experience as ECE graduate students better.

A giant thank you to Hassan Hawilo and Dr. Manar Jammal for everything they have done for me. These two incredible individuals have stood by me since day 1 of my program and continue to do so to this day. Their door was always open for me and delicious food was always on the table every time I visited them. Hassan and Manar were always there to listen to my problems and help me through them. I am so grateful for their continual love and support and the incredible contribution they have made in helping me achieve my goals. I am happy to say that these two individuals are two of my closest friends and I look forward to seeing what the future holds.

Next, I would like to thank three of my closest friends who have been with me since undergrad, Abdullah Ramadan, Tarek Tayeh, and Ayman Hashisho. Words cannot describe how much I love these three individuals. They are always there for me during the ups and downs and always know how to cheer me up when I'm having a bad day. Abdullah, Tarek, and Ayman are definitely my brothers for life and I am incredibly grateful for everything they have done and continue to do for me.

Ibrahim Shaer is an individual who joined our research group one semester after I did and since joining I can without a doubt say that he is one of my closest and most trusted friends. It feels as though I have known Ibra for years even though it has only been months. Ibra is always there for me and knows exactly what to say to cheer me up. 99% of the time we are together laughter is guaranteed (the other 1% results in intense socio-economic and political discussions which inevitably end up resulting in laughter). I am so grateful for everything Ibra has done for me over the past year and look forward to our lifelong friendship.

I would like to take this opportunity to thank Yiota Stamatopoulou for all her help and support throughout this journey. Yiota and I met under the strangest circumstances which had us both saying "what a small world". Yiota is always there for me and is honestly one of the best individuals I have ever met. We constantly complain to each other about our problems and I can honestly say it is one of the most therapeutic ways to de-stress in grad school. I knew from the day I met her that I had made a great friend and can't wait to see what the future holds.

Ming Xu is one of my closest friends from my time in highschool and has definitely been my number 1 supporter throughout the years. She is always willing to go for sushi to discuss life and constantly supports me and motivates me to achieve great things and pursue my dreams. Thanks 'homeslice'.

Having recently met Angelos Almpanis and George Efstathopoulos I have added more members to my ‘Hellenic Crew’. These guys are great friends and are always willing to hang out and have a good time. Our summer road trips have been excellent ways for me to de-stress and I am grateful for everything they have done. I wish them the best of luck and can’t wait to see where our next road trip takes us.

Paulina Himaras ‘Boubouki’ is one of my oldest and closest friends. Paulina and I have grown up together and have shared good times and bad times. We are always there to support each other and I truly value her continual love and support. She is an incredible individual with a bright future and I wish her the best of luck with everything.

I would like to take this opportunity to thank all of my lab colleagues namely Elena Uchiteleva, Cesar Augusto Gomez Suarez, and Ibrahim Tamim. Your presence really made this experience much more enjoyable and your constant support is greatly appreicated.

Now to all the friends I have made along the way who have contributed to this incredible journey. I am so grateful to have met all of you and wish you all the best. Unfortunately, listing you all by name would be an entire thesis by itself but you know who you are.

Next I would like to thank my family for always being there to support me. The amount of love you have all shown me is truly invaluable and I am forever grateful for everything you have done and continue to do for me. If all the friends I have made along this journey would require an entire thesis to list, my big fat Greek family would require 10. I love you all and thank you for everything.

Finally, the biggest thank you to my parents Vicky and Nickolaos Manias. Words can’t describe how grateful I am for everything you have done for me and how much I love you. From a young child, you have pushed me to become a better version of myself and peruse my dreams. Δεν μπορείτε να φανταστείτε ποσό πολύ σας αγαπάω και σας ευχαριστώ για όλα όσα μου προσφέρετε. Κανένα από τα επιτεύγματα μου δεν θα ήταν δυνατά χωρίς την αγάπη και την υποστήριξη σας και ελπίζω να σας κάνω υπερήφανους.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ANN	Artificial Neural Network
BACON	Betweenness Centrality Algorithm for Component Orchestration of NFV Platforms
CAPEX	Capital Expenditures
CART	Classification and Regression Tree
CP	Computational Path
CPU	Central Processing Unit
DAT	Delay Aware Tree
DO-DAT	Depth-Optimized Delay Aware Tree
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
GPU	Graphical Processing Unit
HSS	Home Subscriber Service
ILP	Integer Linear Programming
IP	Internet Protocol
kNN	k-Nearest Neighbours

LTE-A	Long Term Evolution Advanced
MANO	Management and Orchestration
MILP	Mixed Integer Linear Programming
ML	Machine Learning
MME	Mobility Management Entity
NFV	Network Function Virtualization
NSPs	Network Service Providers
OPEX	Operational Expenditures
PC	Personal Computer
PDF	Probability Density Function
PGW	Packet Data Network Gateway
PSO	Particle Swarm Optimization
QoS	Quality of Service
RAM	Random Access Memory
SFC	Service Function Chain
SGW	Serving Gateway
SVM	Support Vector Machine
vEPC	Virtual Evolved Packet Core
VIM	Virtual Infrastructure Manager
VNFM	Virtual Network Fucntion Manager

VNFO	Virtual Network Function Orchestrator
VNFs	Virtual Network Functions

Chapter 1

Introduction

With network connectivity demands at an all-time high and continuing to increase, Network Service Providers (NSPs) are tasked with the challenge of accommodating additional bandwidth requests on their networks while concurrently maintaining or improving their Quality of Service (QoS). When considering the massive growth in Internet Protocol (IP) traffic due to the spike in IP-connected devices, coupled with the imminent introduction of 5G networks, this task can be quite cumbersome. It is estimated that during the five year period from 2017-2022 the global IP traffic will increase by a factor of three, the number of IP connected devices will be three times greater than the global population, and the average mobile connection speed will triple [1]. In order to meet these unprecedented and growing demands, NSPs should enhance the portability, interoperability, performance, reliability, security, and management of their networks while reducing their Capital Expenditures (CAPEX) and Operational Expenditures (OPEX) [2]. One way to address these needs is through Network Function Virtualization (NFV).

1.1 Network Function Virtualization (NFV)

NFV is a technology proposed by the European Telecommunications Standards Institute (ETSI) in 2012 to solve the challenges mentioned above as well as those associated with service availability, scalability, and resilience of current networks [3]. The goal of NFV technology is to isolate the network functions from their underlying hardware and execute them as software-based applications known as Virtual Network Functions (VNFs) on servers and in datacenters. There are several benefits which arise from the implementation of NFV architecture including: reduction in capital and operational expenditures, decreased time to market for new technologies, service testing and implementation efficiencies, network topology optimization, optimized energy consumption, and increased operational efficiencies [2]. However, there are challenges associated with these benefits that must be solved in order to experience the full potential and power of this technology.

Fig. 1.1 illustrates the traditional networks of today. In this figure it can be seen that the network is very static with a rigid structure. All network functions are located on function-specific hardware. It is difficult to add additionaly functionality and scale this network due to the reconfigurations and CAPEX investments required.

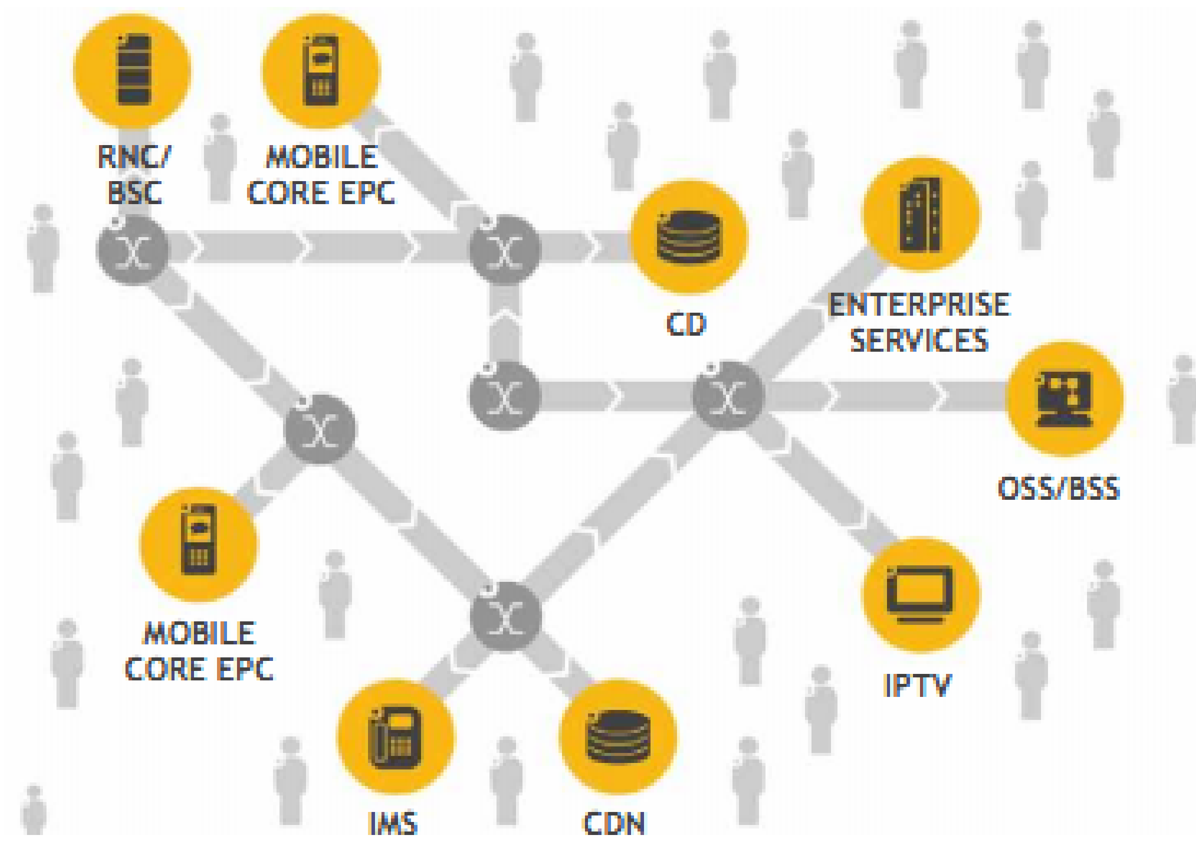

Figure 1.1: Current Networks [4]

Fig. 1.2 illustrates the networks of tomorrow. These networks incorporate concepts of NFV. In contrast with the previous network, the depicted network has one unified platform for all network functions and services. These virtualized functions are available to all end users, and the scaling of network size and functionality does not require rigid network reconfigurations as this is a highly dynamic environment.

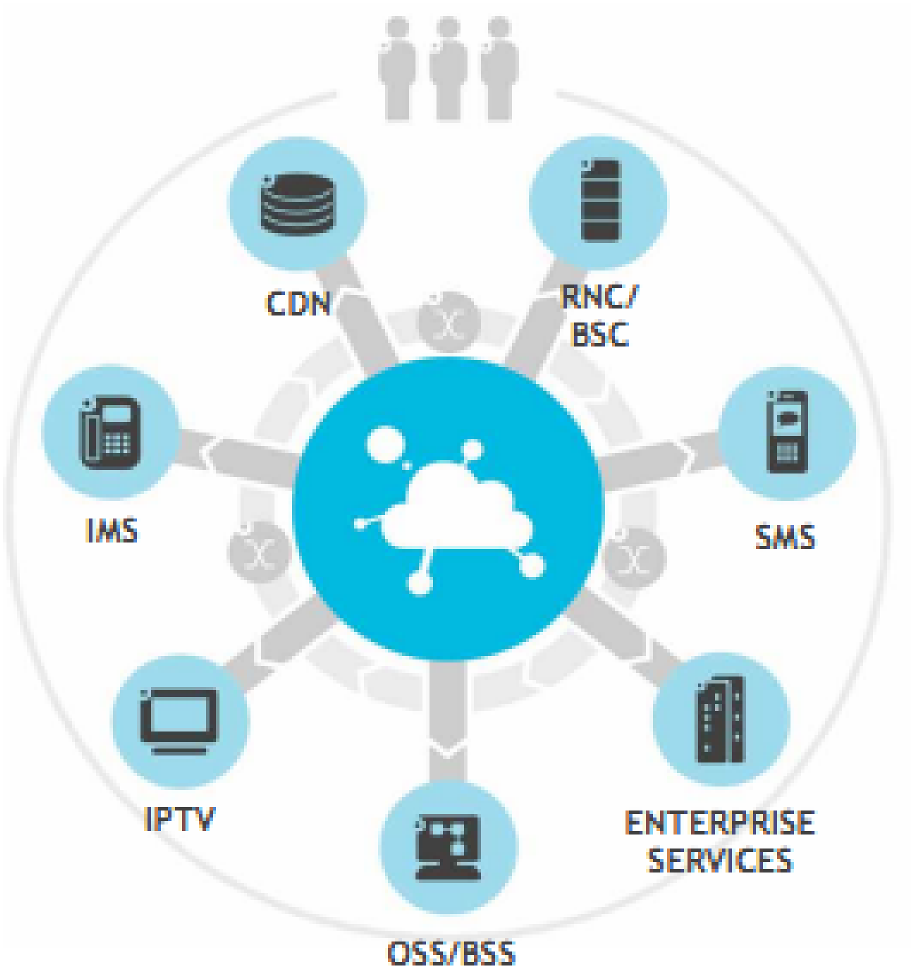

Figure 1.2: Future Networks [4]

NPSs world-wide are held to certain standards when providing a service to a customer. QoS guarantees are a set of standards which describe the required performance of a NSPs delivered service. QoS requirements take into consideration metrics such as: packet loss, jitter, transmission delay, availability, amongst others. The QoS guarantee is a NSPs acknowledgement of and adherence to these requirements. When considering the implementation of NFV architecture, QoS guarantees are of paramount importance and consideration and without meeting them, the implementation of NFV technology in current networks would not be feasible.

Performance, being a key metric of QoS, plays a major role in the successful implementation of NFV architecture. Furthermore, automation has been identified as one of the key factors behind the successful scalability of NFV-enabled networks and is listed as being paramount to the successful implementation of this technology [3].

1.2 Research Contributions

The work outlined in the subsequent chapters introduces several research contributions namely:

- Chapter 3

1. The implementation of the Delay-Aware Tree (DAT) machine learning-based model to predict the placement of dependent VNF instances forming a SFC.
2. The implementation of a machine learning model which learns from previous near-optimal placements when deciding the servers for future placements.
3. The implementation of a machine learning model which learns operational constraints when selecting the best server out of a list of candidate servers to ensure QoS agreements are met.

- Chapter 4

1. An optimization model for the DAT to optimize its performance through the reduction of invalid placement predictions and the delay between interconnected instances forming a SFC.
2. The creation of the Depth-Optimized Delay-Aware Tree (DO-DAT) which exhibits improved performance compared to the DAT.
3. An analysis into the generalization and transferability of DO-DAT to different system configurations.

Bibliography

[1] K. Wiederhold, G. Riva, and G. Graffigna, “Cisco Visual Networking Index: Forecast and Trends, 2017–2022,” Annu. Rev. CyberTherapy Telemed., pp. 2017–2022, 2013.

[2] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, “NFV: State of the Art, Challenges, and Implementation in Next Generation Mobile Networks (vEPC),” Network, vol. Nov, no. December, pp. 18--26, 2014.

[3] M. Chiosi, *et al.*, “Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action,” ETSI, Darmstadt, Germany, 2012. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf

[4] LTT BUSINESS CONSULTING, "LTT Business Consulting," 2019. [Online]. Available: http://www.lttbusinessconsulting.com/snd.php.

Chapter 2

Background

The following section give some background information pertinent to the understanding of the contributions outlined in subsequent chapters of this thesis.

2.1 Service Function Chain (SFC)

While the performance of an individual VNF instance is important, the performance of an interconnected and interdependent group of VNF instances known as a Service Function Chain (SFC) is paramount. SFCs are the end goal of NFV enabled networks; In order to provide an end-to-end service, several VNF instances of differing types will need to be accessed in a specific order, thus creating a SFC.

An example of a SFC is the Evolved Packet Core (EPC) which is a network infrastructure which supports the converging on licensed and unlicensed radio access technologies through IP, commonly refered to as Long Term Evolution Advanced (LTE-A) [1]. Virtual EPC (vEPC) is a solution introduced by 3GPP to harness the full potential of radio access technologies [2]. In this technology there are four main types of VNFs: the Home Subscriber Service (HSS), the Mobility Management Entity (MME), the Serving Gateway (SGW), and the Packet Data Network Gateway (PGW). Fig 2.1 outlines the architecture of this technology. As with any SFC, vEPC is subject to QoS guarantees including performance. When considering a VNF-enable network, the placement of each of the VNF instances forming the SFC directly impacts performance.

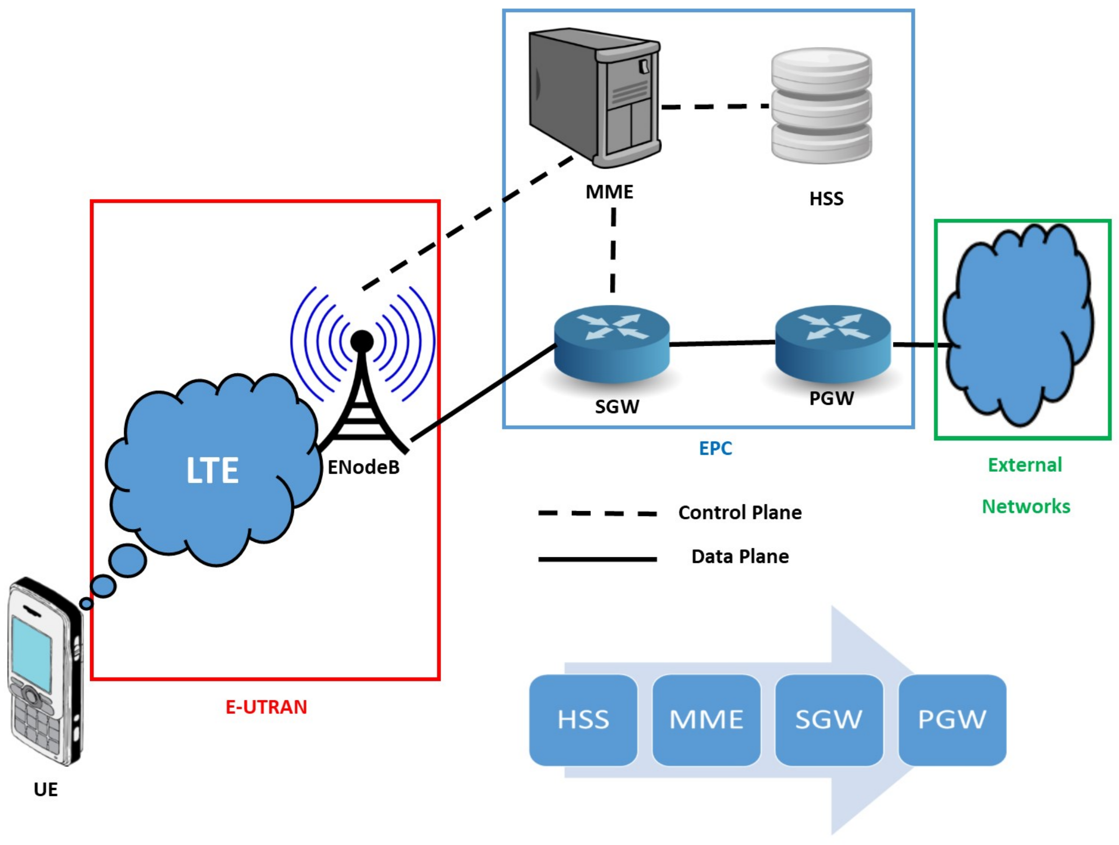

Figure 2.1: Evolved Packet Core (EPC) Architecture

2.2 NFV Management and Orchestration

There are three key functional components which make up the ETSI Management And Orchestration (MANO) framework including the Virtualized Infrastructure Manager (VIM), the VNF Manager (VNFM) and the VNF Orchestrator (VNFO). Combined, the three aforementioned components are responsible for several key operational services in NFV enabled networks including: carrier grade requirements, performance, service availability, management, and VNF placement. Fig. 2.2 outlines the NFV Architecture Framework as proposed by ETSI including the NFV MANO.

