Proactive Network and Traffic Aware Network Optimization

by

Abdolkhalegh BAYATI

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, AUGUST 18, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Mohamed Cheriet, thesis supervisor
Department of Automated Production Engineering, École de Technologie Superieure

Mr. Kim Khoa Nguyen, president of the board of examiners
Department of Electrical Engineering, École de Technologie Superieure

Mr. Michel Kadoch, member of the jury
Department of Electrical Engineering, École de Technologie Superieure

Mr. Jia Yuan Yu, external examiner
Institute of Information System Engineering, Concordia University

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON JUNE 10, 2021

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

FOREWORD

This dissertation is submitted for the degree of Doctor of Philosophy at the University of Quebec, École de Technologie Superieure (ETS). The research described herein was conducted under the supervision of Professor Mohamed Cheriet in the Department of Automated Production Engineering and Professor Kim-Khoa Nguyen in the Department of Electrical Engineering between May 2014 and February 2021. This work is original to the best of our knowledge, except where acknowledgements and references are made to previous work. Neither this nor any substantially similar dissertation has been or is being submitted for any other degree, diploma or other qualification at any other university.

Abdolkhalegh Bayati

March 2021

ACKNOWLEDGEMENTS

I would like to thank all the people who are continuously supporting me with their valuable encouragement. I would never have been able to continue my work without the guidance of my professors and committee members, help from friends, and support from my parents and my family.

I would like to express my deepest gratitude to my supervisor Professor Mohamed Cheriet, for his excellent guidance, endless caring, patience, and providing me with a fantastic atmosphere for doing research.

I am incredibly grateful to Professor Kim Khoa Nguyen, who helped with a friendly and practical approach, and patiently reviewed my work and drove my work path.

I would like to thank the "Synchromedia lab" team members, who were always willing to help and give their best support. My work would not have been possible without their help.

My special thanks to my wife and my son for their tremendous support and sacrifices and standing by me through this journey. I could never accomplish this without their support.

I would like to extend thanks to my parents and family for encouraging and motivating me through this time.

Abdolkhalegh Bayati

Optimisation proactive du réseau et du réseau sensible au trafic

Abdolkhalegh BAYATI

RÉSUMÉ

Le secteur des Technologies de l'Information et de la Communication (TIC) devient rapidement un contributeur important parmi les secteurs énergivores. Des études récentes ont montré que l'impact des TIC sur la consommation d'énergie mondiale est non négligeable - environ 4% (*GeSi SMARTer2030*, n.d.). Cela est dû à une gamme de nouvelles technologies, incluant le passage de la 4G à la 5G, l'avènement de l'Internet des Objets (IdO), la prolifération des appareils électroniques, et les processus de production rentables nécessaires pour la fabrication de tels dispositifs. Les réseaux de télécommunications absorbent 37% de la consommation énergétique des TIC. Dans de nombreux pays développés, les domaines liés aux TIC figurent en tête de liste des consommateurs d'électricité. En outre, par rapport à l'année 2020, une augmentation de la consommation énergétique mondiale liée aux TIC est prévue d'ici l'année 2030.

Traditionnellement, les réseaux de télécommunication ont été conçus pour maximiser la bande passante disponible. Cette stratégie vise à minimiser les coûts de remplacement survenant lorsque les technologies sont mises à jour pour augmenter soit le nombre des utilisateurs, soit le trafic échangé. Cependant, les utilisateurs connectent au réseau à différents moments de la journée et accèdent à des applications hétérogènes en termes de bande passante requise. Les réseaux sont dimensionnés pour faire face aux demandes les plus élevées de sorte que le volume de données circulant sur les réseaux soit garanti en dessous des débits maximaux réalisables. La consommation électrique des dispositifs réseau dépend de la capacité de la technologie installée.

L'idée de base pour la conception et le déploiement des réseaux conscients de l'énergie est de réduire la différence entre l'utilisation et la capacité du réseau. Afin de minimiser la perte d'énergie, ou uniformément, rendre la consommation du réseau égale à la charge de trafic, différentes solutions sont à l'étude. Les approches existantes comprennent deux catégories principales:

1. Approches proportionnelles à l'énergie : Cette catégorie vise à atteindre une proportionnalité énergétique en adaptant la vitesse (et la capacité) des dispositifs au volume du trafic dans les liens. En fait, lorsqu'un lien est sous-utilisé, sa vitesse est baissée car les interfaces réseau ayant des taux de transmission réduits consomment de petites quantités d'énergie.

2. Approches du mode veille : Cette catégorie affecte le réseau entier en répartissant le volume du trafic réseau proportionnellement de façon que certains dispositifs soient complètement utilisés, alors que certains d'autres dispositifs deviennent inactifs et passent automatiquement en mode veille. Dans ce cas, la consommation d'énergie des dispositifs est pratiquement indépendante du volume du trafic. Par conséquence, l'extinction des appareils peut économiser une quantité importante de l'‘énergie. Cependant, le mode de veille introduit

une complexité supplémentaire pour le réseau vu qu'il nécessite une coordination entre les périphériques.

Une méthode réputée parmi les approches proportionnelles à l'énergie est connue sous le nom du débit du lien adaptatif (Adaptive Link Rate –ALR). Le débit du lien adaptatif est un moyen efficace pour économiser la consommation d'énergie des éléments du réseau en ajustant le débit du lien en fonction du trafic transmis grâce à un processus d'optimisation d'allocation des flux réseau. Dans cette recherche, nous nous focalisons sur l'approche ALR pour résoudre le problème d'efficacité énergétique du réseau. Il faut noter que les approches d'adaptation existantes sont principalement réactives avec lesquelles le débit du lien est ajusté selon le changement de la demande du trafic. Au fur et à mesure que les flux du trafic fluctuent, la configuration optimale du réseau en globale change au fil du temps. De l'autre côté, le réseau nécessite d'être reconfiguré pour garder la consommation d'énergie minimale. Ainsi, au cours de chaque itération, plusieurs ré-optimisations nécessitent d'être exécutées, ce qui mène à la dégradation de la qualité de service (QoS) et à la stabilité du réseau, réduisant par conséquence la performance globale du réseau.

Dans cette thèse, nous visons à contribuer à une approche d'adaptation proactive. Nous proposons un système basé sur l'adaptation du débit en utilisant des techniques de prédiction (RAP- Rate Adaptation using Prediction) pour optimiser de manière prédictive les débits des liens en fonction de la prévision d'une demande du trafic à plusieurs étapes. RAP trouve une configuration du réseau qui minimise le coût énergétique au cours du créneau horaire actuel et qui nécessite que les modifications minimales soient ajustées avec les demandes du trafic des créneaux horaires futurs. Nous formulons RAP comme un modèle de programmation linéaire en nombres entiers (PLNE) et proposons une heuristique de recuit simulé (RS) pour le résoudre. Le modèle a été évalué sur deux topologies de réseau bien connues (GEANT et fat-tree) en utilisant des données de trafic réelles. Nos résultats expérimentaux démontrent que notre approche permet d'économiser considérablement l'énergie (61% en moyenne) avec des ré-optimisations liées au routage sensible à l'énergie.

Il y a quatre contributions principales dans cette thèse. Tout d'abord, nous avons conçu un noyau de régression de processus gaussien (GPR) appelé fonction de covariance semi-périodique auto-similaire (SPSS), qui est basée sur les caractéristiques du trafic et démontre une amélioration significative de la précision des prévisions du trafic par rapport à d'autres méthodes. Deuxièmement, nous avons proposé un modèle d'ensemble GPR afin de capturer différents motifs (formes) de trafic volatil. Le noyau proposé de notre modèle d'ensemble présente une nouvelle méthode permettant d'optimiser l'équilibre précision-diversité entre les apprenants de base, ce qui résulte en une amélioration des performances de la prédiction. La troisième contribution est accomplie en caractérisant notre algorithme prédictif à multi-étapes-en-avance avec les attributs du trafic. Nous avons démontré comment la nature multi-échelle du trafic pourrait être reflétée dans la prédiction multi-étapes-en-avance pour réduire la propagation d'erreur dans un horizon de prédiction étendu. Cette approche surpasse les algorithmes existants et puissants de séries chronologiques tels que LSTM. La quatrième contribution concerne le modèle d'optimisation énergétique du réseau. Tandis que les modèles existants d'optimisation de

l'efficacité énergétique ne prennent en compte que la consommation d'énergie, nous prenons en compte la prédiction du volume du trafic pour contrôler et minimiser le nombre de changements dans le réseau lors de l'exécution de notre approche d'optimisation en temps réel. Ces quatre contributions sont intégrées dans notre système énergétique du réseau et mettent en place une approche sensible au volume du trafic pour réduire la consommation d'énergie du réseau avec un trafic dynamique.

Mots-clés: vitesse de liaison adaptative, efficacité énergétique, apprentissage automatique, méthodes d'optimisation, modèles prédictifs, prévision du trafic, régression de processus Gaussien

Proactive Network and Traffic Aware Network Optimization

Abdolkhalegh BAYATI

ABSTRACT

The Information and Communication Technology (ICT) sector is rapidly becoming a big contributor among the energy-consuming sectors. Recent studies have shown that the ICT impact on global power consumption is non-negligible - about 4% (*GeSi SMARTer2030*, n.d.). This is due to a range of new technologies, including moving from 4G to 5G, the advent of the Internet of Things (IoT), the proliferation of electronic devices and the cost-effective production processes needed to manufacture such devices. Telecommunication networks consume 37% of ICT power consumption. In many developed countries, ICT-related sectors are on the top list of power consumers. Moreover, an increase in global ICT-related electricity consumption is projected by 2030 compared to 2020.

Traditionally, telecommunication networks have been designed to maximize the available bandwidth. This policy tends to minimize the replacement costs that occur when technologies are updated to increase either in the number of users or in the exchanged traffic. However, users access the network at different times of the day, and the applications usages are different in terms of traffic loads. Networks are dimensioned for the highest traffic demands so that the quantity of data flowing in the network usually is guaranteed below the maximum achievable data rates. The power consumption of network devices depends on the installed capacity of the underlying technology.

The key idea of the design and operation of Energy-Aware (EA) networks is to reduce the difference between the network's utilization and the provided capacity. In order to minimize energy waste, or equivalently, to make the consumption of the network equal to the traffic load, different solutions are being studied. The existing approaches include two main categories:

1. Energy-proportional approaches try to achieve energy proportionality by adapting the devices' speed (and capacity) to the actual load. In this case, link speed is decreased when the link is underutilized because the interface cards with lower transmission rates consume smaller amounts of power.

2. Sleep mode approaches affect the network as a whole and approximate load proportionality by carefully distributing the traffic in the network so that some devices are fully utilized, and other devices become idle and switched into sleep mode. In this case, the current devices' energy consumption is practically independent of the load. Therefore, switching off the devices can save a consistent amount of energy. However, sleep modes introduce additional complexity to the network because it needs coordination among devices.

A famous method in energy-proportional approaches is known as Adaptive Link Rate (ALR). Adaptive link rate (ALR) is an effective means to save energy consumption of network elements by adjusting the link rate according to the carried traffic through a network-level optimization

of the flow allocation process. In this research, we focus on ALR to provide a solution for the network energy efficiency problem. It is worth noting that current adaptation approaches are mainly reactive, in which link speed is adjusted when the traffic demand is changed. As the traffic flows fluctuate, the globally optimum network configuration changes over time, and the network requires reconfiguring to maintain the minimum power consumption. These approaches require multiple re-optimizations in each iteration which hurt Quality of Service (QoS) and network stability.

In this thesis, we propose a framework based on the *Rate Adaptation using Prediction (RAP)* to predictively optimize link rates based on the multiple-step-ahead forecasting traffic demand. RAP finds a network configuration that minimizes the energy cost in the current time slot as well as requires the minimum modifications to be adjusted to the demands in future time slots. We formulate RAP as an integer linear programming (ILP) model and propose a heuristic simulated annealing (SA) algorithm to solve it. The model has been evaluated over two well-known network topologies (GEANT and fat-tree) using real-life traffic data. Our experimental results show that our approach provides a significant energy-saving while it decreases (on average) %61 of re-optimizations in the energy-aware routing.

In this thesis, there are four notable contributions to this problem. First, we design a Gaussian Process Regression (GPR) kernel called *semi-periodic self-similar (SPSS)* covariance function, which is based on the traffic characteristics and shows a significant improvement in the traffic prediction accuracy compared to other methods. Second, we propose a GPR ensemble model to capture different volatile traffic patterns. The core of our proposed model is a novel method to optimize the *accuracy-diversity balance* between the base learners, which results in enhanced prediction performance. The third contribution is a multi-step-ahead prediction algorithm using the traffic characteristics. We showed how traffic's multi-scale nature could be reflected in the multi-step-ahead prediction to *reduce the error propagation* in an extended prediction horizon. This approach outperforms existing powerful time-series algorithms such as LSTM. The fourth contribution is a network energy optimization model. While exiting energy efficiency optimization models consider only power consumption, we used traffic prediction to control and *minimize the number of changes in the network* during our online optimization solution. These four contributions are integrated into our proposed network energy framework, and we build a traffic-aware solution to reduce network power consumption with dynamic traffic.

Keywords: adaptive link rate, energy efficiency, machine learning, optimization methods, predictive models, traffic prediction, Gaussian process regression

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

LIST OF ALGORITHMS

Page

LIST OF ABREVIATIONS

ALR	Adaptive Link Rate
ANN	Artificial Neural Network
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BTRA	Bit-Timescale Rate Adaptation
CP	Constraint Programming
DCN	Data-Center Network
DP	Dirichlet Process
DTRA	Demand-Timescale Rate Adaptation
EAR	Energy-Aware Routing
EEE	Energy Efficient Ethernet
EM	Expectation Maximization
ERT	Extremely Randomized Trees
FARIMA	Fractional Autoregressive Integrated Moving Average
FIR	Finite Impulse Response
GHG	Greenhouse Gas
GP	Gaussian Process
GPR	Gaussian Process Regression

GTB	Gradient Tree Boosting
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technologies
ILP	Integer Linear Programming
IP	Internet Protocol
KNN	K-Nearest Neighbors
LAN	Local Area Network
LASSO	Least Absolute Shrinkage And Selection Operator
LPI	Low Power Idle
LRD	Longe-Range Dependency
LSTM	Long Short-Term Memory
MCF	Multicommodity Flow
MCMC	Markov Chain Monte Carlo
ME	Mixture of Experts
MGP	Mixtures of Gaussian Process
MIP	Mixed Integer Programming
MLE	Maximum Likelihood Estimation
MLP	Multilayer Perceptron
MSE	Mean Squared Error