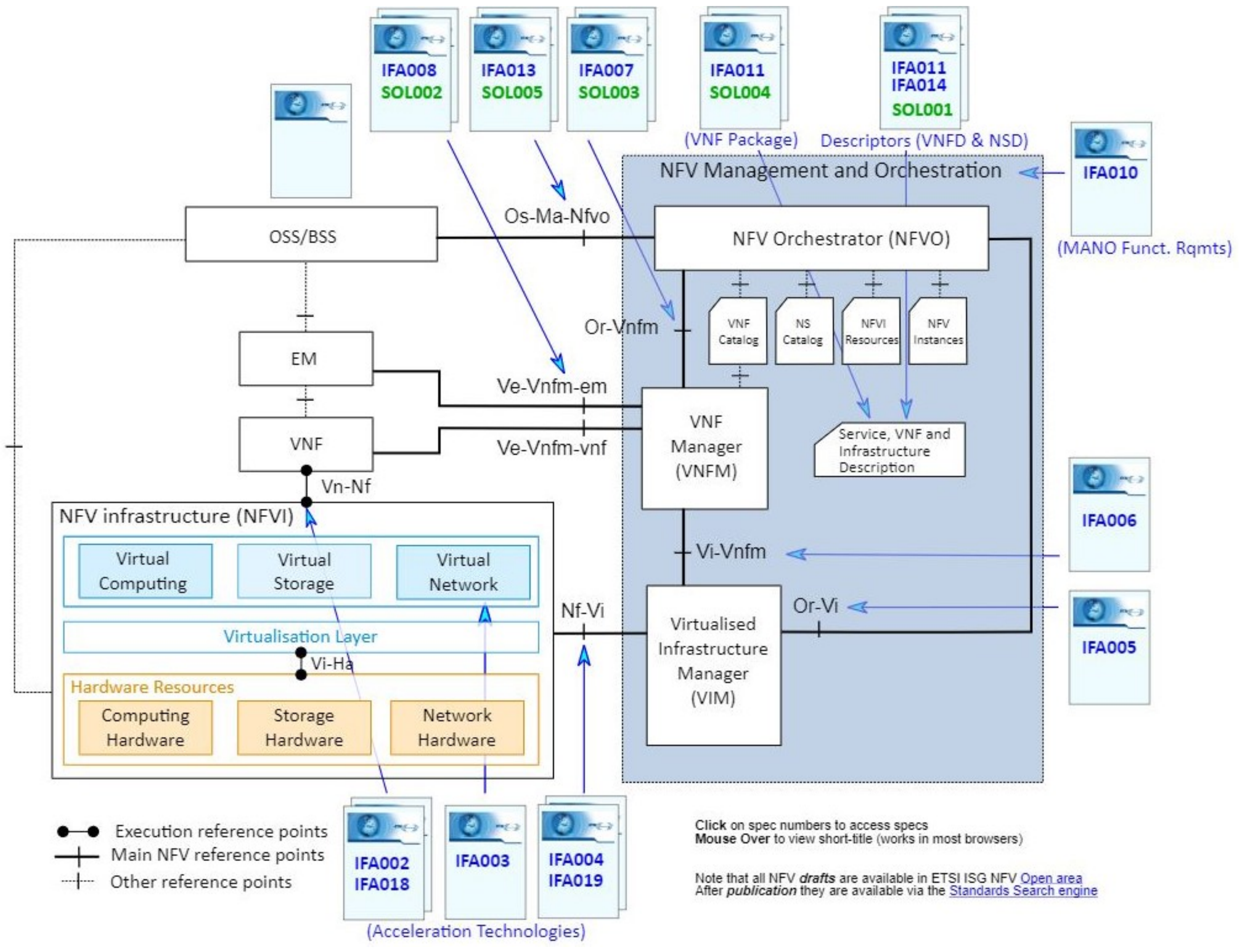

Figure 2.2: NFV Architecture Framework [3]

2.2.1 Virtualized Infrastructure Manager (VIM)

In the ETSI MANO framework, the VIM is responsible for the management of VNF interactions with both physical and virtualized resources [4]. To this end, its responsibilities include awareness of the existence and quantity of resources, resource allocation, and resource management. Furthermore, the VIM provides the network with insights into VNF infrastructure management, performance analysis, fault information, and network metrics to enhance capacity planning and optimization.

2.2.2 VNF Manager (VNFM)

In the ETSI MANO framework, the VNFM is responsible for managing the VNF from a lifecycle perspective. This includes critical functions such as the commencement of a service through VNF instantiation, the expansion of a service through VNF scaling, the recovery of a service through VNF migration, and the ending of a service through VNF termination. By handling the various stages of a VNF lifecycle, the VNFM is also responsible for maintaining the function's performance throughout the duration of its lifecycle using the functions previously mentioned [5].

2.2.3 VNF Orchestrator (NFVO)

The main role and priority of the orchestrator in the ETSI MANO framework is to realize network services on the infrastructure of the NFV-enabled network. By managing resources across several VIMs and performing several orchestration functions through the VNFM, the VNFO is able to allow VNF instances to access and share resources. Having the ability to communicate and direct both the VIM and VNFM makes the orchestrator the prime candidate for the deployment of the solution presented in this thesis.

2.2.4 Key Functionalities of NFV MANO

The following outlines some of the key functionalities of the ETSI NFV MANO framework and their impact on the network as a whole.

2.2.4.1 Instantiation

VNF instantiation is the first step in the NFV lifecycle. In order to instantiate a VNF several aspects must be considered. Initially, the feasibility of the VNF instantiation is considered. This process begins when an instantiation request is received by the NFVO. The goal of this stage is to ensure that the resources required for the instantiation can be reserved before the actual processing of the request is complete. Furthermore, this step ensures that the request formatting is practical through validation as any illogical or ill-formatted request will be deemed in-feasible. Once received, the NFVO communicates with the VNFM to assess the feasibility of the request. Once processed, the VNFM provides a response to the NFVO effectively requesting the allocation of resources (if feasibility was successful). Once received, the NFVO proceeds to initiate the pre-allocation of resources.

The processing of the pre-allocation of resources is one of the most crucial functions executed by the NFVO. This function begins firstly with the validation of policy adherence when the allocation request is received by the NFVO. Once validated, the NFVO is tasked with the selecting a location for the instantiation of the VNF instance. This specific stage is the industrial implementation of the formulated VNF placement problem. The NFVO must consider available resources, VNF type, SFC chains, and operational policies when selecting the placement of the function [6]. Furthermore, additional consideration must be made to ensure that all QoS requirements and external dependencies can be realized and preserved.

If needed, the pre-allocation step can entail communication with the VIM for the requesting of current resource capacities in the case that additional resources (internal connectivity, virtual machines, storage) are required for the pre-allocation phase. If such a request is received by the VIM, it evaluates the current resource allocation and capacity and proceeds to reserve them. Once complete, the VIM interacts with the NFVO to communicate the success or failure of the reservation. Once the additional allocation of resources has been completed (if required) the NFVO communicates with the VNFM to receive the final configuration information required to finalize the VNF instantiation. If successfully completed, the NFVO communicates with the sender of the original instantiation request and outlines that the request has been completed successfully.

2.2.4.2 Scaling

VNF scaling is an important functionality in NFV-enabled networks as it enables increasing flexibility in the system. There are several instances which can be deemed as scaling operations including: the re-configuration of resources, the addition of new resources, the de-activation and elimination of running virtual machines, and the de-allocation previously allocated resource [6]. The first stage in executing the scaling functionality is realizing when there is a need for scaling. Several instances can trigger the need for a scaling event including: the degradation of service performance, requirement of additional resource allocation, underutilization, and over-provisioning.

Scaling can be invoked by several components of the NFV MANO framework depending on the situation. For example, the VNFM can trigger scaling while monitoring the state of a VNF, or the NFVO can trigger scaling due to an external request. In any case, the general procedure for handling the scaling request is the same. Firstly, the nature of the scaling action required must be determined. This can prove to be a complicated step due to the interconnection and interdependence of various VNF instances. If for instance, the performance of a given VNF is being negatively impacted by a second VNF, before any scaling event can be executed, the impact of the scaling of either of these VNFs on the remaining network instances must be considered.

In general, when the NFVO receives a scaling event request, the first stage in processing request is validating it through the evaluation of policy adherence. Once validated, the NFVO communicates with the appropriate VNFM to handle this request. Upon receiving this request the VNFM prepares the ground work for the scaling event by addressing the validation of the request and once validated, responds to the NFVO with a request to change resources. This resource change request is processed by the NFVO and delegated to the VIM. As with instantiation, the VIM begins the modification of the internal network and reserves additional (or fewer) resources as required by the scaling event. The VIM proceeds to communicate the completion of this resource allocation back to the NFVO which goes on to transfer this acknowledgement to the VNFM. Having received the acknowledgement, the VNFM is in a position to begin the configuration of the VNFs according to the designated scaling event. Once configured, acknowledgement is sent back to the NFVO which is transferred to the original entity requesting the scaling event.

2.2.4.3 Termination

The function of VNF termination is the final stage in the lifecycle of a VNF. The process by which the NFVO proceeds to terminate a VNF begins with the request for termination targeting an instantiated and active VNF instance. Once received, the NFVO is responsible for the validation of the request. Special care must be taken when validating the termination request to ensure the author behind the request has the proper authority to make such a request. Once validated, the NFVO communicates with the VNFM which proceeds to terminate this VNF. Once terminated, the VNFM reports back to the NFVO with an acknowledgement of the termination. Upon successful acknowledgement of a VNF termination, the NFVO proceeds to request the deletion of the resources previously allocated to the now terminated VNF. This entails a deletion of the internal network, as well as the deletion of the computational and storage resources by the VIM. Acknowledgement of the resource deletion is then passed back to the NFVO which relays the information back to the initial entity initially requesting the VNF termination.

2.2.4.4 Fault and Failure Management

Fault management is a critical and important functionality provided by NFV MANO. Fault management describes the ability of the network to identify the occurrence of a fault and mitigate its impact. Faults in an NFV-enabled network can occur due to internal or external factors. Impacts of these faults can be devastating on the overall performance of the network. In order for a network to be considered strong, it must be able to respond to faults in a timely and appropriate manner as to not escalate them to severe system failures.

Fig. 2.3 displays a basic failure management system by which a network can recover from faults. Initially, the network is in state S_0. This state is where optimal operation is achieved as the delivered service quality is normal and the service parameter metrics (jitter, packet loss, etc.) are well within the acceptable thresholds. During the event of a failure, the delivered service quality and the service parameter metrics are severely degraded and the resulting state S_1 is observed. In this state the network considered to have failed and the appropriate mechanisms must be invoked to detect the failure. This detection is completed by several different network entities by observing the behaviour of their subordinate entities and components.

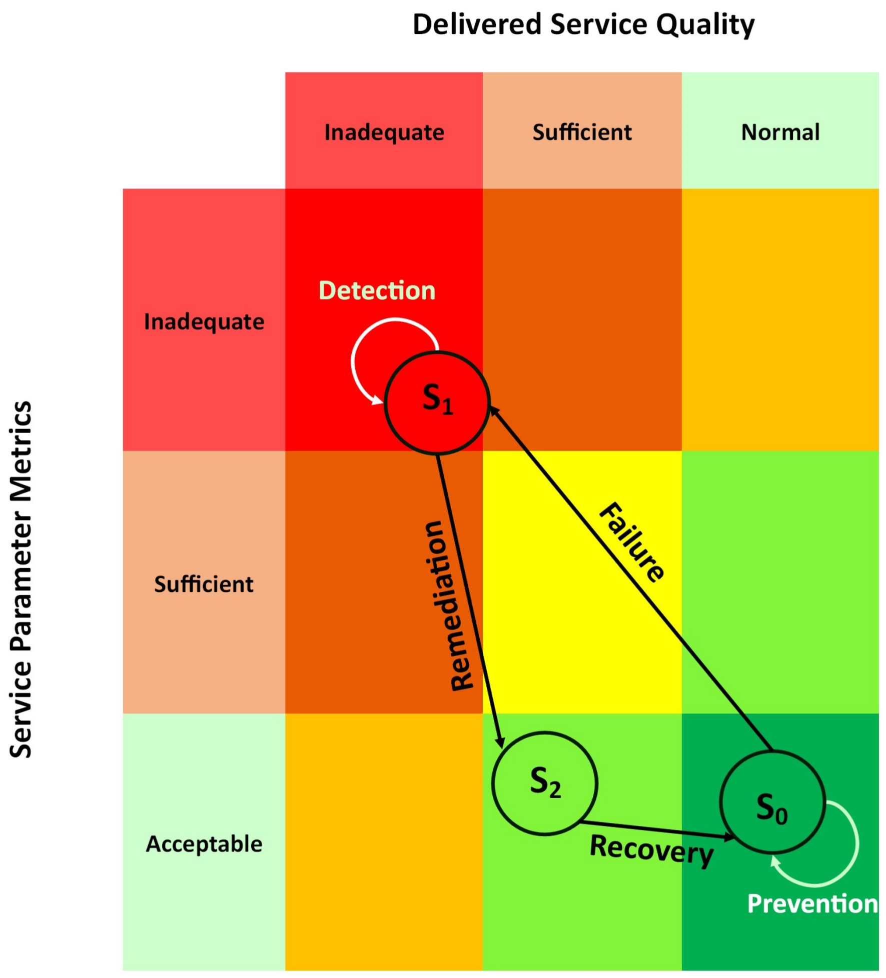

Figure 2.3: Stages of Failure Management

Once the fault has been detected, the remediation process beings. This process entails the propagation of the fault detection information through the various entities of the NFV MANO. Depending on the type of fault observed, different entities of the NFV MANO may be required to intervene for its mitigation. For example, the VNFM has the ability to address the correction of certain faults such as those associated with a specific VNF instance. If the VNFM is unable to adequately address the fault, various other entities including the NFVO can be required to intervene.

The NFVO receives a variety of different fault notifications from a variety of different entities. Once received, the NFVO is responsible for either directly applying fault correction techniques with the goal of further examining the cause of the fault, or delegating the task to the appropriate network entity.

Once action has been taken to remediate the observed fault or failure, the network is in an intermediate phase where service is restored; however, the network operation is not optimal as represented by state S_2. Once full fault recovery has been achieved and the root cause of the fault has been addressed, the system reverts back to the initial state S_0 and optimal operation is re-established.

By examining the fault and failure system presented, it is evident that ideally, preventative measures would be occurring during stage S_0 to pro-actively prevent the occurrence of a fault or failure in the network. These preventative measures can manifest themselves through many different functions of the NFV MANO including the initial placement of instantiated VNFs as well as the injection of resilience to the network.

2.3 SFC Computational Paths

Within a given network topology, SFCs must be placed such that the QoS requirements are met. Ideally, any system will have built in mechanisms or best practices which improve its resilience and reduce its susceptibility to faults or failures. When considering an SFC the way to do this is by introducing the concept of computational paths whereby several instantiations of the main VNF instances forming a SFC are present throughout the network [7]. This means that should a particular instance fail for any reason, there is a backup and alternate path available for the SFC to be executed. This concept is further illustrated in Fig. 2.4 where the various computational paths for a vEPC SFC with multiple VNF instances are shown.

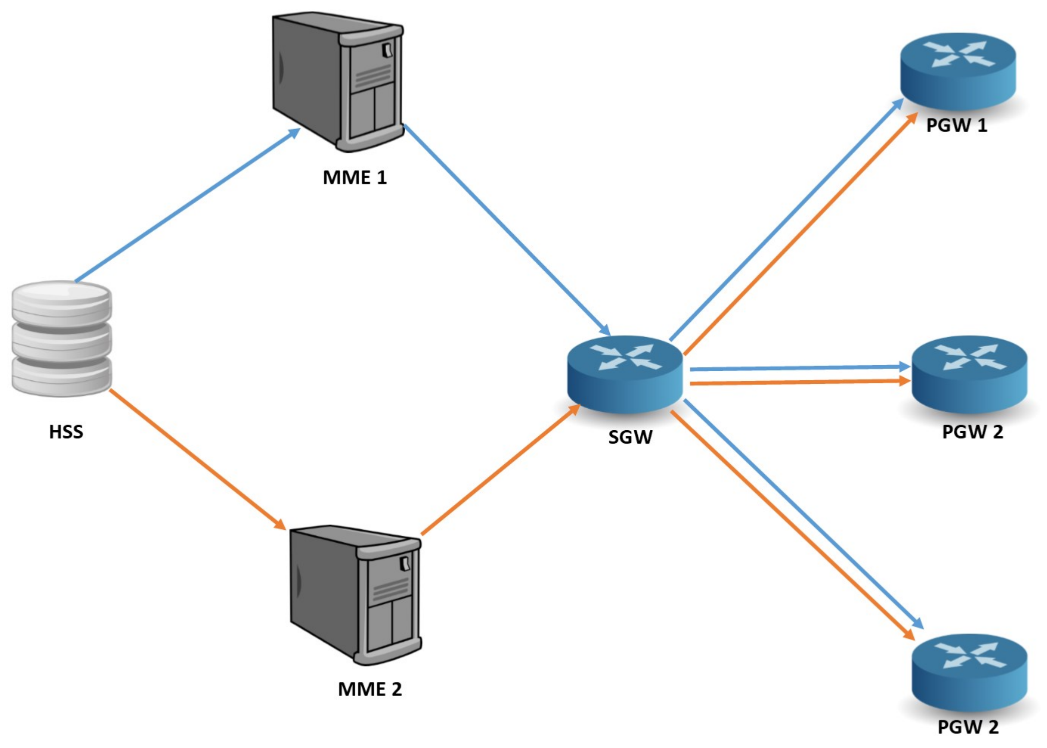

Figure 2.4: Computational Paths for vEPC SFC

When considering the resilience of a system, the overall minimization of the end-to-end delay across all computational paths must be addressed.

2.4 VNF Placement Problem

The VNF Placement problem can be defined as the problem by which a set of VNF instances are placed on a set of network servers while adhering to QoS guarantees and minimizing resource consumption. The VNF placement problem has been defined as NP-Hard, therefore, the use of various different methods and techniques has been suggested as solutions. Since the objectives, needs, and priorities of NSPs can vary over time, there are multiple solutions addressing various operational aspects of a NSP when considering VNF-enabled networks. The solutions are presented in three categories: optimization models, heuristics and meta-heuristics, and machine learning.

2.4.1 Optimization Models

The most common type of solution addressing the VNF placement and chaining problem is Integer Linear Programming (ILP). In this type of problem formulation all variables are restricted to being integers during the optimization process. Some of the optimization objectives found in literature include: the minimization of cost for all links across a network [8], the minimization of the number of servers used [9][10], the minimization of the number of cores used [9], the minimization of the resource cost of placement [11][12][13], the minimization of the number of instances placed in the network [14], the minimization of the service response time [15], as well as the minimization of the resource utilization [15]. Extending on the formulation of the VNF placement and chaining problem though ILP, Mixed-Integer Linear Programming (MILP) has been suggested in several works. A MILP problem can be defined as an optimization problem where only certain variables are constrained to being integers. Some of the objectives considered through MILP formulations include: core minimization [16], utilization minimization [17], resource allocation minimization [17], QoS violation minimization [17], and the minimization of delay between interdependent VNF instances forming a SFC [18].

2.4.2 Heuristic and Meta-Heuristic Solutions

Due to the NP-Hardness of the VNF placement problem, several works propose the implementation of heuristic solutions achieving near-optimal placements with reduced time complexities. Heuristic solutions implemented to solve the VNF placement problem include: greedy algorithms [19], Eigen decomposition [19], Markov approximation [20], set formation [13], graph-based [14], and betweenness centrality-based [18]. Additionally, the use of nature-inspired, population-based, meta-heuristics has become increasingly popular due to their ability to converge to a solution effectively through population mutations. The two most popular meta-heuristic algorithm observed are the genetic algorithm and the particle swarm optimization algorithm. The genetic algorithm has been used with the objective of minimizing network link costs [8] and minimizing the number of servers used [21], whereas the particle swarm optimization has been used to minimize the cost of placement of VNF components [17].

2.4.3 Machine Learning

Recently, the use of machine learning has become increasingly popular in the field due to the increasing requirements for network automation with the impending 5G networks. Currently, several solutions have been proposed which either incorporate machine learning or are solely based upon machine learning techniques however, significant progress must still be made before a fully autonomous VNFO is achieved. By addressing the various functionalities of a VNFO through simplified machine learning problems, a solid foundation is built towards a fully implementable and operational NFVO which meets the requirements of NSPs.

2.5 Supervised Machine Learning Classification

Supervised machine learning classification describes the set of algorithms which take labelled data and build a model which is used to predict the class given a set of features [22]. When considering the effectiveness of a classification algorithm, the predictive accuracy, is generally the standard metric used for evaluation. However, depending on the nature of the dataset and problem at hand, the use of predictive accuracy may not be an effective metric. In the case of a highly skewed distribution between classes, or the existence of several classes and multiple outputs, additional metrics must be considered. Ideally, these metrics will directly relate to the domain of the problem and therefore lead to an acceptable solution. The following presents an overview of some of the prevalent methods of supervised machine learning classificaiton.

2.5.1 Support Vector Machine (SVM)

Support Vector Machines (SVM) are one of the most popular supervised machine learning classification techniques due to their simplicity and understandability. The goal of SVM is the maximization of the hyperplane separating the labelled classes in the data. The training data points which fall on the boundaries of the hyperplane are referred to as support vectors. These support vectors are generally considered the hardest points to classify as they are the closest in term of proximity to the other points of differing classes. The algorithm operates by formulating an optimization function which maximizes the distance between the support vectors of the various classes and solving it through the Lagrangian multiplier technique. The premise of this method is that through the maximization of the distance from each of the classes' respective support vectors to the hyperplane, the greater the predictive performance of the model due to the minimization of the generalization error.

Fig. 2.5 illustrates the concept of separating the classes thought the optimal hyperplane. In this figure, there are two features, X1 and X2 represented. There are also two classes present, class 1 denoted by the blue circles, and class 2 denoted by the green triangles. The optimized hyperplane is displayed as the solid red line. The support vectors are the points in both classes which are shaded darker and are equidistant from the optimized hyperplane and separated by the maximum distance. There is one support vector from class 1 and two support vectors from class 2.