NFV	Network Functions Virtualization
NMSE	Normalized Mean Squared Error
OSPF	Open Shortest Path First
PTRA	Packet-Timescale Rate Adaptation
QoS	Quality of Service
RAP	Rate Adaptation using Prediction
RBF	Radial Basis Function
RF	Random Forest
RNN	Recurrent Neural Networks
RQ	Rational Quadratic
SA	Simulated Annealing
SDN	Software-Defined Networking
SE	Squared Exponential
SLA	Service Level Agreements
SPSS	Semi-Periodic Self-Similar
SRD	Short Range Dependency
SVM	Support Vector Machine
SVR	Support Vector Regression
TCP	Transmission Control Protocol
ToR	Top of Rack
UDP	User Datagram Protocol

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

x_i	feature vector of i-th sample
y_i	label (or target value) of i-th sample
$\mathcal{D}$	dataset of samples
d	length of feature vector (i.e., number of features in x)
$k(x_i, x_j, \theta)$	covariance function with hyperparameters θ
K	covariance matrix
α	innovation parameter in DP clustering algorithm
$f_m(x_i)$	prediction of learner m for sample x_i
$\hat{f}_{ens}(x_i)$	the final prediction of GPR ensemble model for x_i
$\mathcal{P}$	standard Gaussian likelihood in GPR
$\mathcal{L}$	the proposed GPR ensemble likelihood function
$\mathcal{V}$	the measure of accuracy-diversity balance in $\mathcal{L}$
$\mathcal{E}$	the measure of accuracy in $\mathcal{V}$
$v_m(x_i)$	effect of expert m on the accuracy-diversity balance for x_i
$\mathcal{I}$	the measure of diversity in $\mathcal{V}$
M	total number of experts in the GPR ensemble model
$\mathcal{L}$	the set of links in the network
L	number of links
$\mathcal{S}$	the set of possible states for a link

S	number of possible link states
a_l	state of link l
e_s	interface power consumption in state s
c_s	link capacity in state s
C_{max}	maximum capacity of a link
u_l	utilization of link l
$v_{l,s}$	a binary variable which is equal to 1 if link $l$operates in state s
$\mathcal{K}$	set of all the traffic flows in network
K	total number of traffic flows
d_k	bandwidth demand of flow k
P_k	set of precalculated paths for flow k
N	number of precalculated paths for each flow
r_k	the selected route for flow k
$w_{p,k}$	a binary variable which is equal to 1 if path $p$has been selected for flow k
G	network configuration including the state of links and selected paths for flows
$E(G)$	the power cost function
E_{max}	maximum power consumption of network
$R(G_i, G_j)$	the reconfiguration cost function for transition from G_i to G_j
I	available traffic information
T	prediction horizon (the number of future steps of traffic)

$\mathcal{T}$	range of time-slots that are considered in RAP
$Q(G)$	the objective function of RAP

INTRODUCTION

0.1 Context

The demand for computing power has been growing due to the penetration of information technologies in our daily routines at both personal and public levels, encompassing business, commerce, education, manufacturing, and communication services. The wide-scale appearance of online banking, social networking, online shopping, and others unfold workloads of great diversity and enormous scale at the personal level. At the same time, the computing and information processing requirements of many public institutions and private companies have been increasing quickly. Examples include digital services and functions required by various industries. Such a tense jump in the computing resources requires scalable and reliable information technology infrastructure involving servers, storage, network bandwidth, physical infrastructure, electrical grid, personnel and billions of dollars in capital expenditure and operational cost, to name a few.

Energy consumption of networks is a major issue in the information and communication technologies (ICT) area. The continually escalating energy consumption and the obligation to reduce the global greenhouse gas (GHG) emission to protect our environment have turned energy efficiency into one of the primary challenges of the communication community. In this context, the ICT community is expected to engage an active role in reducing worldwide energy requirements through the optimization of energy generation, transportation, and consumption.

In this thesis, we investigate the issue of network energy consumption and propose a solution for network energy optimization. The main elements in our research are introduced in the remainder of this section.

0.1.1 Software-Defined Networking for Network Management Automation

Computer networks typically interconnect hosts using network switches and routers which provide data packet forwarding and routing functionality. Switches transfer data among nodes on the same local area network (LAN) segment, whereas routers function as gateways, enabling packets between hosts in different networks. These forwarding devices usually run various operating systems and vendor-specific protocols that have to be configured through a process in which network operators translate high-level network policies into device-specific low-level commands and manually input these commands using command-line or graphical user interfaces. Scripts and tools significantly reduce the operational workload and errors in the configuration process. However, this approach still requires network administrators to reason carefully about the network and find the right balance between the use of automation, the situation requirements, and the changing state of the network. The lack of unified network control makes network management a challenging task, and the error-prone configuration process is the leading reason for network malfunctions, bugs, and security failures. Furthermore, due to this inflexibility, network innovation has mostly deteriorated, contributing to what some term the Internet ossification phenomenon.

With the advent of Software-Defined Networking (SDN), a paradigm transformation in communication networks has been set in motion. This shift is undertaken towards a logically centralized architecture which decouples control and data plane and thus allows moving control plane functions from network devices to dedicated controller instances. In SDN, the control plane functionalities of the network elements such as routing, signalling and label distribution are dispatched to one or more external entities known as controllers. However, the data plane functionalities such as forwarding, queuing and policing remain within the network elements. This decoupling enables a new network architecture where switches in the network can be

reduced to basic packet forwarding devices containing flow tables populated with localized flow rules.

OpenFlow currently provides the well-known SDN-enabling communications protocol between control and data plane that allows communication and supports the programmability of the control plane via open interfaces and APIs. In the OpenFlow architecture, a logically centralized controller manages switches by determining the rules that control their packet handling behaviour. These rules describe how incoming packets will be handled based on matching fields (such as the packet header content, incoming port, etc.). The rules are managed remotely by a controller entity, which communicates securely with switches, potentially using a standard protocol. To reconfigure a switch to enable a new policy, the controller modifies relevant entries in the flow tables. This modification may also be done reactively; for example, a specific packet may arrive at the switch. The controller updates existing flow rules or specifies new ones accordingly in real-time.

To cover aspects like scalability and resilience, concepts like HyperFlow enable the partitioning of OpenFlow networks into multiple domains that are each handled by individual controllers. A single controller within a network is beneficial as it provides centralized management. That is, routing decisions and policies are based on the global network view. However, this significantly increases the latency of switches that are far away from the controller. Hence, a single controller is a bottleneck in terms of processing power and is the single point of failure for the entire network. Therefore, the control plane is logically centralized but physically distributed across multiple controllers. All the controllers maintain a consistent global view to ensure proper network operation. The controllers must be deployed at locations that result in optimal performance. Each switch receives forwarding rules from a unique controller at any point in time. However, a switch can be assigned to more than one controller (one as primary and others as backup) to ensure reliability.