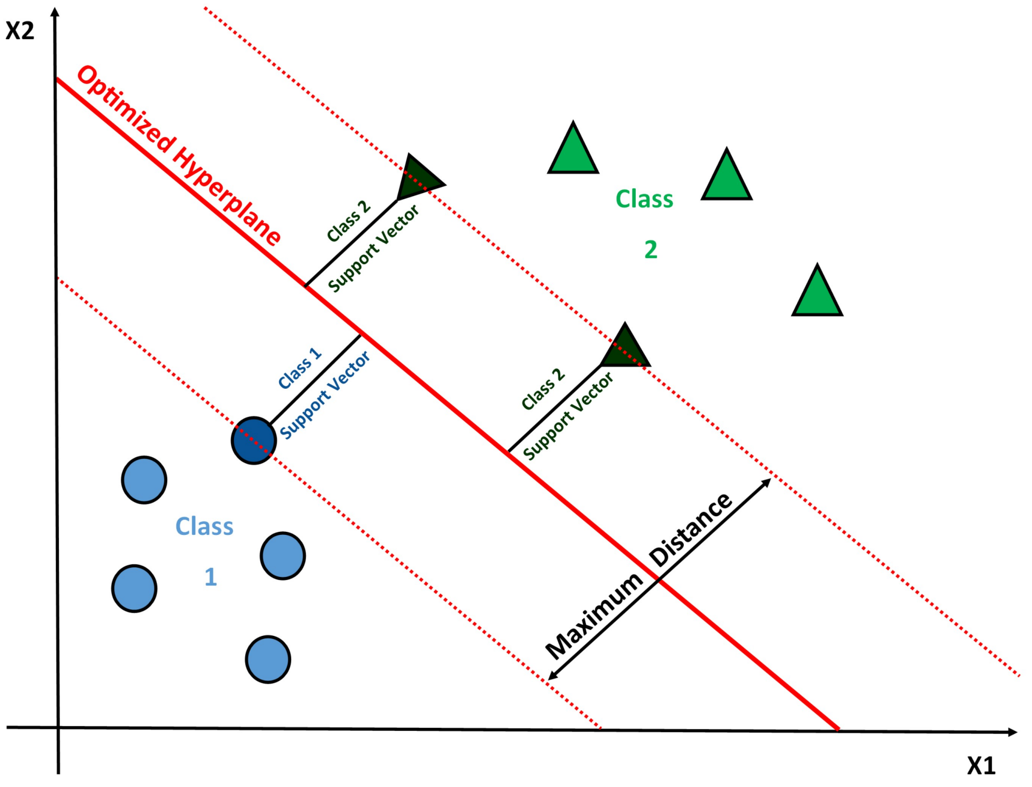

Figure 2.5: SVM Optimized Hyperplane

There are several advantages associated with the use of SVM. Firstly, this method is very effective in high dimensional spaces. This is extremely advantageous when considering the growing data demand and the emergence of numerous big data applications. Additionally, this method is additionally efficient as it only requires the support vectors to create the model. This has incredible benefits as it can reduce a massive quantity of data to a much smaller and more feasible subset thus saving computational and memory resources in the process. Furthermore, SVMs are capable of incorporating the use of kernels to address non-linear decision boundaries and hyperplanes, thus making them suitable for non-linear and multi-class classification problems.

Unfortunately, there are a few drawbacks to the use of SVM which must be considered. Firstly, while SVM is able to address the multi-class classification problem, it is much less efficient when considering a multi-class, multi-output classification problem due to the inherent complexity of the problem. Additionally, the ratio between the number of features and the number of input training data points can lead to over-under fitting if not addressed through properkernel manipulation.

2.5.2 Statistical Learning

Statistical learning models are a set of machine learning models which heavily rely on the probabilistic models and structure of the data. These models in general predict a probability by which a certain point belongs to a given class in contrast to other models which may simply output a prediction. These methods are effective when considering risk assessment as the strength of the prediction can be determined. An example of this is when considering the predictions listed in Table 2.1 with their associated probabilities.

Table 2.1: Statistical Learning Probability Strength

Prediction	Class	Probability
1	1	0.86
2	2	0.65
3	2	0.96
4	1	0.54

Assuming the above results were obtained when considering the classification of points between two classes several observations can be made in terms of prediction strength. By examining the probability associated with each prediction, it can be stated that predictions 1 and 3 are strong predictions and specifically prediction 3 is the strongest prediction. Conversely, when considering the prediction probability of predictions 2 and 4, these are much weaker and prediction 4 especially, is the weakest. This comes into consideration when stating the confidence of a prediction. When machine learning models are implemented to solve problems where misclassification has significant consequences, the prediction strength becomes crucial as it translates to a certain confidence level which can be used when performing risk mitigation and risk analysis.

There are several statistical learning algorithms however, the most recognizable one is the naïve Bayes classifier. This algorithm makes one significant assumption when making a prediction and determining the associated probability which can be extremely detrimental to the performance of the system. The naïve Bayes classifier assumes full independence between features with respect to the output variable. This however, is generally untrue and can significantly hinder the performance of the algorithm. Despite this assumption, naïve Bayes has performed well when applied to complex problems however it still underperforms when compared to other algorithms and approaches.

2.5.3 Instance-Based Learning

Instance-based learning is a set of learning algorithms which are generally less computationally intensive during training, however, more computationally intensive during prediction when compared to other families of algorithms. These algorithms by definition do not construct complex models of the system but rather store certain instances which were observed throughout the training process. The most notable algorithm stemming from the class of all instance-based learning algorithms is the nearest neighbour algorithm commonly refered to as k-Nearest Neighbours (kNN). In essence, this family of algorithms posits the hypothesis that data points of a specific class will be in close proximity to other data points of the same class. When predicting the label of a data point, this algorithm attempts to find a predefined number of neighbours closest to the point in question and predict its class based on the class of its neighbours. One of the main advantages of this family of algorithms is its ability to address multi-class, multi-output problems as well as the ability of this algorithm to suppress noise through the tuning of the number of neighbours considered. The tuning of the number of neighbours considered however, can have a significant impact on the predictive performance of the model as considering too many neighbours can result in weaker boundaries between classes. This concept is displayed in the following figures where the impact of changing the number of neighbours on the boundaries between the classes is clearly visible.

Fig 2.6 represents a model which only considers the single closest neighbour when deciding the class label of the point to be predicted. Through the boundaries presented in this figure we can see that there are several misclassified points (coloured points appearing in the wrong coloured boundary) especially in the green boundary. This is caused by a noisy point which has caused the green boundary to roughly project into the blue boundary.

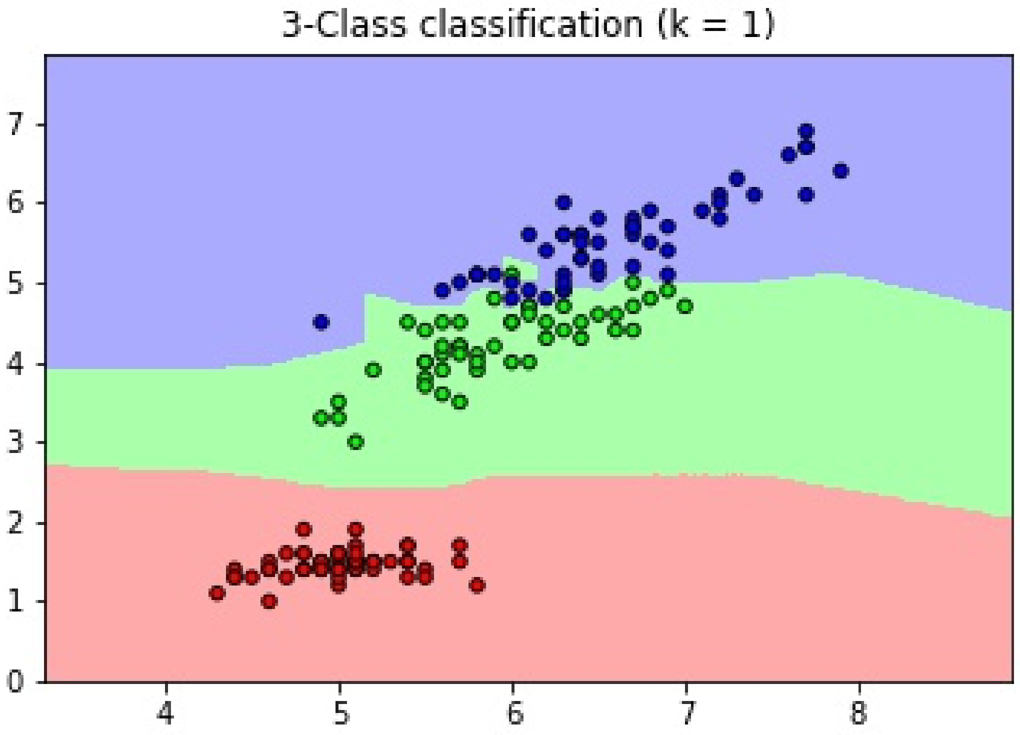

Figure 2.6: kNN 3-Class Classification k=1

Fig 2.7 represents the class boundaries upon increasing the number of neighbours considered by a factor of 2. In this figure we can see the previously observed sharp interjections of the green boundary into the blue boundary begin to be smoothed. Furthermore, this model shows a more generalized solution as certain outliers (most notably the isolated blue point) is not impacting the expansion of the entire blue class boundary as previously observed with one neighbour.

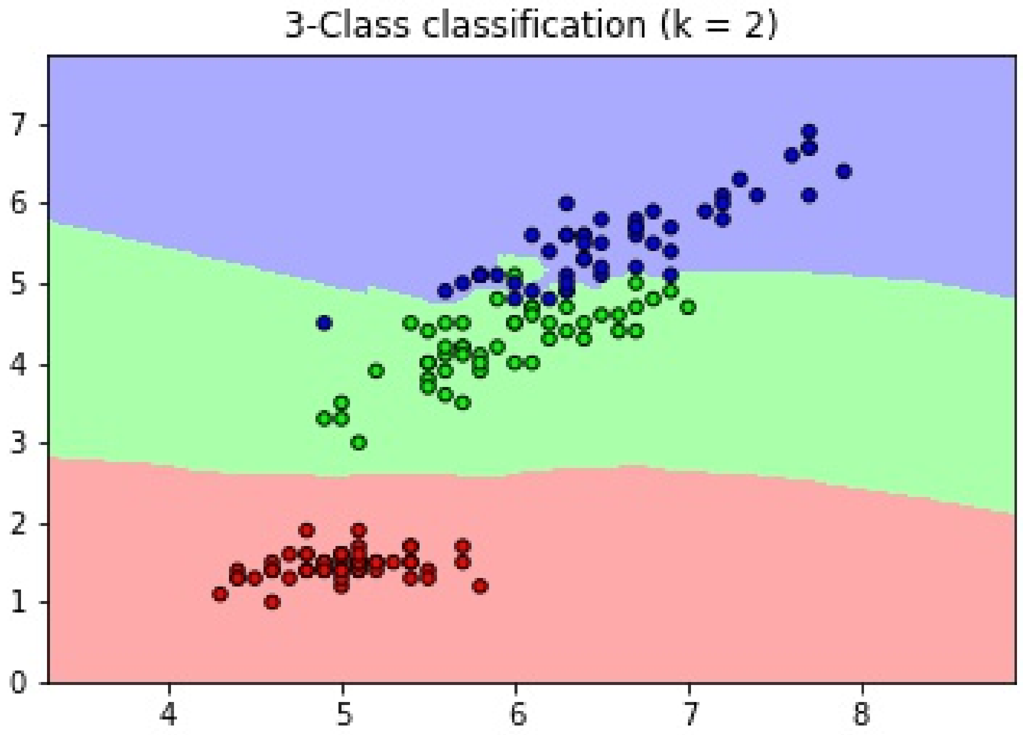

Figure 2.7: kNN 3-Class Classification k=2

Fig 2.8 displays the results when further increasing the number of neighbours considered by a factor of 2. In this figure we see the boundaries between the green and blue class be further smoother and the elimination of a significant protrusion of the green class boundary into the blue class boundary previously observed.

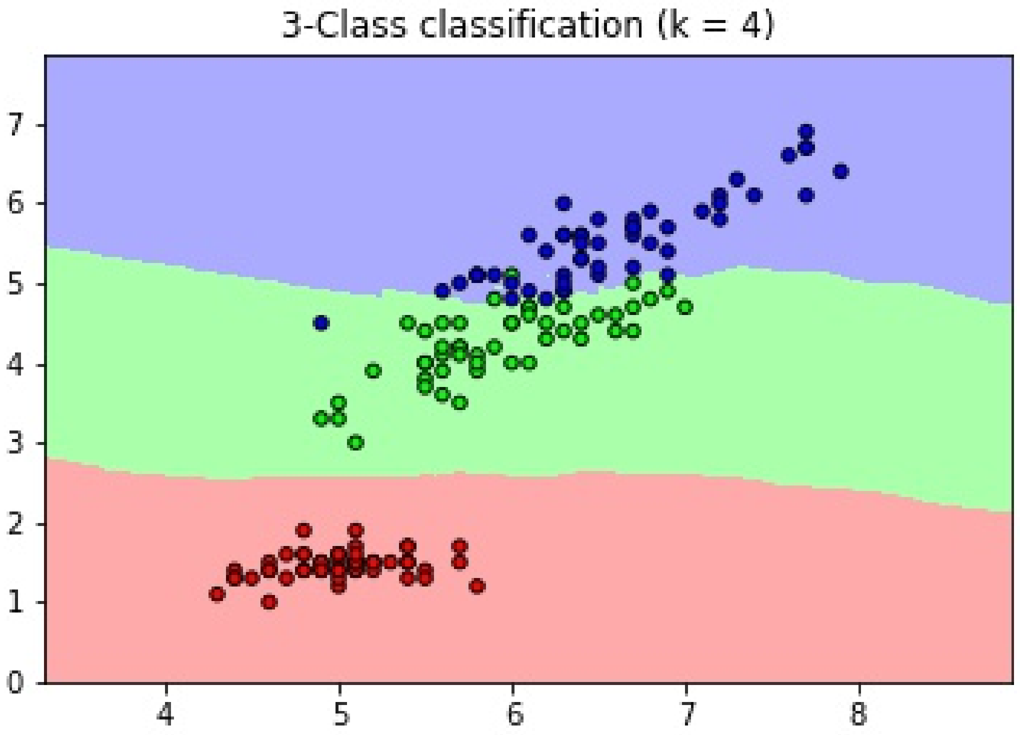

Figure 2.8: kNN 3-Class Classification k=4

Fig. 2.9 displays the result of increasing the number of neighbours considered by a factor of 2 again. In this figure we see the continual smoothing of the class boundaries and the mitigation of noise.

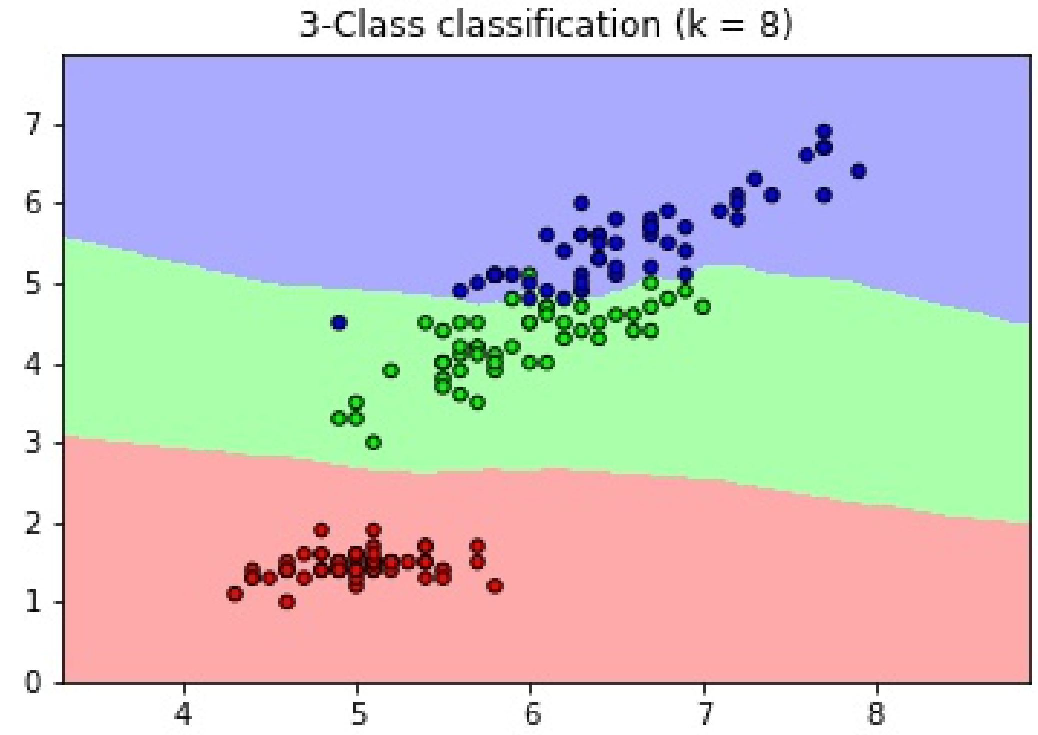

Figure 2.9: kNN 3-Class Classification k=8

However, this process of increasing the number of neighbours considered for the mitigation of noise has a serious tradeoff which is seen in Fig 2.10 . In this figure, the number of neighbours considered has increased substantially to 100. Due to the layout of the data points and the distribution of the classes, we can see that the class boundaries have been greatly distorted and the blue boundary has almost been completely eliminated. This is the manifestation of overfitting as this model is not generalizable and is severely hindered by poor predictive performance. Care must be taken when optimizing the number of neighbours selected for consideration in this algorithm as the noise suppression vs. generalization tradeoff is of paramount consideration.

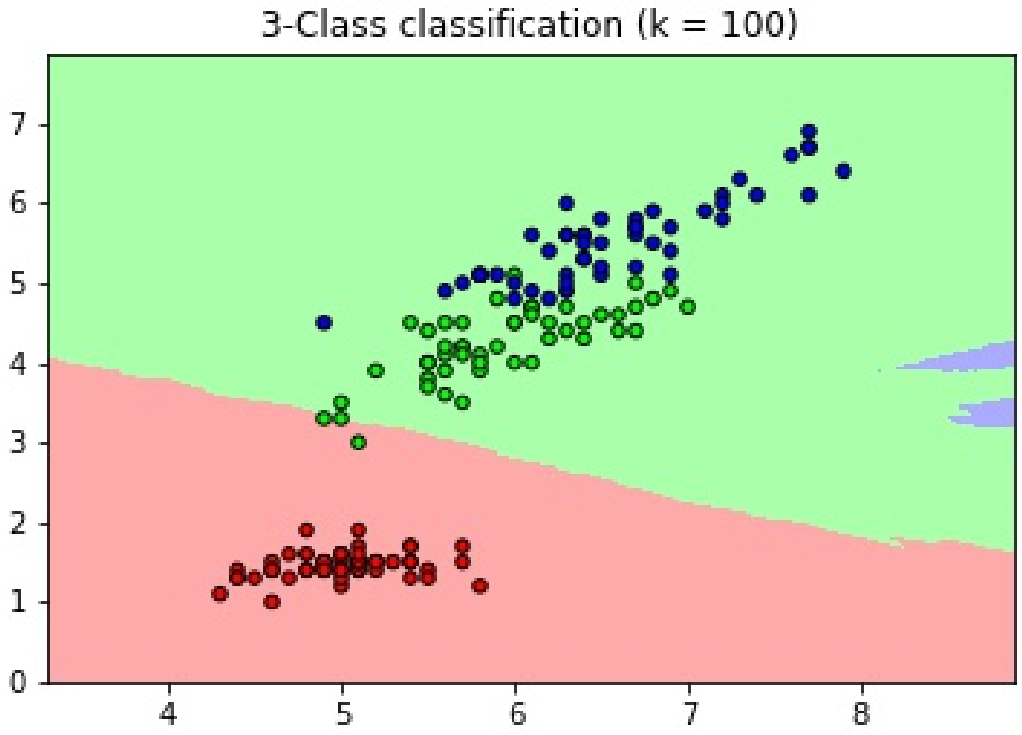

Figure 2.10: kNN 3-Class Classification k=100

2.5.4 Perceptron-Based

The perceptron is the fundamental building block of an Artificial Neural Network (ANN). Each perceptron can individually address linearly separable data however, an individual perceptron is unable to handle non-linearly separable data. To mitigate this problem, the grouping of multiple perceptrons in a single layer and the stacking of multiple layers of these perceptrons has been proposed thus forming an ANN. The basic structure of an ANN consists of an input layer, a number of hidden layers, and an output layer as shown in Fig. 2.11. As seen in this figure, each perceptron is fully connected with every perceptron in the adjecent layers. These connections are know as links.

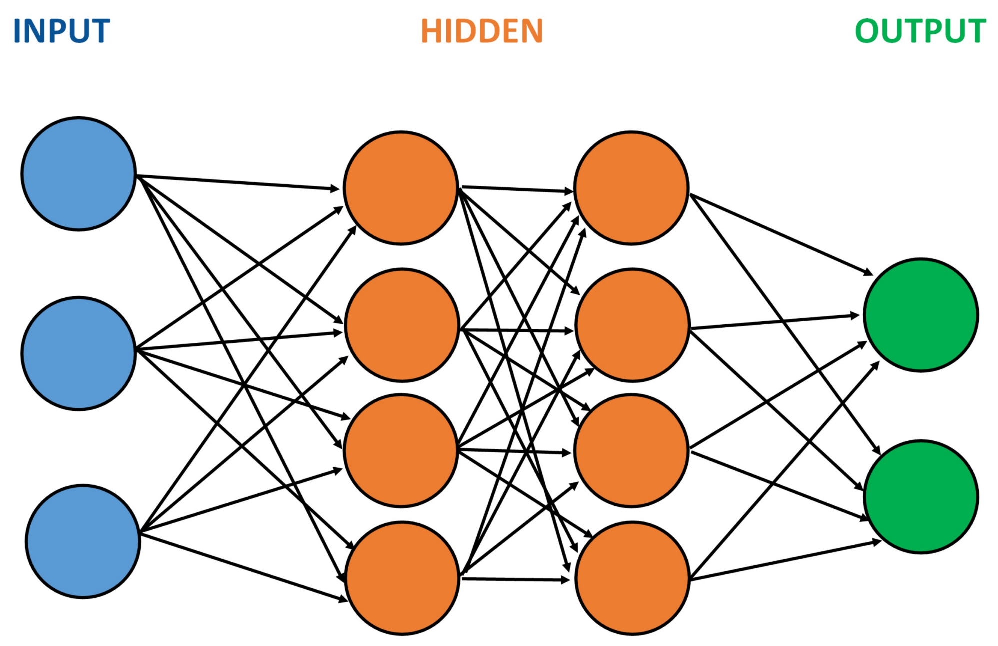

Figure 2.11: ANN Architecture

During the training of these networks, a set of weights associated with each link between perceptrons is adjusted such that the mapping between inputs and outputs is made. Throughout the training phase, certain perceptrons are given greater weight indicating that they are significant in the mapping of input to output. This network mimics the human brain in the sense that perceptrons which are weighted higher correlate to neurons firing in the brain. As such, these networks are commonly referred to simply as neural networks.

The main advantages associated with the use of ANNs is the ability to model both linear and non-linear relationships between the data. This enables the ability of being able to address problems of increasing complexity. Due to the way these models are trained, they can often generalize well during the building of the model. Additionally several techniques including L1 and L2 regularization are available during training to further improve the generalization of the model and reduce the amount of under and over fitting.

The main disadvantage of ANNs is the lack of explainability when assessing the behaviour of the network. Since the network is composed of weights and perceptrons, explaining why certain perceptrons are given higher weights in an intuitive way can be quite a daunting task. Additionally, the training of ANNs, depending on the amount of data and nature of the problem, can take a long time and is very hardware dependent. The training of this type of network is often ineffective for implementation in time-sensitive operations especially in the event of a fault or failure requiring the retraining of the entire network.

2.5.5 Logic-Based

The set of logic-based machine learning algorithms includes the popular tree family algorithms. These algorithms attempt to create rules based on the features of the dataset. When predicting the label of a new point, these rules are tested and a label which satisfies all the rules tested is returned. These algorithms have been extremely successful due to their simplicity, understandability, and their ability to be visualized. Further advantages of decision trees are realized though the low cost of building the tree and its ability to address multi-output, multi-class classification problems. Unfortunately, there are several disadvantages associated with decision trees which must be considered during implementation. Firstly, decision trees are subject to overfitting which can be detrimental to the predictive performance of the model. Additionally, if there is a significant class imbalance, the trees often exhibit bias towards the majority class thereby hindering the overall performance of the model. Being the main algorithm used in this work, decision trees will be further examined in greater detail throughout the duration of this work.

Bibliography

[1] INTEL, “Evolved Packet Core (EPC) for Communications Service Providers,” INTEL,May 2016.[Online].Available: https://builders.intel.com/docs/networkbuilders/Evolved-packet-core-EPC-for-communications-service-providers-ra.pdf

[2] H. Hawilo, M. Jammal, and A. Shami, “Exploring Microservices as the Architecture of Choice for Network Function Virtualization Platforms,” IEEE Netw., vol. 33, no. April, pp. 1–9, 2019.

[3] ETSI, "ETSI - Standards for NFV - Network Function Virtualisation | NFV Solutions," 2019. [Online]. Available: https://www.etsi.org/technologies/nfv.

[4] ETSI, “Network Functions Virtualisation (NFV): Architectural Framework.” ETSI, 2014.

[5] ETSI “Network Functions Virtualisation (NFV); Acceleration Technologies; Report on Acceleration Technologies & Use Cases” 2015. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/001/01.01.01_60/gs_nfv-ifa001v010101p.pdf

[6] Network Functions Virtualisation (NFV); Management and Orchestration, ETSI GS NFV-MAN 001 V1.1.1, 2014

[7] H. Hawilo, M. Jammal, and A. Shami, “Orchestrating network function virtualization platform: Migration or re-instantiation?,” Proc. 2017 IEEE 6th Int. Conf. Cloud Networking, CloudNet 2017, 2017.

[8] F. Carpio, S. Dhahri, and A. Jukan, “VNF placement with replication for Loac balancing in NFV networks,” IEEE Int. Conf. Commun., pp. 1–6, 2017.

[9] H. Moens and F. De Turck, “VNF-P : A Model for Efficient Placement of Virtualized Network Functions,” pp. 418–423, 2014.

[10] M. M. Tajik, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, “Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining,” IEEE Trans. Netw. Serv. Manag., vol. 16, no. 1, pp. 374–388, 2019.

[11] Q. Sun, P. Lu, W. Lu, and Z. Zhu, “Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement,” in 2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings, 2016.

[12] M. Dieye et al., “CPVNF: Cost-Efficient Proactive VNF Placement and Chaining for Value-Added Services in Content Delivery Networks,” IEEE Trans. Netw. Serv. Manag., vol. 15, no. 2, pp. 774–786, 2018.

[13] D. Zeng, L. Gu, Y. Chen, S. Pan, and Z. Qian, “Cost efficient state-aware function placement and flow scheduling for NFV networks,” Proc. - 2018 IEEE SmartWorld, Ubiquitous Intell. Comput. Adv. Trust. Comput. Scalable Comput. Commun. Cloud Big Data Comput. Internet People Smart City Innov., vol. 1, pp. 1352–1357, 2018.

[14] L. Ochoa-Aday, C. Cervelló-Pastor, A. Fernández-Fernández, and P. Grosso, “An online algorithm for dynamic NFV placement in cloud-based autonomous response networks,” Symmetry (Basel)., vol. 10, no. 5, pp. 1–18, 2018.

[15] L. Dinh-Xuan, M. Seufert, F. Wamser, P. Tran-Gia, C. Vassilakis, and A. Zafeiropoulos, “Performance evaluation of service functions chain placement algorithms in edge cloud,” Proc. 30th Int. Teletraffic Congr. ITC 2018, pp. 227–235, 2018.

[16] B. Addis, D. Belabed, M. Bouet, and S. Secci, “Virtual network functions placement and routing optimization,” 2015 IEEE 4th Int. Conf. Cloud Networking, CloudNet 2015, pp. 171–177, 2015.

[17] Ben Jemaa, G. Pujolle, and M. Pariente, “QoS-aware VNF placement optimization in edge-central carrier cloud architecture,” 2016 IEEE Glob. Commun. Conf. GLOBECOM 2016 - Proc., pp. 1–7, 2016.

[18] H. Hawilo, M. Jammal, and A. Shami, “Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud,” IEEE J. Sel. Areas Commun., vol. 37, no. 3, pp. 1–1, 2019.

[19] M. Mechtri, C. Ghribi, and D. Zeghlache, “VNF placement and chaining in distributed cloud,” IEEE Int. Conf. Cloud Comput. CLOUD, pp. 376–383, 2017.

[20] N. H. Tran, S. Ren, W. Saad, C. S. Hong, and C. Pham, “Traffic-aware and Energy-efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach,” IEEE Trans. Serv. Comput., pp. 1–1, 2017.

[21] S. Khebbache, M. Hadji, and D. Zeghlache, “A multi-objective non-dominated sorting genetic algorithm for VNF chains placement,” CCNC 2018 - 2018 15th IEEE Annu. Consum. Commun. Netw. Conf., vol. 2018–January, pp. 1–4, 2018.

[22] B. Kotsiantis, “Supervised Machine Learning: A Review of Classification Techniques,” Informatica, no. 31, pp. 249–268, 2007.

Chapter 3

Delay-Aware Tree (DAT) for VNF Placement

3.1 Introduction

The work outlined in this chapter presents the use of machine learning algorithms in the NFV orchestrator to predict initial VNF instance placement in a network while taking into consideration QoS guarantees.

3.2 Related Works

The following outlines some of the work being done in the field related to the use of machine learning in VNF provisioning, allocation, and function placement.

Khezri, *et al.* [1] propose a dynamic NFV service provisioning solution which incorporates deep Q-learning network and is reliability-aware. Their combined objective was the minimization of the NFV placement cost and the maximization of admitted services. Simulation results demonstrate the model's ability to learn during the training phase through a constant increase in service admission ratio.

Zhang *et al.* [2] propose an intelligent cloud resource management framework which encompasses both deep and reinforcement learning. In the proposed framework the mapping and allocation of resources is made by passing the application demand through a neural network which has been trained using a combination of a stacked auto encoder and reinforcement learning. The two constraints considered for allocation are QoS requirements and overall resource consumption.

Xu *et al.* [3] propose an application aware VNF deployed on a GPU server which analyzed packets and using deep learning classifies their application type using a deep neural network. Initially, when an application request is received the placement is made using the shortest path algorithm however, once information about the application type is received, the DFS algorithm is executed to identify a placement which meets the application requirements including acceptable delay and minimum bandwidth requirement.

Sun, *et al.* [4] propose a dynamic SFC deployment strategy using Q-learning. The goal of this algorithm is to maximize a profit function which is related to the number of successfully deployed SFCs. Their implementation suggests several paths which meet the SFC requirements and a load balancing algorithm ultimately selects the final path.

Riera *et al.* [5] suggest the use of reinforcement learning to solve the service mapping problem. Several reward functions could have been implemented to maximize different objectives including: acceptance rate, cost reduction, revenue generation, etc. however, this work decided to implement a reward function based on the number of successfully mapped VNF instances. Simulation suggest the reinforcement learning approach was able to learn network dynamics and improve allocation while still working within resource and service constraint.

Martin *et al.* [6] propose a resource allocator which uses machine learning for demand prediction for 5G networks. Their model takes into consideration QoS and QoE requirements when determining the most efficient resource allocation scheme given the predicted demand by means of a simulated annealing optimizer which selects the best topology given the demand forecast from the machine learning algorithm.

Mijumbi, *et al.* [7] propose a predictive framework which used machine learning to estimate the future resource requirements of a given VNF component. Their work implements a graph neural network and uses previous information about the VNF component to predict the future resource utilization. Their work only considers the scaling in and out of resources post-placement and assumes initial component placement has already occurred.

Wang *et al.* [8] propose a framework to enable dynamic service chain deployment and scaling through the use of online learning techniques. The main objective of their work is the minimization of the total cost of VNF deployment and operation. The framework uses a combination of the ski-rental algorithm for VNF provisioning and a multi-armed bandit learning technique for the actual placement.

3.3 Motivation

This section outlines the motivation behind this work specifically, the need for intelligence and machine learning, the paradigm shift from conventional to data-driven networks, as well as the benefits experienced through the use of machine learning.

3.3.1 The Need for Intelligence

As connectivity demands and network functions and capabilities increase, so does the amount of network-generated data. This rate at which this data grows and continues to grow demonstrates that it is high volume. Furthermore, this data must be processed faster and faster as new networking technologies are introduced, indicating high velocity. Finally, with many different applications and services, as well as the introduction of 5G nodes which will have in excess of 2000 configurable parameters [9], representing a significant increase from the current number of parameters, suggests that this network data also possesses high variety. These three attributes: volume, velocity, and variety unequivocally posit that network data generated from future networks will be ‘Big Data’ [10].

The need for intelligence however, is not solely manifested in the large quantity of generated data of future networks, it is also attributed to the increased service and functionality of said networks. In order to meet QoS guarantees and service requirements, future networks will need to observe the network environment, understand and adapt to unplanned scenarios, as well as optimize configurations [11]. These functionalities will be achieved by creating a network which is capable of automatic network optimization and provisioning through self-configuration, self-optimization, and self-healing [11][12].

The massive amounts of data generated coupled with the increasing network demands and functionality require the imminent implementation and leveraging of intelligence in existing and future networking technologies.

3.3.2 Why Machine Learning?

The term artificial intelligence has been broadly used throughout industry and academia to discuss the ability of a machine to sense, mine, predict, and reason with a given environment [11]. Machine learning specifically, is the first stage in this all-encompassing artificial entity as it gives a system the ability to make real-time decisions and predictions given previous conditions and data. In the realm of networking, there are several instances where functionalities can be decomposed into smaller subsets of prediction and classification including but not limited to: demand forecasting, network scheduling, and VNF placement. Such functionalities are candidates for the implementation of machine learning techniques.

Perhaps the greatest advantage and need for machine learning comes from its ability to create a general, transferrable model. Traditionally, there have been various network scenarios each with its own parameters, requirements, and attributes which requires a unique and independent solution [9]. This poses a certain limitation since creating individual models is quite intensive and inefficient. Machine learning on the other hand can create general models which are transferrable and adaptable to many scenarios.

Traditionally, when considering the modelling and analysis of a given system, a full system model is required. When considering this type of approach in a complex and dynamic networking environment, it is evident that the development of a full and complete analytical model would be incredibly difficult to the point of impossible due to the great volume of system parameters and uncertainties. Machine learning has excelled across all fields recently due to its ability to provide model estimations which perform within acceptable thresholds without the need exact analytical modelling of the system's behaviour [9]. This paradigm shift is known as data-driven networking and has been discussed as being a method of considering future networks.

3.3.3 Data-Driven Networks

As previously mentioned, traditional mathematical modelling of complex system behaviour can be quite difficult. This is especially true in the case of networking as system complexity constantly increases and NSPs are tasked with solving problems such as NFV resource allocation and VNF placement which are categorized as NP-Hard [13]. Solving these problems are computationally expensive and often require the use of near-optimal heuristic models. Furthermore, when considering networking problems, the use of problem decomposition where a large problem is decomposed into several smaller child problems which are then independently solved is quite problematic as the optimal solution of these sub-problems does not inherently result in the optimal solution of the initial problem [14].

By shifting from the notion of model-based networking to the paradigm of data-based networking several benefits arise. Firstly, an exact analytical network model is not required as the network parameters are directly learned from the data. Furthermore, this paradigm scales well with increasing network size and complexity as the generated data will provide the necessary foundation for the extraction of the updated system characteristics and behaviour.

3.3.4 Benefits of Machine Learning

As previously mentioned, with the exponential increase in demand and data, NSPs world-wide are facing challenges in terms of network operation including: performance, security, efficiency, management, and resilience [12]. Conventionally, these challenges have been addressed through analytical modelling, optimization problem formulations, heuristic solutions, time-series analyses, and statistical approaches [9]. These methods provide great insight and interpretability into how the system is viewed and how these results are obtained, something which has been a key criticism of machine learning models as they are commonly considered to be ‘black boxes’ and opaque. However, despite their interpretability they have several limitations which can be addressed through the use of machine learning.

Firstly, machine learning models have an incredible ability to reduce the complexity of a system due to their offline/online training and prediction capabilities. When considering most machine learning models, their most computationally intensive part is training of the model. This however, can be completed offline such that the much less complicated prediction stage is the only requirement during run-time. This enables many machine learning models to be implemented in real-time as this reduction in complexity overcomes the inability of conventional analytical models to achieve a solution in acceptable time.

Furthermore, the ability of machine learning to develop models which are robust and dynamic which can adapt to changing environments and conditions prove to be advantageous as model-based networking would require the development of a new system model and the creation of a new solution. This point can be extended to include the ability of a machine learning model to develop a generalizable solution capable of applying to many different topologies and situations.

3.3.5 Why Decision Trees?

There are three main reasons motivating the use of decision trees in this work. Firstly, machine learning models, as previously stated, have receive a lot of criticism for their opacity and ‘black-box’ quality. This however, is not the case when considering decision trees as the construction of the trees is a visible and interpretable group of binary decisions which lead to a predictive outcome. Second, the end goal of autonomous networking is a unified entity capable of sensing, observing, managing, and reason with a networking environment. Constructing this entity using a top-down approach however, is infeasible; by considering a bottom-up approach starting with simple models and specific tasks and gradually increasing model complexity through architecture and functionality is a sound approach to meet this end goal. Finally, the task of placing a given number of instances (multi-output) on a given number of network servers (multi-class) presents a prime opportunity for tree-based algorithms as they are able to address the family of complex multi-class, multi-output problems [15].

3.4 Methodology

This section outlines the various steps taken to design and implement the Delay Aware Tree.

3.4.1 Dataset Generation

The dataset used to train the machine learning model is generated by using an adaptation of the BACON algorithm presented by Hawilo *et al.* [13] to place VNF instances. This algorithm is selected for its ability to achieve near-optimal placement with significantly decreased computational complexity compared to the Mixed Integer Linear Programming model. The objective of the optimization model is to minimize the delay between two dependent VNF instances forming a SFC. The dataset contains placements from two network layouts. Below is a high-level formulation of the MILP optimization problem and a simplified pseudocode representation of the BACON algorithm as proposed in [13].

Objective:

minimize $(Delay\,between\,dependent\,VNF\,instances\,forming\,a\,SFC)$

Subject to:

1. *Availability Constraints*

- *The placement of a given instance on a given server is constrained to a binary integer value*
- *Each individual instance can only be instantiated on one server*
- *Dependent VNFs should be placed on separate servers given that the delay tolerences are not violated*

- *VNFs of the same type should be placed as far away from each other as permitted by the delay tolerence*

2. *Capacity Constraints*

- *The placement of a VNF instance on a given server should meet the VNF instance's CPU demand and not exceed the candidate server's CPU capacity*

- *The placement of a VNF instance on a given server should meet the VNF instance's memory demand and not exceed the candidate server's memory capacity*

3. *Network Delay Constraints*

- *Candidate servers must satisfy VNF latency requirements*

Algorithm 3.1 BACON [9]

```
1: Rank VNF component criticality
2: Divide components into sub-groups
3: Associate criticality with sub-group
4: Build network graph
5: Calculate betweeness centrality of servers
6: Sort servers in descending order with respect to betweenness centrality
7: Sort sub-groups in descending order with respect to criticality
8: Identify mediator VNF and median nodes
9: Place components on servers based on descending criticality and betweenness centrality
```

The first network generated represents a small scale network with 15 servers and 6 VNF instances while the second network represents a medium scale network with 30 servers and 10 VNF instances to place. The parameters of the network include server-to-server delay, server resources, VNF instance resource requirements, and VNF instance delay tolerances. They are all generated following the structure of a three-tier data center. The instances are then placed using the aforementioned BACON algorithm yielding the near-optimal placement [13]. This is conducted 10,000 times for each network, resulting in the placement of the VNF instances given the respective trial's network conditions. Table 2.1 lists the components of the two network models evaluated.

Table 3.1: Network Components

Component	Network 1	Network 2
Total Available Servers	15	30
MME VNF Instance	2	2
HSS VNF Instance	1	3
SGW VNF Instance	1	3
PGW VNF Instance	2	2

3.4.1.1 Dataset Distribution

The following figures will illustrate the distribution observed across the various features present in the dataset as well as the distribution of the output labels. These distributions were generated using the small-scale server network however, the same distribution is observed across the medium-scale network.

Fig. 3.1 illustrates the distribution of the first server resource as observed accross all 10,000 trials and across all 15 servers.

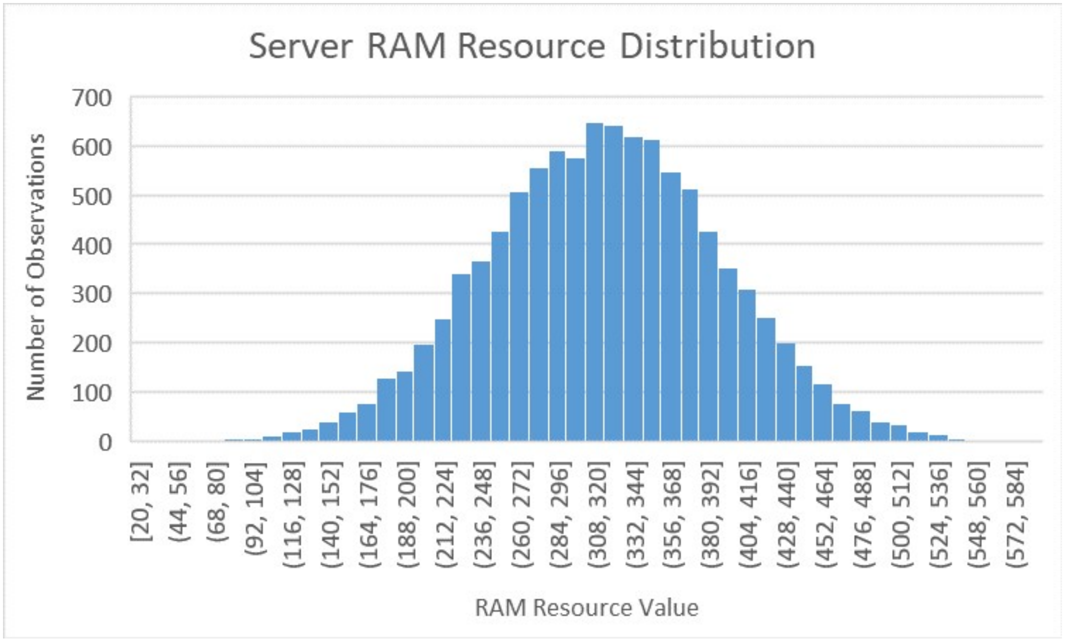

Figure 3.1: Server RAM Resource Distribution

Fig. 3.2 illustrates the distribution of the second server resource as observed accross all 10,000 trials and across all 15 servers.