0.1.2 Network Energy Consumption

It is estimated that the power consumption related to ICT itself, ranging from 4% of the worldwide power consumption (*GeSi SMARTer2030*, n.d.), is expected to grow. Google, for example, consumed 260 Mega Watt of power in 2011 to run their data centers, which is sufficient to power 200000 homes and translates into an electricity bill of millions of dollars per month (Alanazi et al., 2017). The amount of electricity used for U.S. data center will reach 73 billion kWh in 2020, which will be over 1.8% of the total electricity consumption in the U.S (Lyu et al., 2019).

Energy consumption is a critical issue in ICT, which has two adverse effects. First, it leads to environmental problems and global greenhouse gas (GHG) emissions, and also, it increases the expense of communication. In this thesis, we investigate this issue and propose a solution by analyzing and modelling traffic behaviour.

Unfortunately, network energy consumption is independent of network traffic load and remains high even during underutilized periods. Internet Protocol (IP) networks are traditionally built with resource over-provisioning to guarantee the availability and reliability of services. They are designed with the redundancy of network devices and dimensioned to maintain critical applications at peak traffic load and recover from failures quickly. This design principle is opposed to the objectives of green networking and energy awareness. The core network is also over-provisioned to provide: reliable service during peak demand periods and survivability in case of link failures. Networks are designed and dimensioned to serve the estimated peak traffic demand. During network operation, traffic requests change notably over time, and even during peak hours, they are typically well below the network capacity. Moreover, protection techniques were utilized during the design phase to increase network resilience, increase capacity and reduce average link utilization.

A recent study on data-centers (D. Li et al., 2015) demonstrates that the switches or links with zero utilization are less than 5 percent. However, most switches and links are working with low utilization. About 80% switches have a utilization ratio below 55%, and about 20% of switches have a utilization ratio of less than 40%. The utilization of 80% of links is no more than 65 percent, and about 24% of links have less than 40% utilization. Only 1% (or less) of switches and less than 10% of links have a utilization higher than 90%. This under-utilization comes from bandwidth competition among flows. These flows fairly share the bandwidth of bottleneck links by TCP congestion control and only fully utilize the bandwidth of the bottleneck links. Briefly, the power consumption of current network architectures and transmission technologies is almost traffic-load-independent.

0.1.3 Traffic-Aware Network Optimization

In traffic-aware network optimization, the network configuration (i.e., routing, resource allocation, scheduling, etc.) is adjusted according to traffic demands to optimize the network performance to achieve a specific goal. This optimization can fulfill various network functions, including load balancing, reducing energy consumption, and network provisioning. A fundamental requirement for traffic-aware network optimization is collecting traffic data from the network to perform network monitoring.

Network monitoring is the application of a system that continually observes a computer network to collect data. Unlike an intrusion detection system which monitors a network for threats from the outside, a network monitoring system monitors the network for problems originated by overloaded or crashed servers, network connections or other devices. Network monitoring is required to examine a network for any failures or deficiencies to ensure continued network performance. For example, network monitoring will monitor the states of network elements such as routers, servers and firewalls. If a slowing or failing component is discovered, the network

monitoring software in use will notify network administrators of the issue, avoiding a network outage.

In traffic-aware network optimization, the problem is formulated as an optimization model with a particular objective function and a specified set of constraints. The information collected by network monitoring is used in the optimization process to achieve the goal. Figure 0.1 illustrates the concept of traffic-aware network optimization. In SDN networks, the traffic monitoring and the optimization process can be accomplished in the network controller. The optimal configuration selected by solving the optimization model can be applied to the network using the SDN controller. This provides flexibility in the design and implementation of the optimization process because the optimization process is decoupled from network elements and protocols.

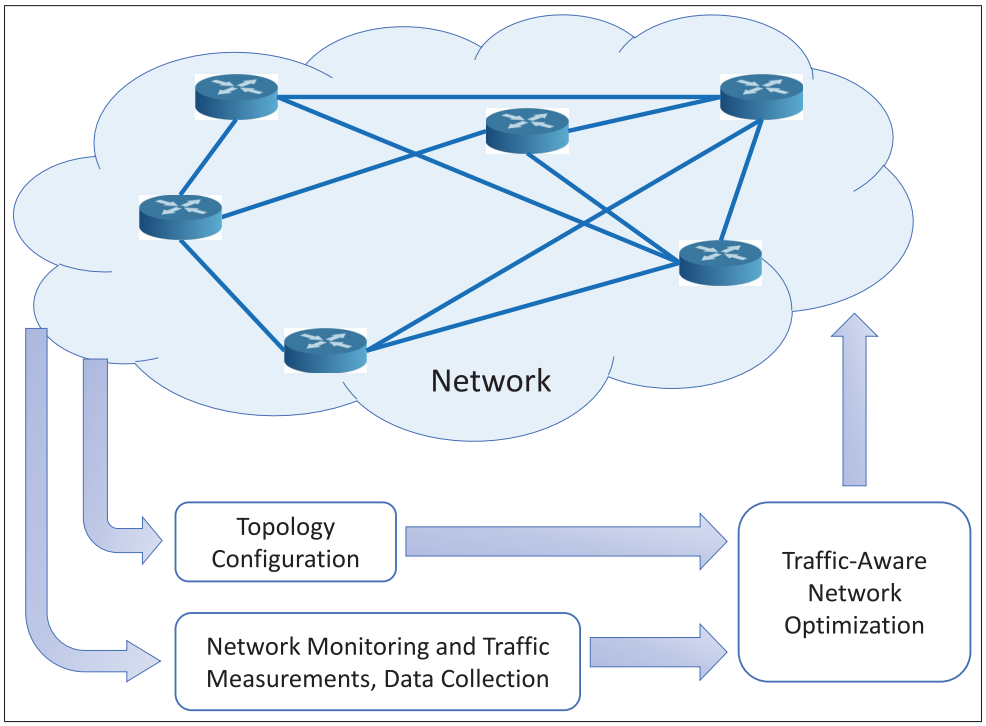


Figure 0.1 Traffic-aware network optimization

0.1.4 Traffic Characteristics and Behaviour

Network traffic characterization and modelling are substantial for network optimization and resource management. Modelling network traffic is a crucial first step towards understanding and embedding traffic behaviour in a traffic-aware solution that manages the limited resources in the network. An accurate traffic model must be able to describe and capture the salient traffic characteristics and behaviour. A generic approach to design traffic-aware optimization models is as follows. The first step is to investigate and understand the traffic characteristics in that type of network. After measuring and characterizing the traffic, a traffic model is needed which can capture the traffic characteristics. Different models are used to describe different types of traffic. Accurate traffic models are necessary for the optimization process to maintain quality of service (QoS) properly. The traffic model will be embedded into the traffic-aware optimization model.