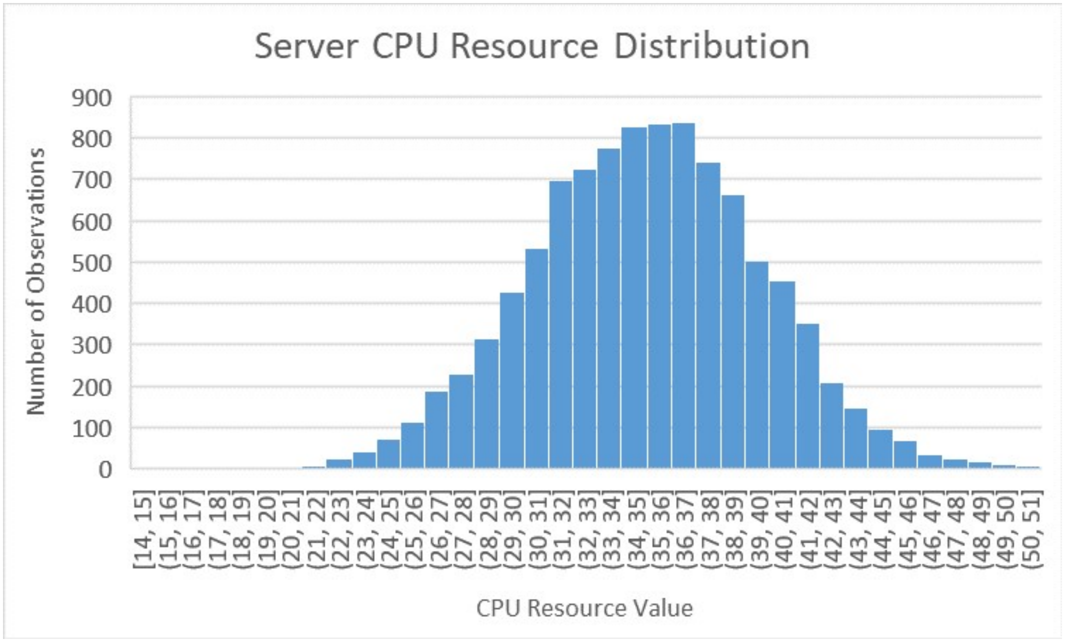

Figure 3.2: Server CPU Resource Distribution

Fig. 3.3 illustrates the distribution of the delay between servers as observed accross all 10,000 trials and across all 15 servers.

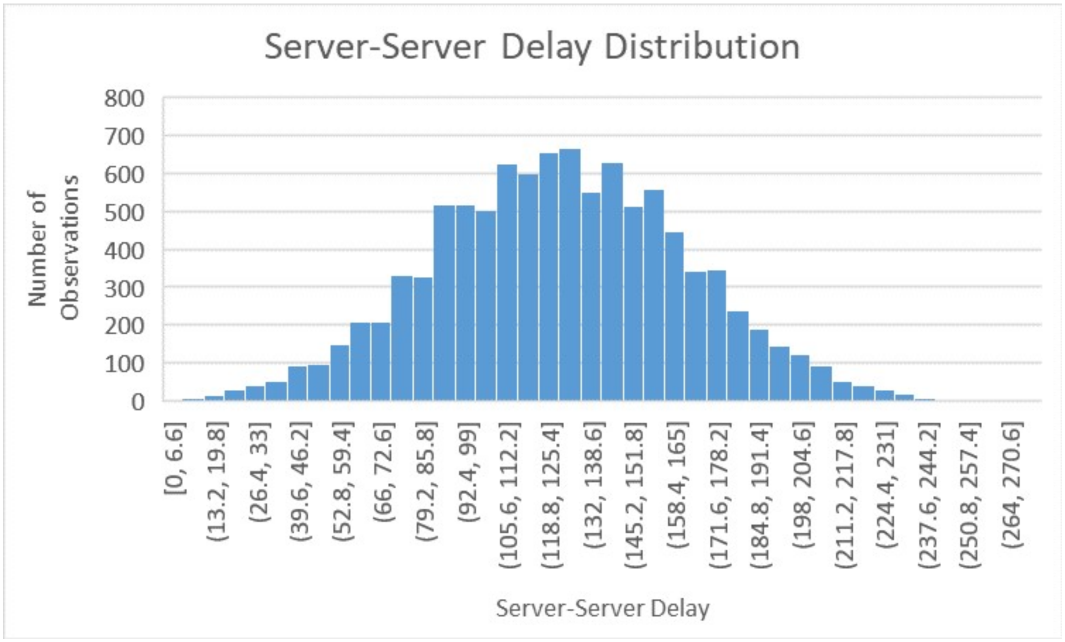

Figure 3.3: Server-Server Distribution

Fig. 3.4 illustrates the distribution of the delay tolerance of each instance as observed accross all 10,000 trials and across all 6 instances.

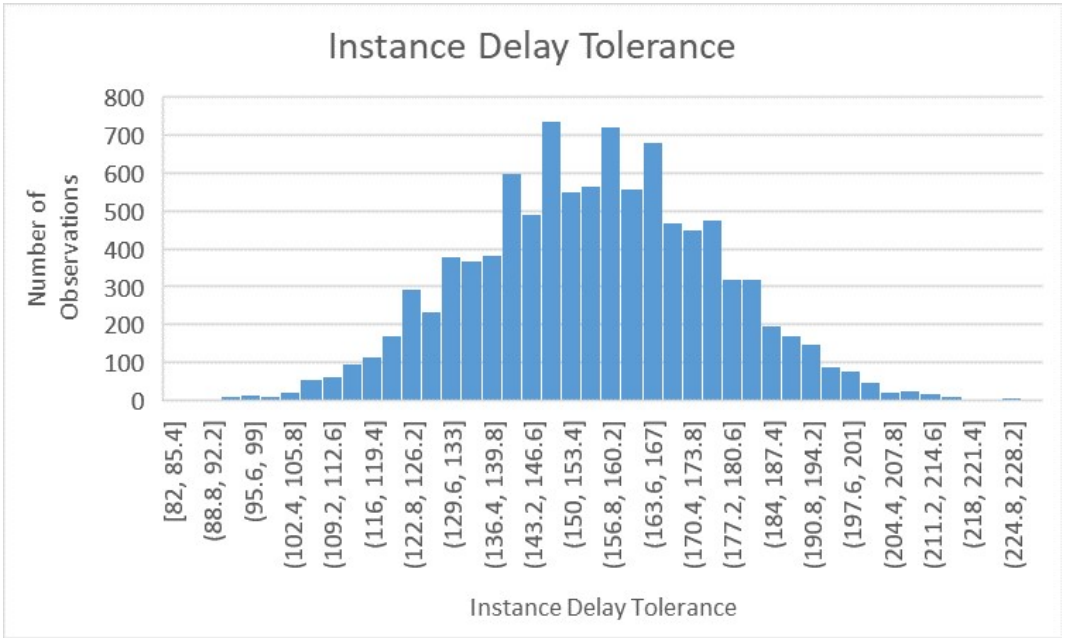

Figure 3.4: Instance Delay Tolerance Distribution

Fig. 3.5 illustrates the distribution of the first resource of each instance as observed accross all 10,000 trials and across all 6 instances.

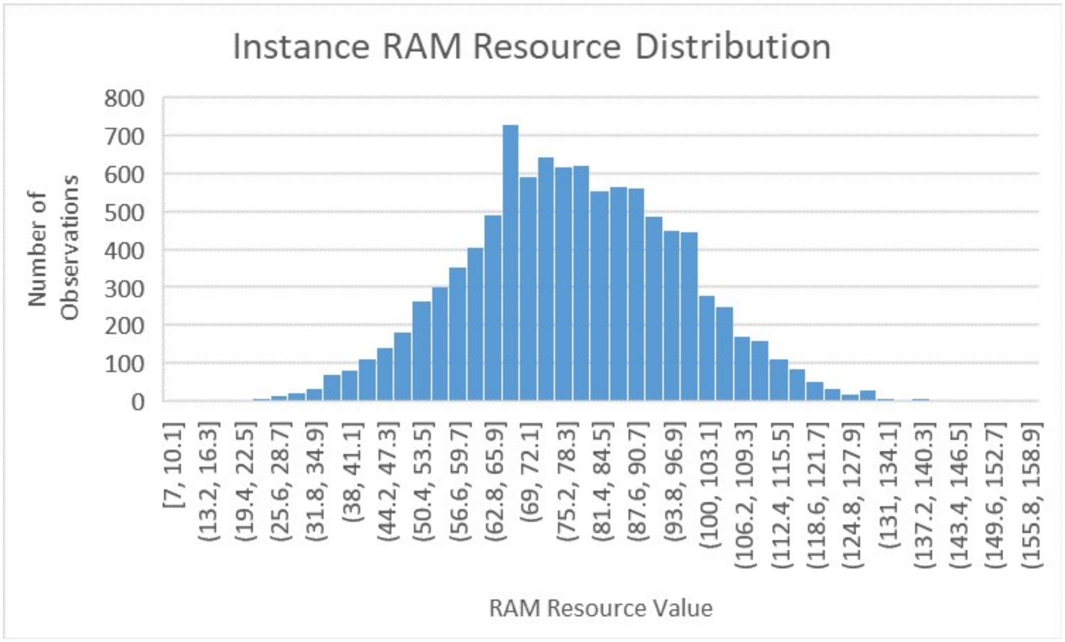

Figure 3.5: Instance RAM Resource Distribution

Fig. 3.6 illustrates the distribution of the second resource of each instance as observed accross all 10,000 trials and across all 6 instances.

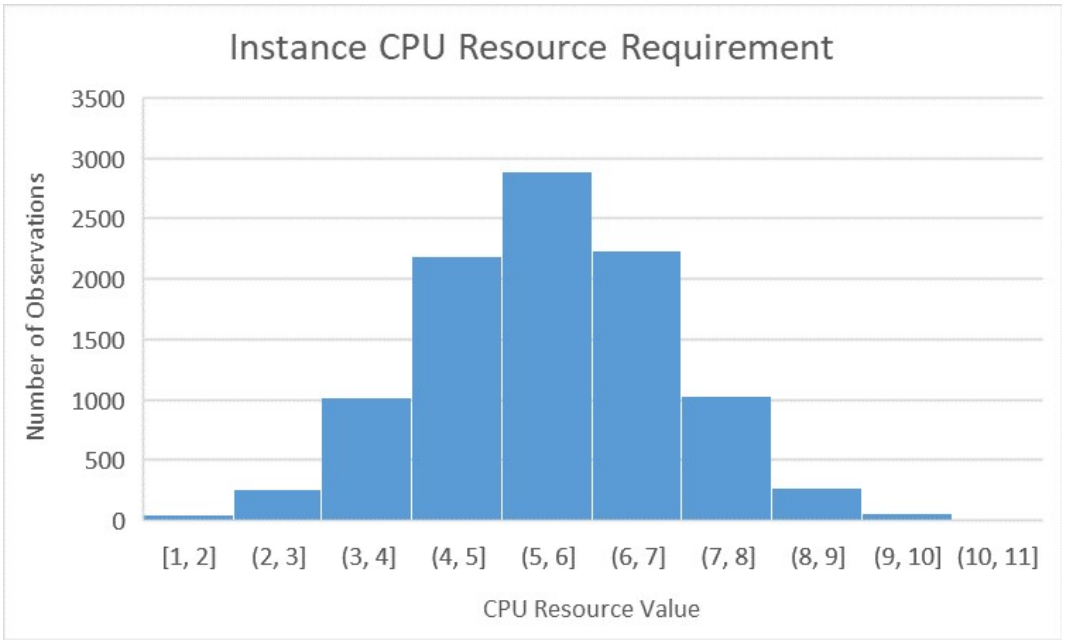

Figure 3.6: Instance CPU Resource Distribution

Fig. 3.7 illustrates the distribution of the output through the placement of each instance as observed accross all 10,000 trials and across all 6 instances.

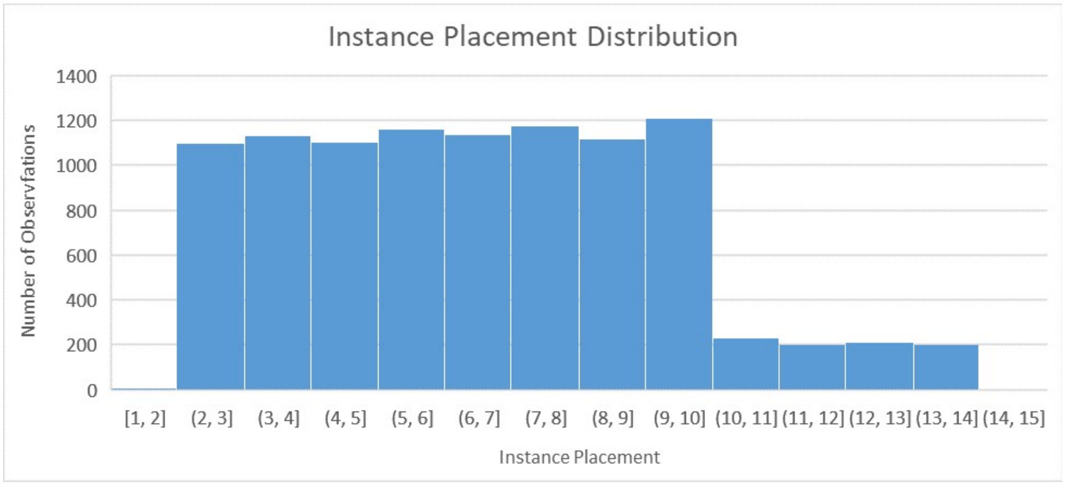

Figure 3.7: Instance Placement Distribution

3.4.2 Machine Learning Model

The features extracted from the previous dataset generation stage include: resource requirements for the VNF instances, resource capacity of the network servers, delay between servers, delay tolerance between dependent VNF instances, and component dependence. These features are used to predict the placement of each of the VNF instances on one of the network's servers. The machine learning model here can be treated as a multi-output, multi-class classification problem as there is a prediction for each of the VNF instances (multi-output) and each prediction selects a server from the set of network servers (multi-class).

Due to the nature of the problem, two approaches are evaluated, neighbour-based algorithms and tree-based algorithms. These two families of algorithms are selected for their ability to address the multi-class, multi-output requirement, something which many learning algorithms are unable to do due to the inherent complexity of the solution [15]. After training the model and performing a 10-fold cross validation test, it is determined that the tree based algorithms are the best performing family of algorithms specifically the decision tree algorithm.

The building of the decision tree follows a top-down approach starting with the root node. The goal of the tree is to purify the nodes by increasing the homogeneity of their associated samples. There are two main metrics used to assess the purity of the node, Gini index (1) and entropy (2) where the probability p_{mk} is the proportion of class observations at a given node [15]. The proposed decision tree uses the optimized Classification And Regression Tree (CART) algorithm [15]. This algorithm uses the Gini index by default for evaluating node purity.

$$Gini(X_m) = \sum_{i=1}^{k} p_{mk}(1 - p_{mk}) \tag{3.1}$$

$$H(X_m) = -\sum_{i=1}^{k} p_{mk}(\log_2 p_{mk}) \tag{3.2}$$

When initially constructing the tree, the Gini index of all features with respect to the output label is calculated. The feature with the lowest Gini index has the most purity and therefore is selected as a root node. Once the root node is selected the dataset is split into subsets depending on the feature values using thresholding. The Gini index of each of the features in each respective subset is then calculated and the lowest is selected as a branching attribute. This process is completed until homogenous leaf nodes (pure) are achieved or the maximum depth of the tree is reached. Since decision trees are prone to overfitting when dealing with large quantities of data, the maximum depth of the tree is set, essentially limiting the tree's ability to expand vertically, and 10-fold cross validation is used to further ensure overfitting does not occur during the training phase.

By using the data generated from the previous placement of VNF instances we create a data-driven network model whereby the algorithm is responsible for determining and extracting the inherent relationships from the data. Using this method, the complex mathematical modelling of the system is bypassed however, as demonstrated in the results section, the algorithm has learned from the training phase and is able to predict placement that approaches and outperforms the results obtained from the heuristic.

3.4.3 Time Complexity

When comparing the computational complexity of the two methods, the Delay Aware Tree (DAT) method exhibits a computational complexity $O(n_{features} \cdot n_{samples} \cdot \log n_{samples})$ when creating the tree and $O(\log n_{samples})$ when executing a query [15]. The original algorithm has a computational complexity of $O(\frac{s^3-s^2}{2})$ where s denotes the number of available servers in the network [13]. Comparing these two complexities, we can see that during runtime, the DAT method would operate at a lower complexity since its initial training phase would have already been completed and it would simply need to execute the query request.

3.5 Algorithm Performance Comparisson

In order to evaluate the results of the machine learning model it is compared to the BACON algorithm [13]. The performance is measured by calculating the delays between interconnected VNF instances once the placement is made. Furthermore, the overall delay of each computational path is also calculated.

3.5.1 Implementation Setup

The generation of the dataset is executed in Java while the data processing and machine learning models are implemented using Python. Both the generation of the dataset and the model implementation are run on a PC with an Intel ® Core™ i7-8700 CPU @ 3.20 GHz CPU, 32 GB RAM, and an NVIDIA GeForce GTX 1050 Ti GPU.

3.5.2 Results

The results of the simulations are displayed below. In terms of the small scale 15 server network Fig. 3.8 displays the delay between the various interdependent VNF instances.

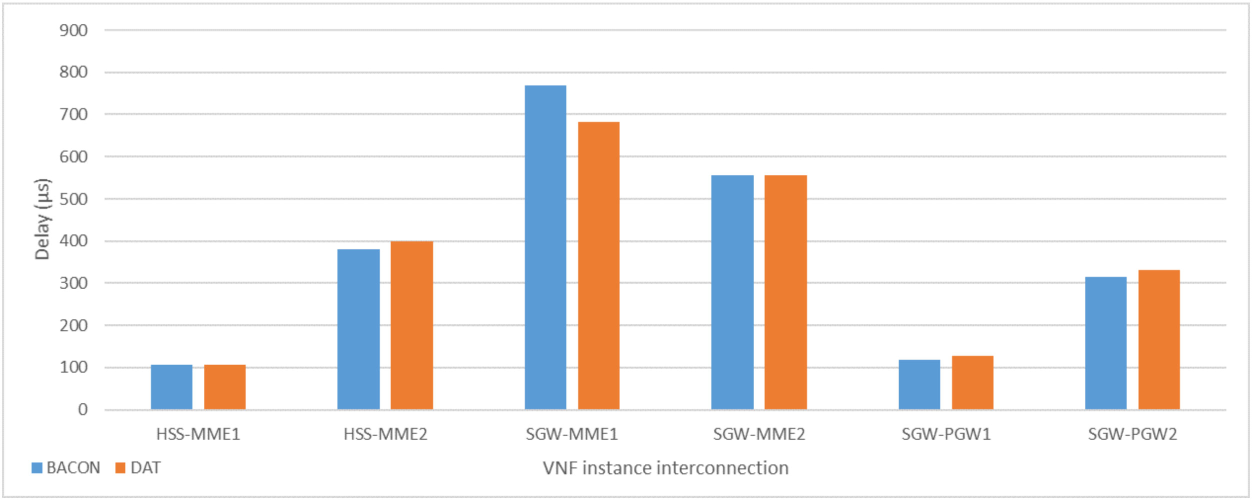

Figure 3.8: Delay between Interconnected VNF Instances

Fig. 3.9 presents the overall SFC delay across the 4 computational paths previously discussed.

As seen from these results, it is clear that the Delay Aware Tree (DAT) placement model has learned well from the near-optimal placement of the BACON algorithm. When comparing the delay between the interconnected VNF instances, DAT shows very good performance as the delays observed are very close to the delays observed through BACON's placement. In this case, it can be seen that DAT performs slightly better than BACON for the first two paths and slightly worse for the last two paths.

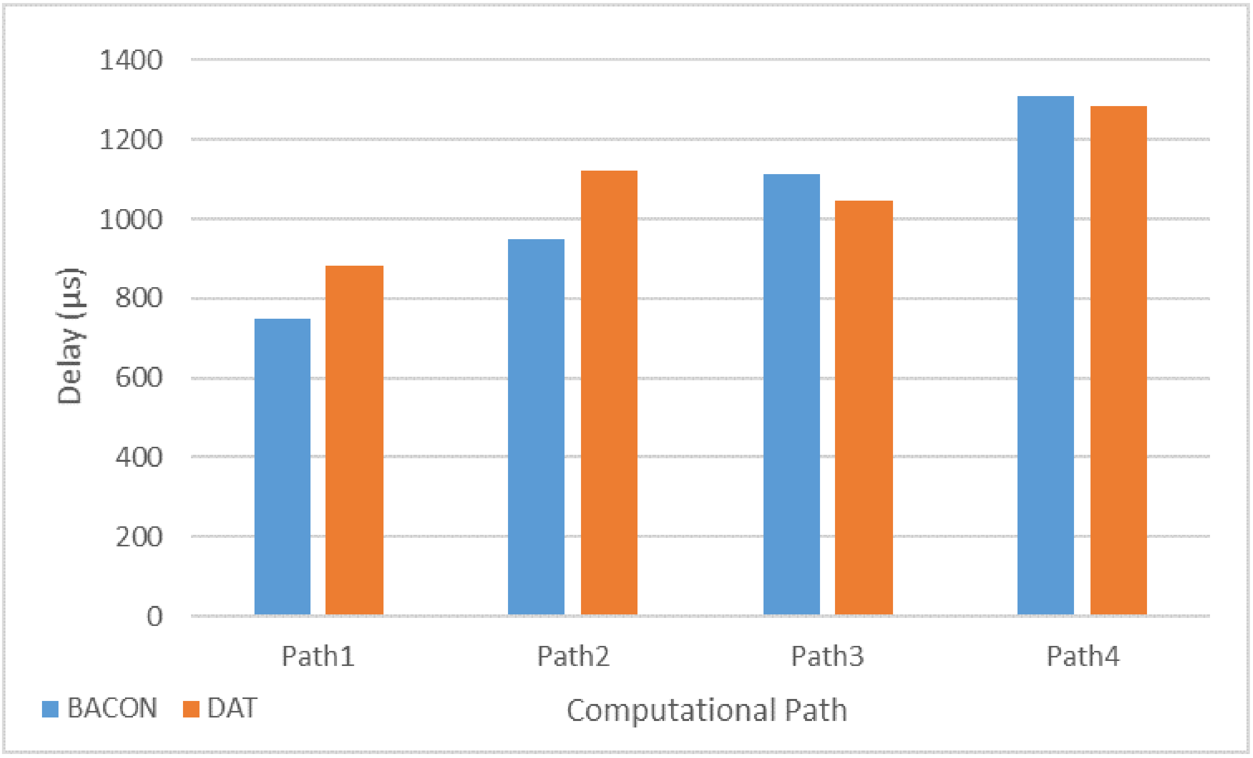

Figure 3.9: SFC End-to-End Delay

Fig. 3.10 shows the PDF of the difference between the delay of the computational paths using BACON and DAT. The mean of the distribution suggests that across all computational paths, BACON has on average 34µs less delay compared to DAT. However, the left tail of the distribution greatly skews the mean through outliers. These outliers can be labeled as invalid placement predictions and are caused by the fitting of the tree on the data. This problem can by mitigated through the optimization of hyperparameters associated with the mitigation of over and under fitting of a decision tree such as the max depth parameter. By optimizing the model hyperparameters the overall performance of a machine learning algorithm improves [16]. By improving the performance of DAT through hyperparameter optimization, the mean of the distribution will shift further towards the positive side and the tails of the distribution will be suppressed.

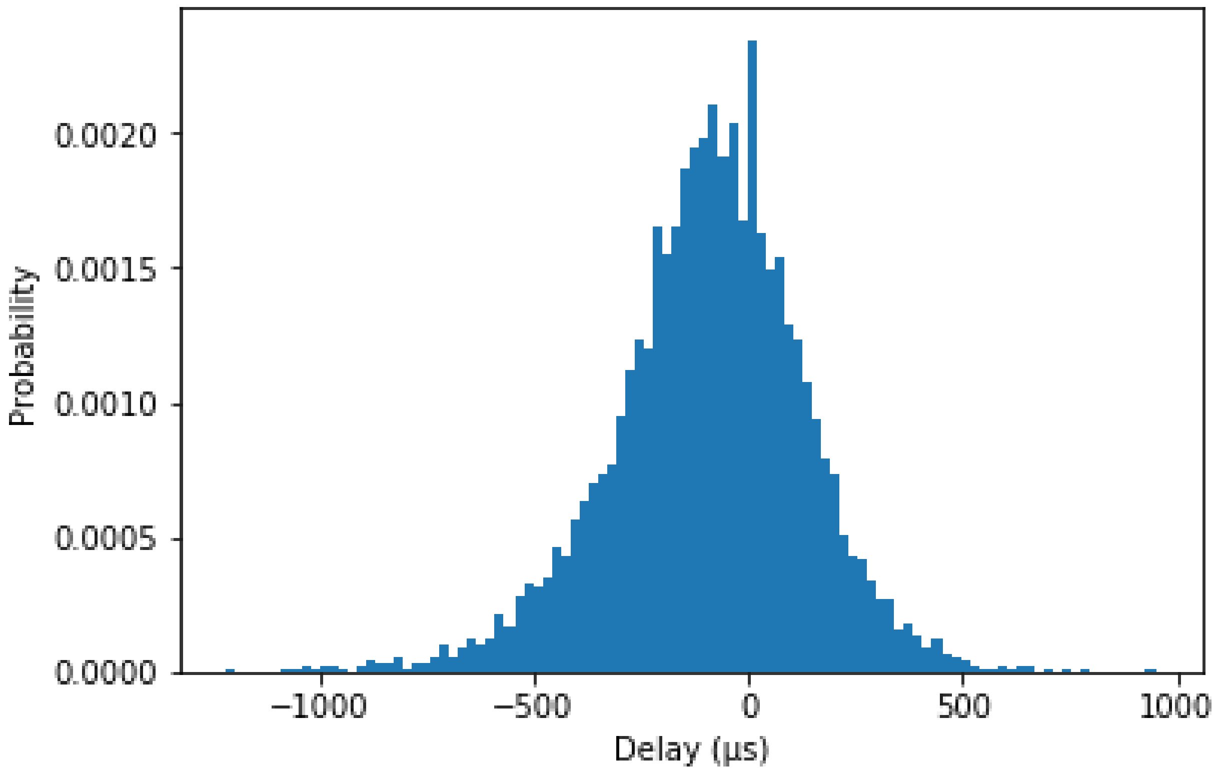

Figure 3.10: Delay Difference between Bacon and DAT per Computational Path

Fig. 3.11 displays the delay across the 36 computational paths in the medium 30 server network. As seen in the figure, DAT continues to perform well despite the increase in network size. Assuming a maximum allowable delay is imposed at 2000µs we can see that DAT successfully produces more computational paths that do not violate this threshold thereby increasing the resiliency of the network.

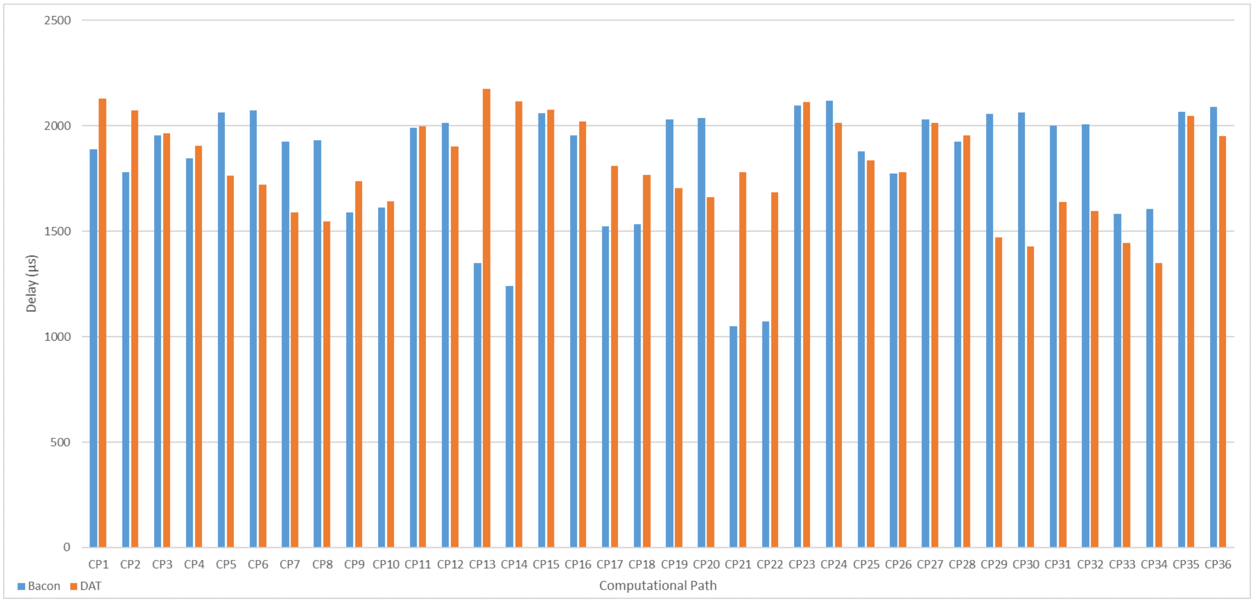

Figure 3.11: Computational Path Delay – 30 Server Layout

3.6 Conclusion

The work presented in this chapter describes the first step towards an implementable, intelligent, and delay-aware VNF placement strategy for the NFV Orchestrator. The Delay Aware Tree presented is able to predict the placement of VNF instances forming an SFC based on previous effective placements while considering operational constraints and ensuring QoS guarantees are met. On average, the DAT resulted in 34µs of added delay per computational path when compared to the BACON heuristic solution. Further work will aim to improve the predictive performance by reducing the average delay observed across all computational paths as well as the minimization of invalid placement predictions.

Bibliography

[1] H. R. Khezri, P. A. Moghadam, and M. K. Farshbafan, “Deep Q-Learning for Dynamic Reliability Aware NFV-Based Service Provisioning,” arXiv Prepr. arXiv1812.00737, 2018.

[2] Y. Zhang, J. Yao, and H. Guan, “Intelligent Cloud Resource Management with Deep Reinforcement Learning,” IEEE Cloud Comput., vol. 4, no. 6, pp. 60–69, 2017.

[3] J. Xu, J. Wang, Q. Qi, H. Sun, and B. He, “IARA: An intelligent application-aware VNF for network resource allocation with deep learning,” 2018 15th Annu. IEEE Int. Conf. Sensing, Commun. Networking, SECON 2018, pp. 1–3, 2018.

[4] J. Sun, G. Huang, G. Sun, H. Yu, A. K. Sangaiah, and V. Chang, “A Q-learning-based approach for deploying dynamic service function chains,” Symmetry (Basel)., vol. 10, no. 11, 2018.

[5] J. F. Riera *et al.*, “TeNOR: Steps towards an orchestration platform for multi-PoP NFV deployment,” IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conf. Work. Software-Defined Infrastruct. Networks, Clouds, IoT Serv., no. June, pp. 243–250, 2016.

[6] A. Martin *et al.*, “Network Resource Allocation System for QoE-Aware Delivery of Media Services in 5G Networks,” IEEE Trans. Broadcast., vol. 64, no. 2, pp. 561–574, 2018.

[7] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, “Topology-Aware Prediction of Virtual Network Function Resource Requirements,” IEEE Trans. Netw. Serv. Manag., vol. 14, no. 1, pp. 106–120, 2017.

[8] X. Wang, C. Wu, F. Le, and F. C. M. Lau, “Online Learning-Assisted VNF Service Chain Scaling with Network Uncertainties,” IEEE Int. Conf. Cloud Comput. CLOUD, vol. 2017–June, no. ii, pp. 205–213, 2017.

[9] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, “Machine Learning for Networking: Workflow, Advances and Opportunities,” IEEE Netw., vol. 32, no. 2, pp. 92–99, 2018.

[10] S. Suthaharan, “Big Data Classification: Problems and Challenges in Network Intrusion Prediction with Machine Learning,” Perform. Eval. Rev., vol. 41, no. 4, pp. 70–73, 2014.

[11] R. Li *et al.*, “Intelligent 5G: When Cellular Networks Meet Artificial Intelligence,” IEEE Wirel. Commun., vol. 24, no. 5, pp. 175–183, 2017.

[12] T. S. Buda *et al.*, “Can machine learning aid in delivering new use cases and scenarios in 5G?,” Proc. NOMS 2016 - 2016 IEEE/IFIP Netw. Oper. Manag. Symp., pp. 1279–1284, 2016.

[13] H. Hawilo, M. Jammal, and A. Shami, “Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud,” IEEE J. Sel. Areas Commun., vol. 37, no. 3, pp. 1–1, 2019.

[14] T. Wang, S. Wang, and Z. Zhou, “Machine learning for 5G and beyond: From model-based to data-driven mobile wireless networks,” China Commun., vol. 16, pp. 165–175, 2018.

[15] F. Pedregosa, R. Weiss, and M. Brucher, “Scikit-learn : Machine Learning in Python,” vol. 12, pp. 2825–2830, 2011.

[16] M. Feurer and F. Hutter, “Hyperparameter Optimization,” Encycl. Mach. Learn. Data Min., pp. 625–625, 2017.

Chapter 4

Depth-Optimized Delay-Aware Tree (DO-DAT) for VNF Placement

4.1 Introduction

With the incoming introduction of 5G technologies, automation in network management and orchestration is essential to meet requirements and connectivity demands. Machine Learning (ML) is a trending field which encompasses the use of various families of algorithms, both supervised and unsupervised, which learn from a given set of input (training) data and predict an output. Increasingly, ML has been adopted across many fields of research due to its ability to extract meaningful relationships from data, its ability to develop a general solution to a given problem, as well as its ability to adapt to changing environments.

The set of all machine learning algorithms exhibits a large range of functionality and complexity. When considering very basic algorithms, there are few parameters which affect the training and performance of a model; conversely, when considering more complicated algorithms, there are many parameters which affect the training and performance of a model. These parameters, referred to as hyperparameters, are extremely important when conducting the initial training phase of a model as they can influence the predictive performance of the model and introduce (or eliminate) unwanted qualities inherently present during the training phase including under-fitting and overfitting which hinder the model's performance.

When considering the decision trees, specifically CART, there are several hyperparameters which can be optimized to improve model performance. Table 4.1 outlines four critical hyperparameters related to the CART decision trees, a brief description of what each hyperparameter controls, as well as the type of values which they can assume.

Table 4.1: Main Hyperparameters for CART Decision Trees

Hyperparameter	Description	Value Type
max_depth	maximum depth of tree	integer
min_samples_split	the minimum number of samples to split a node	integer
		float
min_samples_leaf	the minimum number of samples required to form a leaf node	integer
		float
max_features	the number of input features to use when building the tree	integer
		float
		string

The work presented in this chapter is an extension of the work presented in Chapter 2 dealing with the use of machine learning for the delay-aware prediction of VNF instances forming a SFC. In the previous chapter, the Delay Aware Tree (DAT) which shows promising results when compared to current heuristic solutions was proposed. When considering the predicted placements exhibited by the DAT, some invalid placement predictions were observed. These invalid placement predictions are manifestations of the fitting of the machine learning model on the training data. The predictions were deemed invalid due to their violation of the constraints outlined in the initial MILP optimization problem formulation, namely the violation of the availability and delay constraints. In order to mitigate this, the maximum depth of the tree, a key hyperparameter in terms of the under and overfitting of a decision tree has been selected as the target for improving the performance of the DAT placement strategy through the minimization of invalid placement predictions and the continual minimization of the delay between interdependent VNF instances forming a SFC. In this chapter, the DAT model is further considered by optimizing the tree depth hyperparameter and proposing the Depth-Optimized Delay-Aware Tree (DO-DAT) placement strategy. The optimization presented in this work was done using the meta-heuristic Particle Swarm Optimization (PSO) algorithm.

4.2 Related Works

The tuning of machine learning model hyperparameters has been proven to increase the model performance.

The work of Mantovani, *et al.* [1] deals with the tuning of hyperparameters in decision tree algorithms. Their work uses several techniques including: random searches, heuristics and meta-heuristics. They evaluate the performance of each algorithm comparatively using accuracy as their main objective. Furthermore, they identify that when using the CART decision tree, the minimum bucket and the minimum split are the two most important hyperparameters.

Sureka and Indukuri [2] explore the optimization of both tree-based model selection and hyperparameter tuning of machine learning models using the meta-heuristic genetic algorithm. Their setup uses the predictive accuracy of the model as a metric of fitness to be maximized. Their work concludes that the genetic algorithm functions well as a meta-heuristic optimization method for both model type and hyperparameter setting however, it does not guarantee a convergence to a global optimum in the search space.

Stiglic, *et al.* [3] apply a method of virtual tuning to optimize the hyperparameters of decision tree algorithms based on the visual interpretability of the resulting model. Their study does not use any predictive performance metric as a basis of model fitness. Their methodology was implemented on various bioinformatics datasets and results show that the construction of simple, visually constrained models achieved comparable predictive performance to the default setting of a visually unconstrained decision tree.

Thornton, *et al.* [4] suggest the use of Bayesian optimization for the combined model selection and hyperparameter optimization problem. Their method was evaluated across several benchmarking datasets and their work evaluated the results of various classification models in terms of predictive accuracy. Results suggest their methodology outperforms traditional methods of individually considering model selection and hyperparameter optimization. Furthermore, their analysis shows significant overall improvement when compared to exhaustive grid-based search methods.

4.3 Methodology

The following section will outline the various stages which lead to the development of the DO-DAT including: problem formulation, data generation and analysis, as well as model construction and validation.

4.3.1 Problem Formulation

The problem formulation for this work was conducted in a two-fold manner; the first dealing with the problem formulation of the DAT and the second dealing with the problem formulation of the PSO depth optimization.

4.3.1.1 Delay-Aware Tree (DAT)

The methodology behind the construction of the DAT as defined by the previous work outlined in Chapter 2 and in [5], takes the previous placements made by the near-optimal heuristic BACON algorithm. Inherently, the problem formulation for the DAT follows the problem formulation for the BACON algorithm outlined in the work of Hawilo *et al.* [6].

4.3.1.2 PSO Optimization

The PSO optimization algorithm belongs to the set of meta-heuristic bio-inspired algorithms which converge to a solution based on the creation and updating of populations. Just as in nature, in order for a population to progress it must be fit, the same applies for this optimization method. With PSO, the population is formed by particles which move throughout the predefined search space with a given velocity. Initially, each particle is given a random position and velocity. Through each iteration, the position of each particle is revaluated based on its associated velocity in conjunction with the best observed position of the population. This continues based on the predefined number of iterations and the historical optimum is reported [7][8]. The following is a pseudocode representation of the PSO algorithm.

Algorithm 4.1 Particle Swarm Optimization

```
1: initialize particle position
2: initialize particle best known position
3: while end criterion not met do
4:     calculate fitness value of each particle
5:     compare with best fitness value and update if necessary
6:     identify particle with best fitness value as swarm's best position
7:     update particle velocity
8:     update particle position
9: end while
```

The PSO is part of the evolutionary meta-heuristic family of algorithms which also contains the popular Genetic Algorithm (GA). Both PSO and GA converge to a solution through the evolution of populations across function iterations. Despite the similarities between these two algorithms, PSO was selected due to its parallelized nature. The work of Hassan, et al. has suggested that while both the PSO and GA converge to a solution with a very quality index (proximity to known optimal solution) when evaluated across several different optimization problems, the PSO algorithm significantly outperforms the GA in terms of efficiency as it generally converges to a solution with fewer iterations [9].

4.3.1.3 PSO Depth Optimization

The PSO depth optimization was conducted through the development of a unique optimization function related to the domain of the NFV-enabled network and irrespective of the predictive accuracy of the model. By adopting this process, it is possible to move past the point of matching the performance of the BACON algorithm and instead focus on the continual development of the predictive placement model as a whole.

When considering the construction of a decision tree, the maximum depth of the tree has been identified as a key hyperparameter in the overall fitting of the model. In an effort to prevent over and underfitting, this work presents a joint optimization objective which considers both the average delay across all computational paths of a predicted placement as well as a penalty factor related to fitting. In the previous construction of the DAT improper model fitting manifested itself through invalid predictive placements which violate certain constraints outlined in the initial BACON optimization problem. The penalty factor term operates like a regularization term in the objective function penalizing invalid predicted placements during the training phase of the model.

The generic formulation of a multi objective optimization problem consolidated into a single objective function is defined below.

Given the set of hyperparameters h , the set of objectives f , and the importance vector of each objective expressed through the set of weights w, we can consolidate the multi-objective optimization problem into a single objective function O [10]. The evaluation criteria E is an expression of the consolidated objective function, the model trained with the set of hyperparameters h, the training set Ts and the validation set Vs [1]. Assuming E is formulated as a cost function, the optimization problem when implementing a b-fold cross validation is the minimization of $P(h)$.

$$h = \{h_1, h_2, ..., h_n\}$$

$$f = \{f_1, f_2, ..., f_n\}$$

$$w = \{w_1, w_2, ..., w_n\}$$

$$O = w_1 f_1 + w_2 f_2 + ... + w_n f_n$$

$$E(O, model(h), T_s, V_s)$$

$$P(h) = \frac{1}{b}\sum_{i=1}^{b} E(O, model(h), T_s^{(i)}, V_s^{(i)})$$

Objective:

$$minimize\ P(h)$$

Taking the above into consideration, our PSO optimization problem can be formulated as follows:

The hyperparameter to be minimized is the max depth of the tree.

$$h = \{maxDepth\}$$

The main goal of the optimization is to minimize the delay across all computational paths and to minimize the number of invalid predicted placements during the training phase. Let *i* represent the trial number and *j* represent the computational path; the average delay across all computational paths can be defined as:

$$avg_{delay,CP} = \frac{\sum_{i=1}^{n}\left[\frac{\sum_{j=1}^{k} delay_{CP_j}}{k}\right]}{n} \tag{4.1}$$

Where n is the total number of trials and k is the total number of computational paths. Additionally the regularization term follows the form:

$$\alpha * \log_{\beta}(\gamma + 1)$$

Where α represents the penalty factor, β represents the penalty magnitude and γ represents the quality. In this case, the α value was set to 1000 to match the scale of the delay, the β value was set to 2 to have a higher penalty magnitude, and the γ value was set as the number of invalid predicted placements (ip). Taking this into consideration, the following holds true:

$$regTerm = 1000 * \log_2(ip + 1) \tag{4.2}$$

By inspection, it is evident that as the number of invalid predictions approaches 0 so does this regularization term suggesting that in the ideal case where there are zero invalid predictions, the effect of this regularization term is 0 as illustrated bellow.

$$\lim_{ip \to 0} 1000 * \log_2(ip + 1) = 0$$

Fig. 4.1 illustrates the effect of the regularization term as a function of the number of invalid predictions.