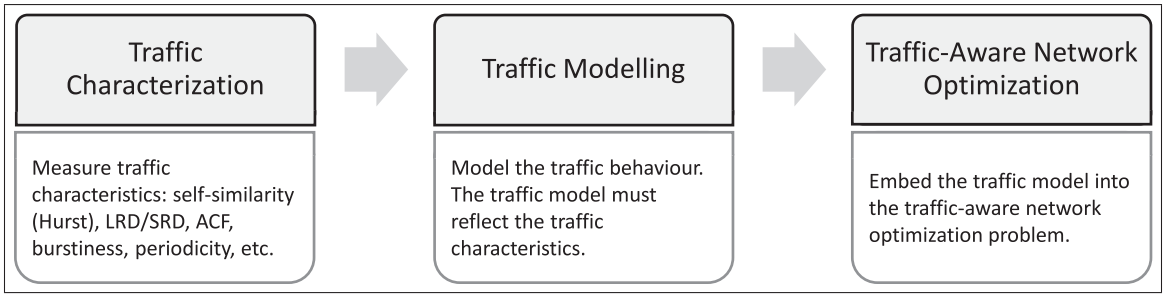


Figure 0.2 A generic approach for embedding traffic characteristics into traffic-aware network optimization

The study of traffic characteristics is an important branch of research in ICT. As one of the earliest findings, researchers realized the non-Poisson nature of Internet traffic (Leland & Wilson, 1991). Other studies followed that discovery and identified that Internet packet inter-arrival times are both self-similar and long-range dependence (Leland et al., 1995). Self-similarity (or fractal-like pattern) means the network traffic looks statistically similar at different time scales. The major reason for self-similarity in network traffic is heavy-tailed distribution corresponding

to large file transfer over the network (Karagiannis et al., 2004). It has been shown that the presence of self-similarity depends on traffic time-scale (Willinger et al., 2003). At long time scales, traffic exhibits long-range dependency (LRD) behaviour because it is affected by the users' and applications' behaviour. However, LRD is not observed at small time scales because network protocols and operations influence traffic behaviour. Instead, traffic shows short-range dependency (SRD) at short time scales. Several traffic characteristics mentioned in previous studies are as follows: Long-Range Dependence (LRD), and Short-Range Dependency (SRD), self-similarity, periodicity, non-stationarity, non-linearity, and burstiness.

Building an accurate traffic model is challenging due to the heterogeneous and complex nature of traffic behaviour. Network traffic behaviour is a function of many factors such as traffic characteristics, users' behaviour, transmission protocols, network topology, network type, etc. Many general time-series methods and machine learning algorithms have been employed in previous works to model network traffic. Examples of those models include neural networks (Morales et al., 2017), kernel-based methods (Haghighat et al., 2015), time-series models (Chen, 2017).

0.2 Problem Statement

This section explains the problem investigated in this thesis and gives a big picture by outlining various aspects of this problem and defining the relation between them.

The network design methodology brings many advantages to the users and guarantees network resiliency, redundancy, quality of service (QoS), and fast failure recovery. Despite its benefits, this design strategy is not consistent with green networking and energy awareness. The redundant network equipment consumes the same amount of energy constantly, regardless of its utilization. It means the power consumption remains the same: either the device is experiencing maximum workload or underutilized. In real-world cases, network utilization is usually far from peak

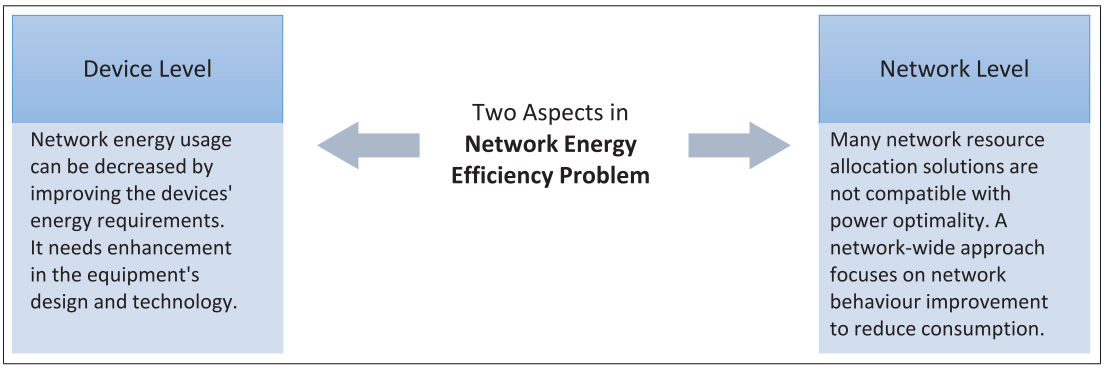


Figure 0.3 Two main aspects in network energy efficiency problem

utilization during long windows; however, the network energy consumption does not decrease even during the minimal operating load. We can have a better perspective when we note that the link usage (at the edge layer) in most data centers is, on average, less than 40% (Jiang et al., 2016). In other words, the gap between the actual and optimal power consumption is around 60% of the power usage. Energy awareness is the missing element in the traditional network design and configuration. Different layers of the traditional networks interact to provide various network functions, including redundancy, resiliency, load-balancing, QoS, provisioning, and resource allocation; however, they do not consider optimal energy consumption.

This problem includes two aspects. The first aspect corresponds to network devices (e.g., switches, link interfaces, routers) and analyzing their power usage specifications. It is about the equipment's technology and design features to measure their impact on power consumption. At this level, the focus is on the individual network devices to redesign their hardware or software in order to decrease their power consumption. The second aspect of this problem studies the network as a system and considers network-wide analysis. This aspect examines the network management and configurations, pinpoints the operations and behaviours in the network that are not compatible with optimal power consumption. It includes the analysis of resource allocation, networking protocols, traffic management and routing algorithms. The first aspect of this energy

efficiency problem (i.e., the equipment-level perspective) is not in our research scope. Instead, we explore the second aspect of this problem (i.e., network-wide viewpoint).

The network is energy-proportional if its consumption is proportional to its carried traffic. Existing network designs are not energy proportional. For achieving energy proportionality, the traffic load on the network must be considered in the resource allocation. In several cases, the traffic requirement is static and nonvarying. For example, when the traffic load is determined based on a service level agreement (SLA). In this case, the resource allocation problem can be solved once in order to optimize energy consumption. Traffic behaviour and fluctuations are not affecting the power proportionality problem in the case of static traffic. There is another situation when the traffic load on the network is not determined. For example, the traffic on the campus or the traffic on a data-center network may not be according to an SLA. Traffic fluctuation is a primary factor in energy proportionality for the case of time-varying traffic flows. The optimal level of energy usage changes with traffic load variation.

Unlike the static traffic example, in networks with time-varying flows, the instance of optimization problem evolves over time due to changes in traffic demands. In this case, the problem has to be solved repeatedly when the traffic varies. There might be some differences between the optimal solution and the actual network configuration at each optimization iteration. Since the optimal solution changes over time in response to the dynamically changing traffic demands, the network must be reconfigured at each iteration to have the minimum energy cost. The *network reconfiguration* includes the traffic flow reroutings and the link state transitions. However, this strategy is not effective because traffic demands are changing in consecutive time slots. The network needs to be re-optimized to minimize the energy consumption based on the demands in each time slot. Re-optimizing the network in consecutive intervals causes an excessive number of changes in the network configurations (i.e., the link-states and the traffic routes).

Unfortunately, these reconfigurations (i.e., link-state transitions and flow reroutings) degrade the network performance.

Another element of this problem is the traffic behaviour when we consider the networks with time-varying traffic. Traffic behaviour is affected by many factors, including users' behaviour, network protocols, topology, and management policies. Moreover, traffic behaviour is not the same in different types of networks. Therefore, it is extremely challenging to model such complex behaviour and integrate it into an optimization model.