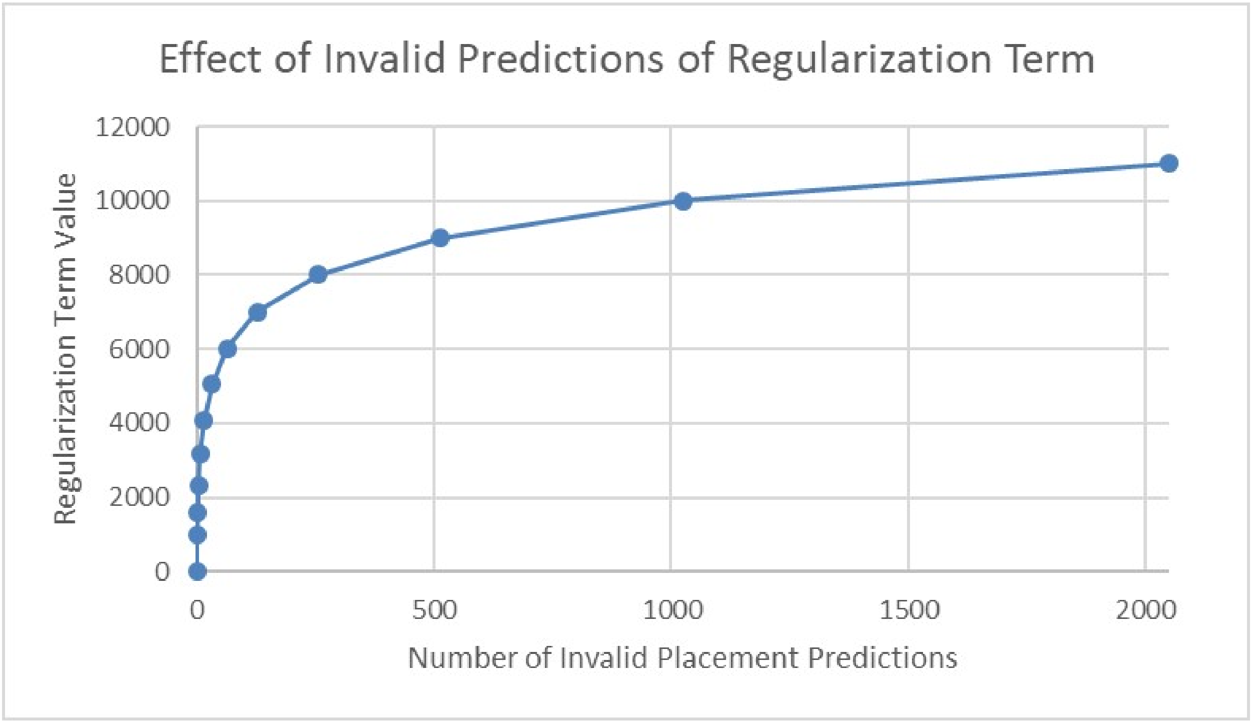

Figure 4.1: Effect of Invalid Predictions on Regularization Term

By combining the two expressions above we get the following as our functions:

$$f = \left\{avg_{delay,CP}, regTerm\right\}$$

Since the weights of each function are already expressed through the penalty factor of the regularization term, the weights are set to 1.

$$w = \{1, 1\}$$

Combining the above into a single objective function the following is obtained:

$$O_{PSO} = avg_{delay,CP} + regTerm \tag{4.3}$$

The above objective is expressed through the evaluation criterion which considers the objective, the training, and the validation set.

$$E(O_{PSO}, model(h), T_s, V_s)$$

After performing cross validation where b represents the number of folds the following is the resulting function requiring optimization.

$$P(h)_{PSO} = \frac{1}{b} \sum_{i=1}^{b} E(O_{PSO}, model(h), T_s^{(i)}, V_s^{(i)})$$

Finally, since this is a cost function the optimization problem objective should be as follows:

Objective:

$$minimize\ P(h)_{PSO} \tag{4.4}$$

When considering the above optimization problem the possible range of hyperparameter values is constrained to the functional range of the system. In this problem, the functional range is defined as the range where the number of invalid placement predictions falls below a specified error threshold and when it achieves steady state across 10 depth iterations. Effectively, the constraints on the hyperparameter values can be defined by:

Constraints:

$$a_1 \leq h \leq a_2$$

where the functional range is defined on the interval $[a_1, a_2]$.

4.3.2 Data Generation

In order to generate the training and testing datasets initial network topologies were constructed. These topologies were structured as 3-tier datacenters and the various network parameters were calculated according to the findings of Microsoft Corporation [11]. In order to generate the topology, an initial number of network servers and VNF instances were selected. For each server-instance permutation, 10,000 topologies were generated with differing network conditions and the respective VNF instances were placed using the BACON algorithm.

In this work, there were 4 different server-instance permutations considered and therefore, 40,000 topologies with varying conditions were generated. The first permutation contained 6 VNF instances to be placed on 15 network servers. Taking into consideration the distribution of VNF instance, there were a total of 4 different computational paths available for this topology. The second permutation considered the placement of 10 VNF instances to be placed on 30 network servers. 36 computational paths resulted from this permutation due to the distribution of instances. The final two permutations were used to assess the effect of topology on optimal tree depth.

In order to support the scalability of this design, the hypothesis that the max depth of the tree is irrespective of the number of instances and servers is proposed. To test this hypothesis we created the final two instance-server permutations such that in one the number of servers remains constant but the number of instances changes and in the second, the opposite holds true. The first of these server-instance permutations had 15 instances placed on 30 servers. Due to the distribution of instances, a total of 192 computational paths were present in this topology. The final permutation we evaluated had 10 instances placed on 45 network servers. As per the previous permutation with

10 instances, given their distribution, a total of 36 computational paths were present. Table 4.3 lists all of the server-instance permutation along with the distribution of instances and computational paths (CP).

Table 4.2: VNF Instances and Computational Paths per Topology

Topology	Total Servers	HSS	MME	SGW	PGW	CP
1	15	1	2	1	2	4
2	30	3	2	2	3	36
3	30	4	4	4	3	192
4	45	3	2	2	3	36

4.3.3 Data Analysis

Upon the creation of the various topologies through the data generation and initial placements using the BACON algorithm, the next stage in the methodology relates to the feature extraction. In order to predict the placement of VNF instances on network servers, a snapshot of the network conditions is taken and used as input features to the model. The output labels are the placements of the components on the network servers; as previously stated, this is a multi-class, multi-output problem, therefore there is a set of outputs predictions each with their respective set of possible labels. Given that s represents a network server and v represents an instance to be placed the following holds true:

$$outputs = \{v_1, v_2, ..., v_n\}$$

$$labels = \{s_1, s_2, ..., s_n\}$$

Table 4.3 lists the various input features which were extracted from a snapshot of the current network conditions along with their respective size where i is the number of instances and s is the number of servers:

Table 4.3: Feature Set Dimensionality

Feature	Dimension
Resource Requirements (instance)	(i x 2)
Resource Capacity (server)	(s x 2)
Interdependent instance delay tolerance	(3 x 2)
Delay between servers	(s x s)
Instance interdependence	(i x i)

4.3.4 Model Construction

The construction of the DO-DAT follows a 3 step process. The first stage involves the determination of the range of under/overfitting with respect to the tree depth. To evaluate this, PSO optimization is run given a range of [2,100] for the maximum depth hyperparameter. As previously mentioned, the evidence of under/overfitting in the DAT was evident through the number of invalid placement predictions; knowing this, the goal of this PSO optimization stage is to determine the depth at which the number of invalid placement predictions are minimized.

The result from the first stage are then used to determine the range of values to be further considered. The range of values is determined by considering when the number of invalid placement predictions falls below a pre-defined error threshold arbitrarily set at 7.5% and when steady state is reached meaning that there is no further improvement observed. The optimization performed in the second stage considers the entire objective function (6) evaluated across the previously determined range. The result of this optimization will show the effect of the range of depths on the joint consideration of invalid predictions and delay.

The final stage of the construction of the DO-DAT is to identify the optimal tree depth obtained from the previous stage and construct the model using this value as the initialized value of the max-depth hyperparameter.

4.4 Results and Analysis

The following is an a presentation of the results obtained as well as an analysis of their implication on the DO-DAT.

4.4.1 Implementation

The generation of the dataset was executed in Java while the data processing and machine learning models were implemented using Python. Both the generation of the dataset and the model implementation were run on a PC with an Intel ® Core™ i7-8700 CPU @ 3.20 GHz CPU, 32 GB RAM, and an NVIDIA GeForce GTX 1050 Ti GPU.

4.4.2 Functional Range

The first set of results pertains to the PSO optimization and its effectiveness in presenting the optimizal value of the tree depth. Fig. 4.2 displays the effect of varying the depth of the tree on the number of invalid placement predictions.

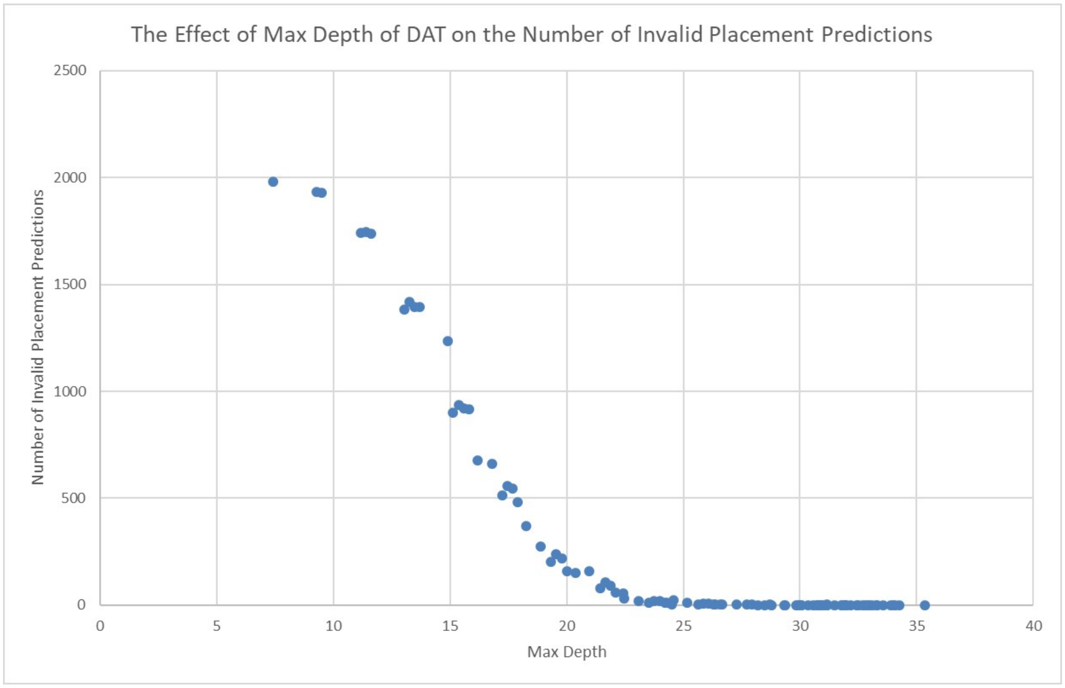

Figure 4.2: Effect of Max Depth on Invalid Predictions

As seen, the number of invalid placement predictions decreases while the tree depth is less than 25 and stabilizes at this minimum value of 0 while the tree depth is greater than 25. At a max depth of 20, the number of invalid placement predictions (error rate) is 7.5%. The range of tree depths spanning [20,35] is selected as the functional range of the first stage and will be further evaluated while taking into consideration the full objective function (6) as previously oulined in the methodlogy.

Fig. 4.3 presents the values of the max depth of the DAT evaluated during each iteration of the PSO spanning the range of interest previously identified. This figure accurately illustrates the search space coverage of the optimization process. Due to this coverage, it can be stated that the optimimal value obtained from the PSO optimization is a global optimum in the search space as the coverage of possible values is extensive.

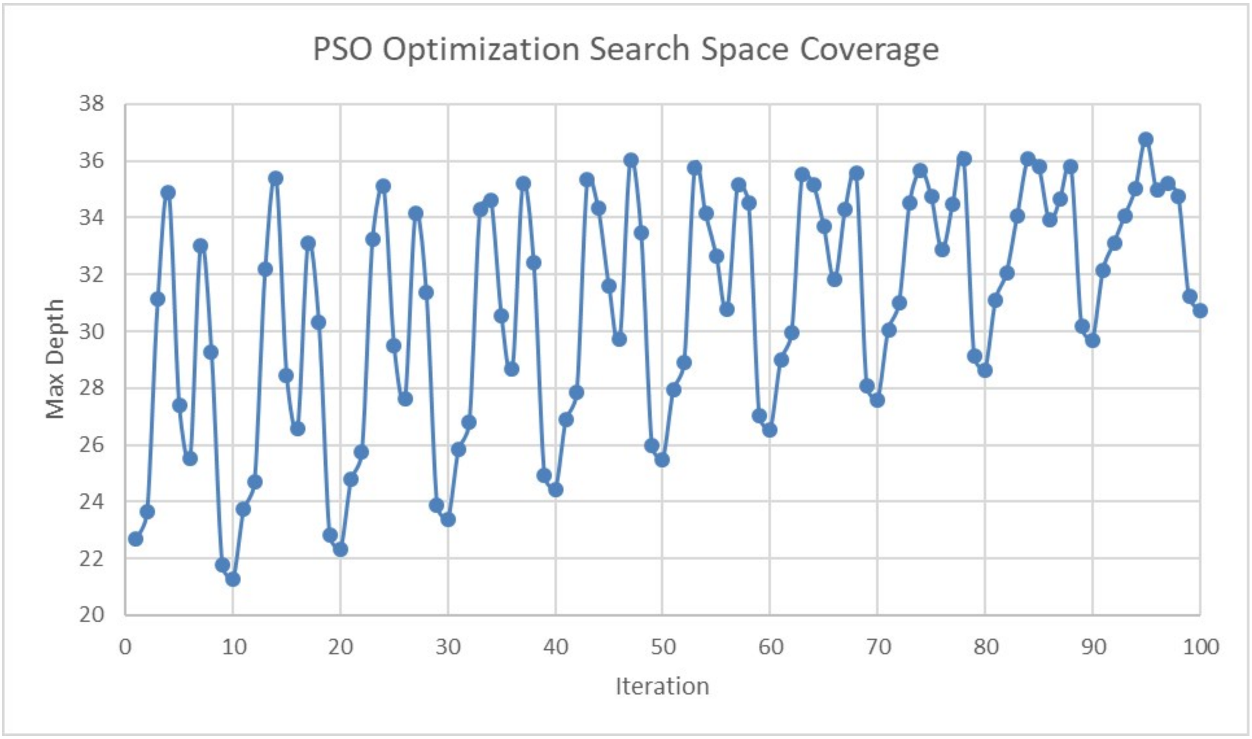

Figure 4.3: PSO Search Space Coverage

4.4.3 Optimal Depth

Results from the optimization of the functional range of interest are presented in Fig. 4.4. From this figure we can see the objective function *P(h)* is decreasing on the interval [20,28] and plateaus on the interval [29,35]. The interval [28,30] represents the interface between under and overfitting of the DAT and therefore, since there is no further significant improvement on the interval [29,35], the optimal depth is 29.

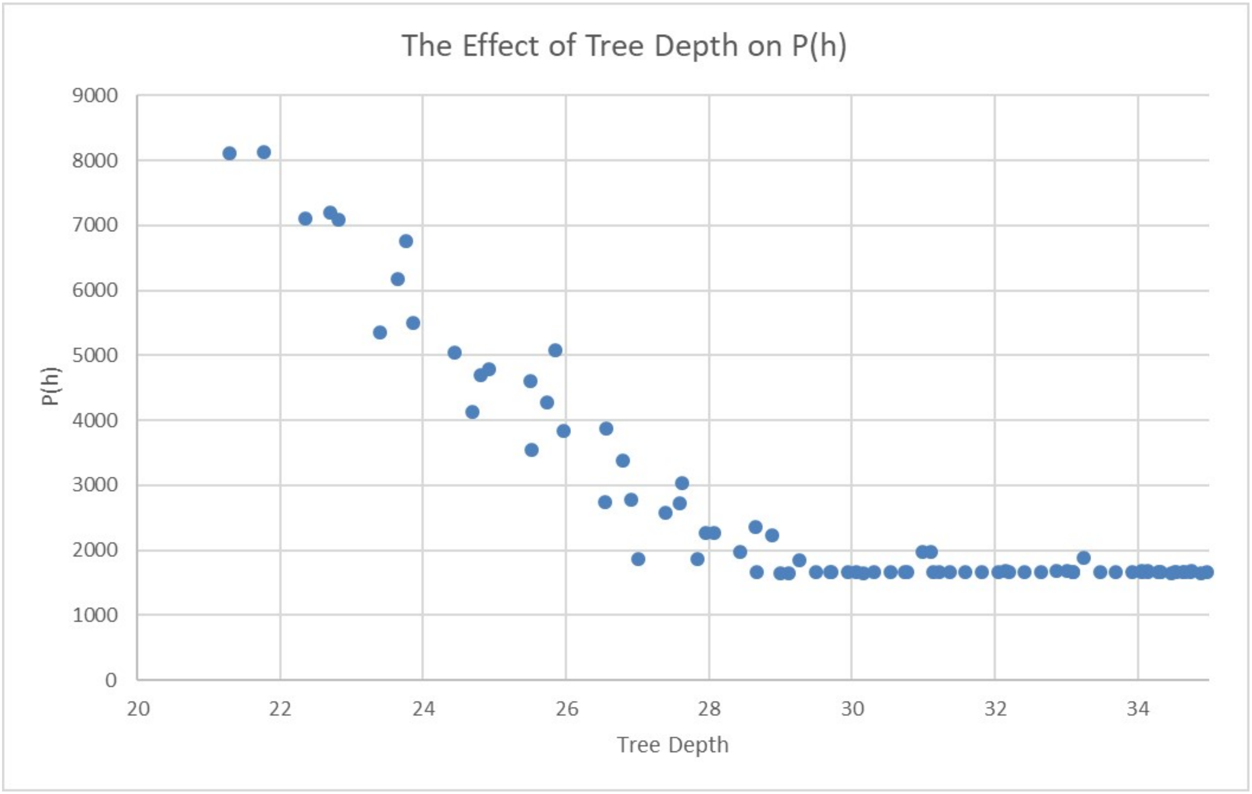

Figure 4.4: Effect of Depth on *P(h)*

4.4.4 Performance Comparisson

Taking this into consideration, the following figures will compare the placement of BACON, DAT, and DO-DAT. Fig. 4.5 illustrates the delay across the various computational paths in the small scale network.

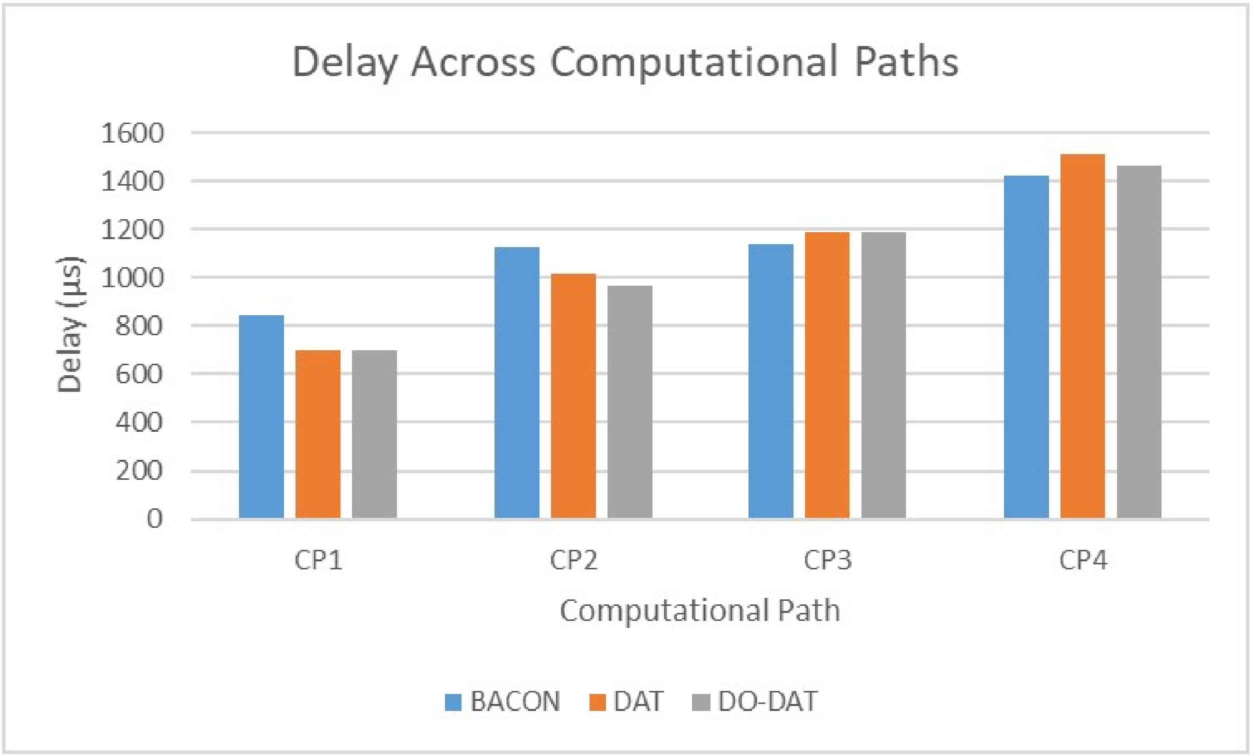

Figure 4.5: Delay Across Computational Paths of Small Network

As observed in this figure, DO-DAT exhibits improved performance when compared to its predecessor DAT; while the delay across CP1 and CP3 remains constant, the delay across CP2 and CP4 has slightly decreased. Furthermore, when comparing DO-DAT to BACON, it can be seen that CP1 and CP2 exhibit a lower delay by DO-DAT, whereas CP3 and CP4 exhibit a lower delay by BACON. However, it must be noted that the delay difference across CP1 and CP2 is significantly greater than that observed across CP3 and CP4 suggesting that overall, across all computational paths, the average delay observed through the DO-DAT placement is less than that observed through the DAT and BACON placements.