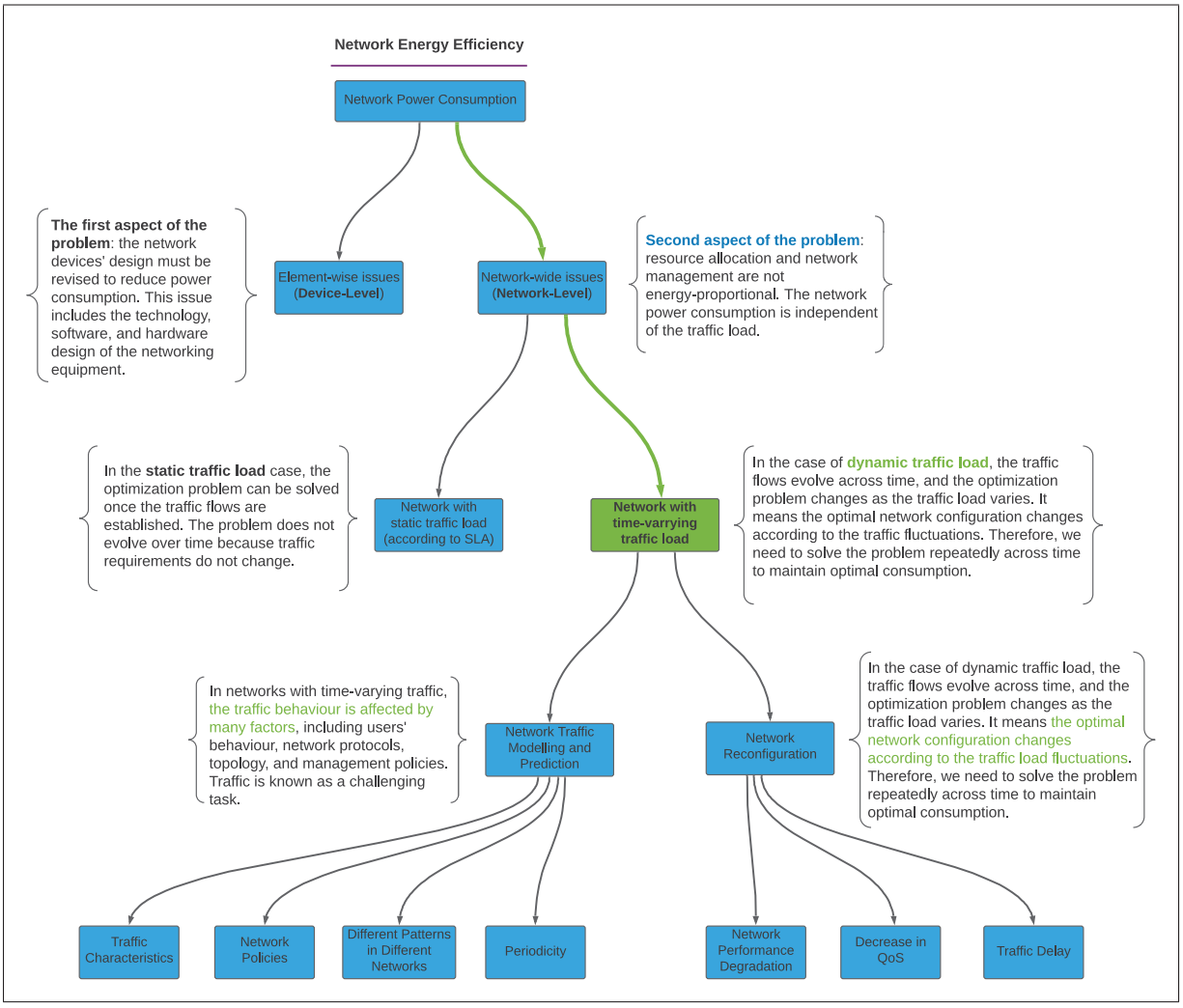


Figure 0.4 The problem of network energy efficiency and its related issues

In this thesis, we are focused on the problem of energy efficiency in networks with time-varying traffic. Figure 0.4 illustrates the scope of this thesis. We investigate energy efficiency from a network-wide viewpoint and concentrate on the network with time-varying traffic load. In the rest of this chapter, we provide details of this problem.

0.2.1 Energy-Aware Routing

Various solutions have been proposed to address this issue. Those solutions can be categorized into two classes: equipment-level solutions and network-wide solutions. In the first class (i.e., equipment-level approaches), the focus is on network devices to enhance their energy consumption independent of other elements in the network. Power proportionality in network devices is an example of equipment-level solutions (Vitturi & Tramarin, 2015). A network element is energy-proportional if its power consumption is proportional to its carried traffic (Christensen et al., 2010). In the second class of solutions (i.e., network-wide approaches), the global view of the network is considered in an optimization model to reduce overall network energy consumption. Energy-aware routing is an example of this class.

Researchers are actively studying new approaches to reduce energy consumption while assuring network performance guarantees. These approaches can dynamically adapt network resources and capacity to workload, thereby enabling energy conservation. In general, these strategies reduce energy consumption by putting network devices, or parts of components, in sleep mode whenever they are not in use. Many modern network elements are equipped with dynamic energy management and can operate in sleep (or standby) mode. However, this feature cannot be activated independently in a network because it must preserve network connectivity and ensure that service requirements are met. Thus, network devices must interact to determine what elements can be put into sleep mode without compromising service satisfaction.

Uncoordinated sleeping is a form of adaptive link rate (ALR) where links automatically are set into a low-power mode during idle periods between packets, for example, in Energy-Efficient Ethernet (EEE) (Tang et al., 2012; G.-R. Liu et al., 2018). Coordinated sleeping is a network-wide approach that chooses to put network elements into sleep mode based on global network knowledge. Such techniques can be traffic-oriented or topology-oriented, and they can be centralized or distributed (Manjate et al., 2018). Centralized approaches are often formulated as optimization problems for Mixed Integer Programming (MIP) or Constraint Programming (CP). These solutions are computationally demanding, NP-hard and require heuristic solutions, and it is not straightforward to embed them into traditional IP networks. Nevertheless, centralized approaches may be facilitated within SDN, where network functions are centrally implemented in a controller to manage the state of the network devices through the Openflow protocol.

Energy-Aware Routing (EAR) is a coordinated approach in which traffic flows are routed through the available paths in the network to reduce the total number of active devices in the network. EAR is a form of traffic-aware network optimization where the objective function is minimizing energy consumption. As a coordinated approach, EAR relied on the network-wide view and information to manage the traffic paths.

Traffic data and matrix is a vital input to the EAR optimization problem. The EAR model uses traffic information (collected through network monitoring) and network topology to select the optimal routing configuration for the flows. Its objective function is reducing energy cost, and its constraints depend on the network and its applications. In many existing EAR proposals, traffic demands have been considered constant (and fixed) in the long term (e.g., a fixed amount of demand during months). This assumption is valid for some networks, such as the backbone network, where traffic demands must satisfy the service-level agreement (SLA). However, in many cases, such as data-center networks, this assumption is inaccurate because traffic demands between servers vary significantly.

0.2.2 Effects of Traffic Fluctuations on QoS in EAR

In networks with static nonvarying traffic, the instance of the EAR problem does not change over time, and it needs to be solved just once the flows are established. Thus, the flow paths are stable under this assumption. In contrast, in networks with time-varying flows, the instance of optimization problems evolves across time due to changes in traffic demands. In this case, EAR problems have to be solved repeatedly and constantly reconsider the network configuration. At each optimization iteration, there might be some differences between the optimal solution and the actual network configuration. Since the optimum configuration changes over time in response to the dynamically changing traffic demands, the network must be reconfigured at each iteration to have the minimum energy cost. The *network reconfiguration* includes the traffic flow reroutings and the link state transitions.

However, this strategy is ineffective in networks with varying traffic demands (e.g., a data center with fluctuating traffic flows between servers). In such a situation, traffic demands are changing in consecutive time-slots, and the network needs to be re-optimized to minimize the energy consumption based on the demands in each time slot. Re-optimizing the network in consecutive intervals causes an excessive number of changes in the network configurations (i.e., the link states and the traffic routes). Unfortunately, these reconfigurations (i.e., link-state transitions and flow reroutings) degrade the network performance. A link rate transition leads to packet loss and delay because it requires a considerable amount of time (Gunaratne et al., 2008), and the link is not functional during the transition time (B. Zhang et al., 2008). Also, flow rerouting affects the order of packets, and they can be received out of order. The number of reconfigurations must be reduced to prevent network performance degradation in the ALR-based EAR.

The network reconfigurations introduce instability to the system and decrease the QoS (Destounis et al., 2018). A link rate transition leads to packet loss and delay because it requires a considerable amount of time (Gunaratne et al., 2008), and the link is not functional during the transition

time (B. Zhang et al., 2008). Also, flow rerouting affects the order of packets, and they can be received out of order. The number of reconfiguration needs to be minimized to avoid their negative impacts (Destounis et al., 2018). To the best of our knowledge, previous ALR-based EAR studies did not consider the effects of network reconfigurations in their path optimization.

This result is because of performing repeated EAR when traffic is fluctuating. Traffic fluctuations are not considered in many existing EAR solutions. In other words, they assumed the traffic flows require constant (fixed) bandwidth demand during their lifetime. Also, the number of flows inside the network is a constant value that does not change frequently. These assumptions are not accurate in many types of telecommunication networks such as DCN. Therefore, employing EAR in those types of networks will introduce instability and inefficiency. Instead, we need a solution that is agile in responding to traffic demand changes and link/switch failures. As the EAR problem instance evolves over time due to time-varying demands, the SDN controller must solve a sequence of routing problems and reconsider the flow configuration constantly. Also, it needs to solve these problems under tight timing constraints to satisfy application requirements.

The problem of EAR in networks with fluctuating traffic demands is still open and needs more research and investigations. In this work, we studied this problem and proposed a proactive solution based on traffic modelling and prediction.

0.3 Summary

In this thesis, we focus on the problem of energy consumption in data center networks with two features:

- they carry thousands of traffic flows;
- the traffic demands are fluctuating, and they are not constant in the long term.

An example of such a network is DCN, where thousands of servers are connected via thousands of links. It has been shown that DCN is responsible for 10% to 20% of consumption in the data centers. Unfortunately, current approaches are mainly reactive, in which link speed is adjusted when the traffic demand is changed. As the traffic flows fluctuate, the globally optimum network configuration changes over time, and the network requires reconfiguring to maintain the minimum power consumption. This approach requires multiple re-optimizations in each iteration, which hurt QoS and network stability, reducing overall network performance. This thesis investigates this problem to propose a proactive approach that considers traffic fluctuations in energy-aware network optimization.

CHAPTER 1

LITERATURE REVIEW

1.1 Network Energy Saving Approaches

Energy-saving approaches can be divided into three major categories: resource consolidation, power proportionality, and virtualization. Resource consolidation approaches (also known as coordinated approaches) are applied at the network level (Manjate et al., 2018). They achieve energy efficiency by putting network devices into sleep mode based on global network information. Resource consolidation approaches consider global network information (i.e., all the devices and traffic flows in the network) to optimize energy consumption. Their goal is to reduce consumption by maximizing the number of deactivated devices (or links) in the network. Such approaches can be traffic-oriented or topology-oriented, and they can be centralized or distributed. We review existing solutions regarding these two approaches in this section.

Power proportionality (in the second category) refers to the idea of consuming energy proportional to resource utilization (Andrews et al., 2012; Hernández & Donoso, 2017). These approaches focus on network devices to improve their energy efficiency by providing power proportionality in network elements. A network element is energy-proportional if its power consumption is proportional to its carried traffic. The network elements that are not energy-proportional constantly consume energy as they are fully utilized even when there is no traffic load on them. They are based on the fact that an idle or underutilized component or a device consumes a smaller amount of energy.

The third category (i.e., virtualization approaches) concentrates on the servers in the network increase the utilization of computational resources (X. Li et al., 2017). They decrease the number of physical servers by eliminating amounts of unused computing resources residing on underutilized servers and combining them onto virtualized environments (Islam et al., 2015). These approaches comprise of two types: software-level methods, e.g., (X. Li et al., 2017), and hybrid methods, e.g., (Y. Li et al., 2019). Figure 1.1 illustrates different energy efficiency

approaches. This thesis employed the first two approaches (i.e., resource consolidation and power proportionality) to propose our proactive energy-saving solution.

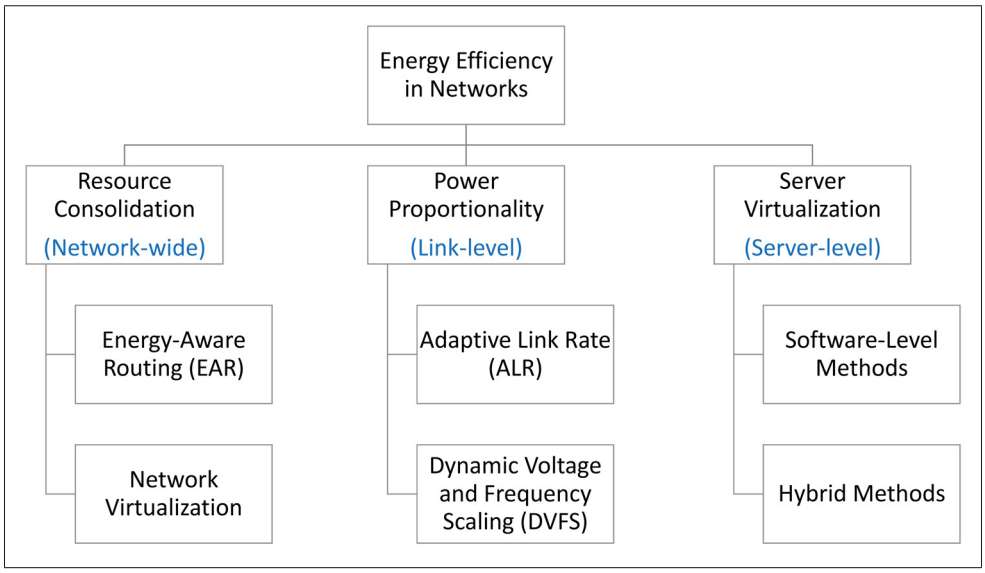


Figure 1.1 Existing energy saving approaches in different categories

1.1.1 Resource Consolidation

The *resource consolidation* approaches aim to aggregate and merge traffic flows on fewer links (and switches), while some of the non-utilized links (and switches) can be disabled. They employ traffic engineering approaches to reduce energy consumption by removing all (or some) redundancy in the network. These approaches target the over-provisioning of the resources, shut down underutilized devices (lightly loaded routers), and consolidate the network load and traffic on a selected cluster of active network equipment. An optimal solution shuts down the maximum number of cables while still routing all traffic demands on paths with sufficient bandwidth and QoS. The energy consumed is the minimum required to support the offered network load, but it comes at a cost to reliability as there are no redundant paths in the topology. The energy-saving has been achieved by focusing on bundled links (also called aggregate or composite links) in the backbone (core) networks (Fisher et al., 2010). The bundles are shut down and turn on according to the traffic demands in the network. In another work, it has been proposed to reduce

energy consumption by putting some or all components of a device into low-energy sleep states or clocking the hardware slower (Gupta & Singh, 2003). Resource consolidation has been investigated in different researches, including (Manjate et al., 2018).