This result can be further extended to the second network topology as expressed in Fig. 4.6. From this figure it is evident that DO-DAT, when considered across all computations paths, produces more paths with less delay when compared to the other two placement methods. Furthermore, the success of the DO-DAT in minimizing the number of invalid placement predictions is observed. Visually, it is evident that the DAT produced invalid placement predictions on CP5, CP17, and CP29 due to the uncharacteristically low delays attributed to a violation of the availability constraints. When considering the performance of DO-DAT across these aforementioned paths, it is evident that the invalid predictions have been mitigated and the predicted placements produce a lower delay than those observed through the BACON placements.

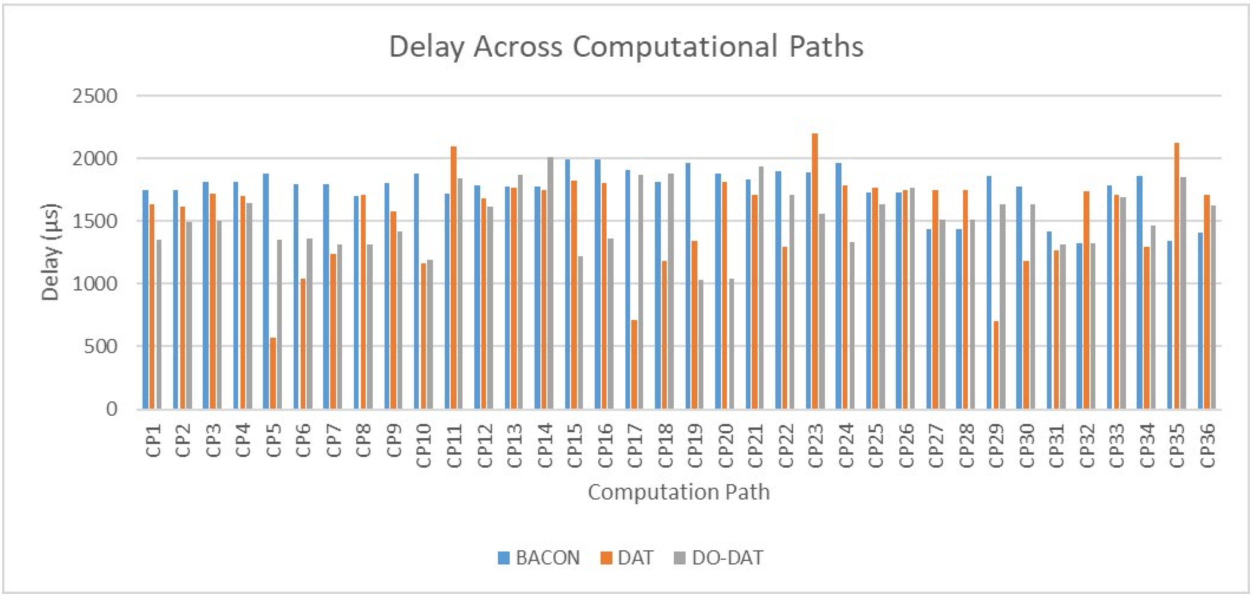

Figure 4.6: Delay Across Computational Paths Medium Network

A further clarification of this point is evident when considering Fig. 4.7. This figure shows a PDF of the difference between the DO-DAT and BACON algorithms in terms of placement delay. By calculating the difference in delay between the DO-DAT across every computational path placement, we can determine the probability of DO-DAT performing better than BACON.

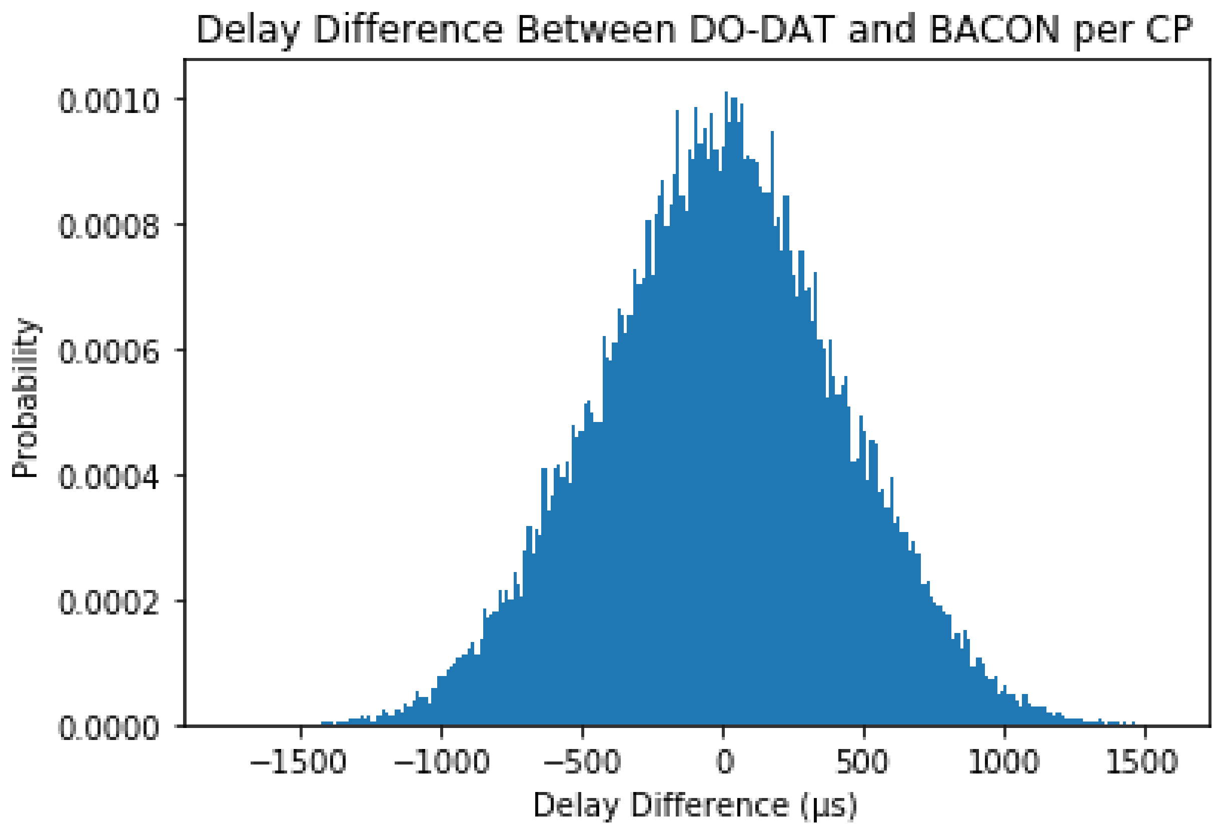

Figure 4.7: Delay Difference between DO-DAT and BACON

The mean in the above figure is -10µs meaning that on average, DO-DAT provides a computational path 10µs less of a delay when compared to BACON. This is an improvement on our previous work related to DAT which on average had 34µs more delay. The work presented in this chapter effectively improved the placement of VNF instances by 44µs on average. This is a very significant feat when considering the time-sensitive nature of NFV-enabled networks.

4.4.5 Scalability of Solution

Fig. 4.8 and Fig. 4.9 display the effect of server-instance permutations on the optimal depth of the DO-DAT. In order to prove scalability, the depth of the tree should remain constant across the various server-instance permutations. As seen through these figures, this is in fact the case as all permutations result in a maximum depth of 29.

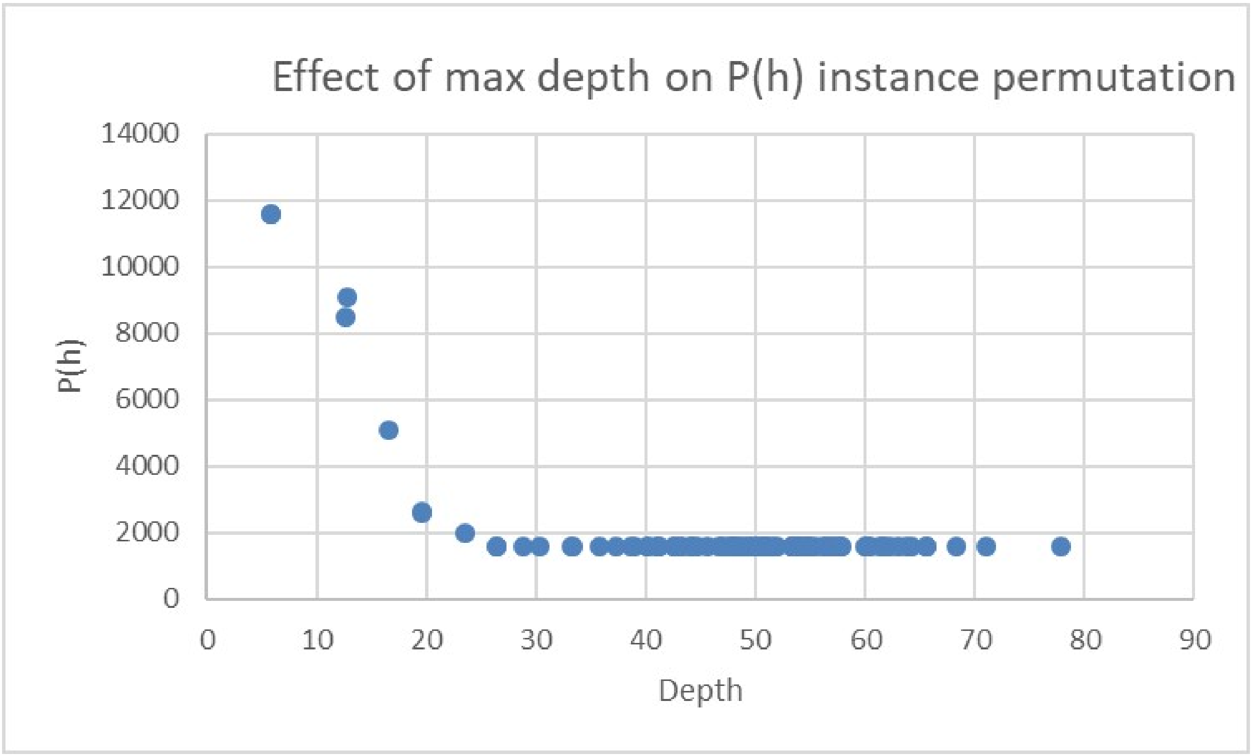

Figure 4.8: Depth Optimization Instance Permutation

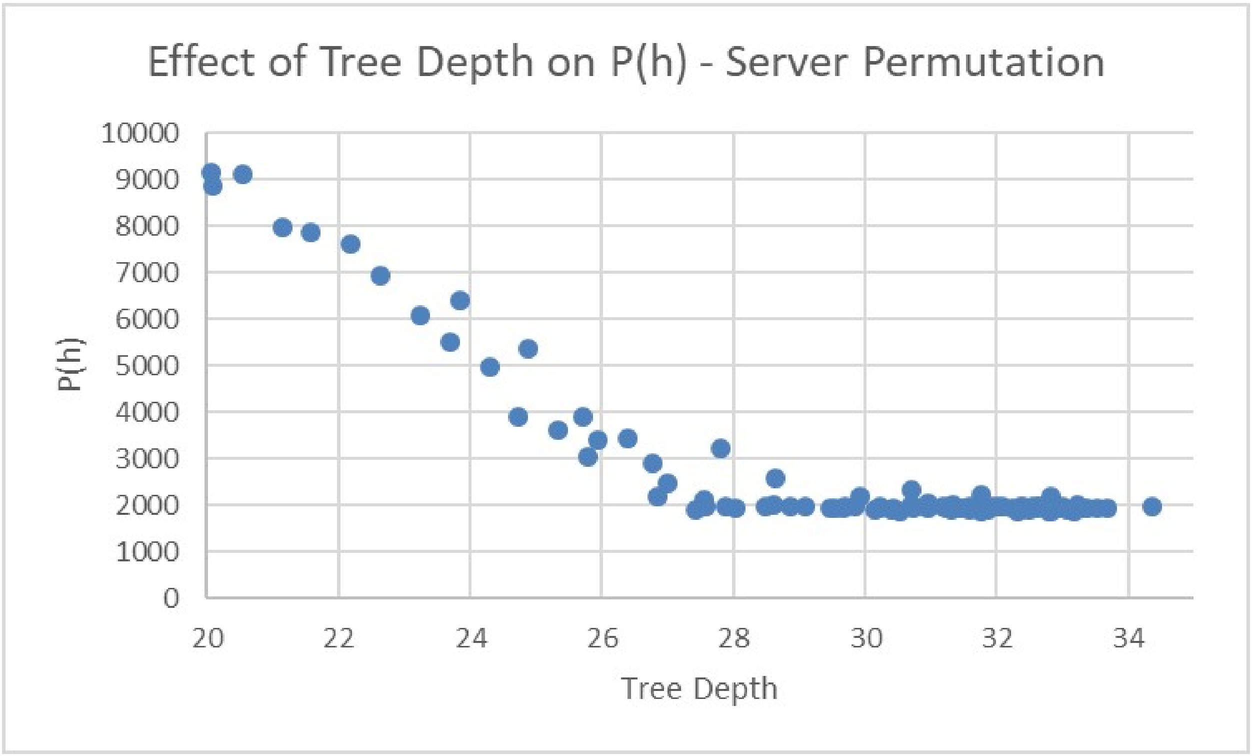

Figure 4.9: Depth Optimization Server Permutation

The confirmation that the DO-DAT is scalable across both the instance and server permutation topologies further reinforces the value of using a light-weight machine learning algorithm for the placement of VNF instances on network servers. However, in order to fully prove the added benefit of the DO-DAT a time complexity analysis must be completed.

4.4.6 Time Complexity Analysis

One of the benefits of the use of machine learning in networks is the reduction of system complexity. This is evident through the time complexity analysis of our proposed model. When considering the BACON algorithm, it has a computational complexity of $O(\frac{s^3-s^2}{2})$where S denotes the number of available servers in the network [6]. Our previous work outlined the complexity of constructing a decision tree denoted by complexity $O(n_{features} * n_{samples} * \log n_{samples})$ when creating the tree and $O(\log n_{samples})$when executing a query [12]. Additionally, the DO-DAT has an additional offline optimization component with the complexity defined by $O(n^2 t)$ where n denotes the population and t the iteration [13]. Since we have proven the scalability of the DO-DAT, the impact of the PSO stage is greatly minimized since the depth of the tree remains constant across various topologies.

4.5 Conclusion

The work presented in this chapter describes a key step towards an implementable, intelligent, and delay-aware VNF placement strategy for the NFV Orchestrator. This work has demonstrated not only an improvement upon the previously suggested DAT model but also a scalable solution capable of operating on various different network layouts and topologies. Through the optimization of the max tree depth, we have addressed the under/overfitting phenomenon which plagues large decision trees and negatively impacts performance.

Bibliography

[1] R. G. Mantovani et al., “An empirical study on hyperparameter tuning of decision trees,” arXiv Prepr. arXiv1812.02207, 2018.

[2] A. Sureka and K. V. Indukuri, “Using genetic algorithms for parameter optimization in building predictive data mining models,” in International Conference on Advanced Data Mining and Applications, 2008, pp. 260–271.

[3] G. Stiglic, S. Kocbek, I. Pernek, and P. Kokol, “Comprehensive decision tree models in bioinformatics,” PLoS One, vol. 7, no. 3, p. e33812, 2012.

[4] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms,” in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 847–855.

[5] D. M. Manias *et al.*, “Machine Learning for Performance-Aware Virtual Network Function Placement,” in GlobeCom, 2019. (accepted)

[6] H. Hawilo, M. Jammal, and A. Shami, “Network function virtualization-aware orchestrator for service function chaining placement in the cloud,” IEEE J. Sel. Areas Commun., vol. 37, no. 3, pp. 643–655, 2019.

[7] J. Kennedy, “Particle Swarm Optimization,” in Encyclopedia of Machine Learning and Data Mining, 2017.

[8] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” Evol. Comput. IEEE Trans., pp. 58–73, 2002.

[9] R. Hassan, B. Cohanim, O. De Weck, and G. Venter, “A comparison of particle swarm optimization and the genetic algorithm,” in 46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference, 2005, p. 1897.

[10] E. K. Burke and G. Kendall, Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, 2nd ed. Springer Publishing Company, Incorporated, 2013.

[11] A. M. Caulfield et al., “A cloud-scale acceleration architecture,” in The 49th Annual IEEE/ACM International Symposium on Microarchitecture, 2016, p. 7.

[12] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, no. Oct, pp. 2825–2830, 2011.

[13] D. S. Ruhela, “A study of computational complexity of algorithms for numerical methods,” 2014.

Chapter 5

Conclusion

As a way of dealing with rising costs and connectivity demands, network operators have turned to NFV as a viable solution. By abstracting network functions and executing them as software applications irrespective of the underlying hardware several benefits arise including network portability, flexibility, and scalability. Unfortunately, these benefits give rise to new challenges which must be addressed for the continual feasibility of this solution. The work presented in this thesis describes the first step towards an implementable, intelligent, and delay-aware VNF placement strategy for the NFV Orchestrator. This work addresses the NP-hard VNF palcement problem successfully through the implementation and training of a Delay-Aware Tree machine learning model, DAT, which is able to learn the near optimal placement of VNF instances forming a SFC. Further improvements are presented with the introduction of the Depth-Optimized Delay-Aware Tree (DO-DAT) which uses machine learning and an offline Particle Swarm Optimization (PSO) optimization to provide an effective, real-time placement solution. Results suggest that there is a significant reduction in time complexity achieved by using the DAT and DO-DAT as the majority of the computationally intensive training is completed online. While the DAT on average produced placements with an average additional delay of 34µs per computational path when compared to the current heuristic solution BACON, the DO-DAT exhibited an average reduced delay of 10µs per computational path. The overall improvement of the DAT throught the optimization of the depth hyperparameter has therfore resulted in an average delay reduction of 44 μs per computational path, a result which is significant given the delay-sensitive nature of the problem.

Future work in the field will address the following topics:

1. A full scalability analysis of the DO-DAT taking into consideration additional server-instance permutations

- This scalability analysis will consider the scalability of the DO-DAT by observing the time required to train, optimize, and predict the placements of VNF instances using the DO-DAT, as well as the memory and computational resources required during each of those stages.

- Based on the current work, and the time-complexity analysis performed, it is hypothesized that the results of this scalability analysis will further solidify the benefit of the DO-DAT as the time and resources required for the prediction of a placement using DO-DAT will be compared to that of BACON and the solution of the MILP model obtained through a commercial optimizer such as CPLEX

2. The optimization of additional hyperparameters and an insight into their impact on overall performance

- As previously mentioned, there were 3 additional key hyperparameters (min_samples_split, min_samples_leaf, and max_features) which contribute to the performance of CART decision trees. As a next step, the optimization of the DAT hyperparameters will be expanded to include these three hyperparameters

- There are two potential options for optimizing these hyperparameters, sequential or joint optimization

- Sequential optimization is the process by which each hyperparameter is optimized individually. This method reduces the search space however doesn't necessarily provide a global optimum when considering the configuration of all four aforementioned hyperparameters

- Joint optimization on the other hand greatly increases the search space as all four hyperparameters are considered simultaneously however, the solution obtained from this optimization will be the global optimum

3. The feature reduction of network data and the ranking of feature importance

- Considering the increasing complexity and network size, feature reduction techniques such as principal component analysis and auto encoders will be considered
- The objective of this work will be the isolation of the key features related to the prediction of VNF instance placement
- By knowing the key features required to make a prediction, the remaining features can be discarded thus reducing the time and resourced required during the training phase of the DO-DAT

4. The expansion of functionalities offered by the DO-DAT to address additional requirements of an NFV Orchestrator

- As previously mentioned, the work described in this thesis is the first step towards a fully automated intelligent orchestrator; as such, the functionalities offered by the DO-DAT must be expanded to address the other services provided by the NFVO including:
- VNF scaling
- VNF migration
- VNF re-instantiation
- Additionally, with the imminent introduction of 5G networks additional functionalities must be addressed such as:
- Network sensing
- Self-healing

Curriculum Vitae

Name:	Dimitrios Michael Manias
Post-Secondary Education and Degrees:	University of Western Ontario London, ON, Canada 2018 Bachelor of Engineering Science (Honours)
Honours and Awards:	NA Engineering Scholarship NA Engineering / AHEPA Foundation 2015
	AHEPA Foundation Undergraduate Scholarship AHEPA Foundation 2013
Honours and Awards (continued):	Dean's Honour List Western University 2013-2018
	Western Scholarship of Distinction University of Western Ontario 2013-2014
Work Experience:	Graduate Teaching and Research Assistant University of Western Ontario 2018-2019
	Senior Technical Student - System Planning Core Toronto Hydro Electric System Limited 2016-2017
Volunteer Experience:	Secretary IEEE London Section 2019

Patronage Chair
IEEE CCECE 2020
2019

Publications: D. M. Manias et al., “Machine Learning for Performance-Aware Virtual Network Function Placement,” in GlobeCom, 2019. (accepted)