Resource consolidation is the subject of Energy-Aware Routing (EAR) which formulates the network energy efficiency as a routing optimization to determine the paths used for each origin-destination pair or, equivalently, to ingress-egress routers in a transit network. The objective function is the minimization of energy consumption due to devices underutilized at a given time, and the control variables include the traffic paths. The key idea is to use as few network devices to accommodate the routing service as possible, with no or little sacrifice on the network performance. In another example, EAR has been investigated for high-density data-center networks (Huin et al., 2018).

A smooth EAR has been proposed to address the issues (related to energy) in legacy networks and migration to SDN-enabled networks (Huin et al., 2018). Also, a coordinated network management solution has been proposed, assuming that traffic varies according to a given set of time period scenarios and that routing is handled per-flow over a single path (i.e., unsplittable flow) (Addis et al., 2014). The problem of energy-aware routing in SDN-based carrier-grade Ethernet networks has been addressed (Maaloul et al., 2018). Their approach is based on turning off network nodes. It links to reduce energy consumption while respecting the rule space capacity for each Openflow switch and maintaining an allowable maximum link utilization. Identifying the optimal set of network elements to be turned off is NP-hard which has been solved using a heuristic algorithm suitable for large-sized networks. (Maaloul et al., 2018).

EAR has been in studied for different types of networks and technologies including NFV-enabled networks (Hong et al., 2018), IoT networks (Nguyen et al., 2018), mobile edge computing (Trinh et al., 2018), wireless sensor networks (Tunca et al., 2014), and delay tolerant networks (Naeem et al., 2020).

1.1.2 Power Proportionality

In link-level approaches, links automatically are put into a low-power mode during idle periods between packets to provide *power-proportionality*. There are two main methods to providing energy proportionality: deactivating (or powering off) the device during idle periods (Addis et al., 2014), and using adaptive link rate (ALR) (L. Wang et al., 2013; Andrews et al., 2012). In the first method, data is sent faster in the active interval to have a longer sleep interval. In another example, the authors proposed reducing energy consumption in a router or switch by putting some or all components of the device into low-energy sleep states or clocking the hardware slower (Gupta & Singh, 2003). The development of Energy Efficient Ethernet (EEE) has been considered by using a Low-Power Idle mode to reduce the energy consumption of a link when no packets are being sent (Christensen et al., 2010). They showed that packet coalescing could significantly improve energy efficiency while keeping absolute packet delays to tolerable bounds. However, this approach can cause packet loss in the downstream buffer, especially when using TCP/IP.

In the second method, link speed is decreased when the link is underutilized. It is based on the fact that the interface cards with lower transmission rates consume smaller amounts of power (Tang et al., 2012). ALR is an energy-saving technique that establishes a relationship between traffic workload and power consumption. The ALR methods aim to reduce the energy states of the network interfaces by reducing their capacities as a function of the network link loads. Either may reduce the energy consumption of links: (a) scaling down the data rate of the Ethernet communication link or (b) changing the state of the communication link to sleep/low power mode. The key issues of ALR are: (a) *ALR Mechanisms*: for a fast adjustment (and negotiation) of the link data rates, and (b) *ALR Policies*: for controlling the link date rate switching (B. Zhang et al., 2008). Many solutions proposed to answer these issues.

Considering the operation time-scale, ALR approaches consist of three broad classes (Francini et al., 2015): Demand-timescale rate adaptation (DTRA) with the period ranges from seconds to

minutes, Packet-timescale rate adaptation (PTRA) which works in time-scales of microseconds to milliseconds, and Bit-timescale rate adaptation (BTRA) that involves nanoseconds periods.

The BTRA and PTRA techniques apply to the individual network devices in the hardware-level and link-level solutions. They have been studied for real-time applications and different network protocols (Francini, 2012). Also, the power-proportionality gained by embedding these techniques in various interfaces, including small form-factor pluggable (SFP) modules and integrated twisted-pair Ethernet ports, has been measured (Francini et al., 2015). While the PTRA and BTRA techniques provide significant power saving (Francini et al., 2015), they cause performance degradation by triggering many state transitions (B. Zhang et al., 2008).

The DTRA methods have been involved in the *network-level* solutions, also known as energy-aware routing (EAR) algorithms applied to different networks such as SDN (Manjate et al., 2018; Rahnamay-Naeini et al., 2016), and data center networks (L. Wang et al., 2013). In EAR, the energy-saving problem is solved as a multicommodity flow (MCF) optimization model considering the speed-power curve of the networking devices (Andrews et al., 2012). The volumes of the traffic demands are used as inputs to the MCF optimization. Unfortunately, in the presence of variable traffic, the network re-optimization imposes many reconfigurations, which are the consequences of the temporal flow allocations and link speed transitions.

This thesis integrates ALR at demand-timescale (e.g., DTRA) with EAR to create a proactive energy-saving mechanism. Therefore, we focus on ALR and EAR solutions. Our solution addresses the issues related to ALR policies (i.e., controlling the link date rate switching) to reduce energy consumption and maintain QoS.

1.2 Traffic Modeling and Prediction

Network traffic prediction has been utilized to solve various problems. For example, it has been used in proactive resource management (Challita et al., 2018), mobile data offloading (D. Liu et al., 2018), data-center traffic management (Zhan et al., 2018), optimizing inter-data-center traffic flows (Y. Li et al., 2016), and forecasting big-data applications demands (Peng et al., 2016).

State-of-the-art traffic predictors are based on different machine learning algorithms including neural networks (Morales et al., 2017), Wavelet transform (Y. Li et al., 2016), kernel-based methods (Haghighat et al., 2015), time-series analysis (Chen, 2017), and LASSO (Choudhury et al., 2018).

Time-series analysis methods are common tools in this context. Examples of such time-series algorithms are autoregressive (AR), autoregressive moving average (ARMA), and autoregressive integrated moving average (ARIMA) models (Adas, 1997). These models are simple and require only a small number of previous samples, thus are proper choices for traffic prediction. ARIMA is a class of statistical models for analyzing and forecasting time-series data that has been used for Short-Range Dependent (SRD) traffic modelling and prediction (Adas, 1997). The major problem of these models is that they cannot capture long-range dependence (LRD) in real traffic traces (Karagiannis et al., 2004). The autocorrelation functions (ACF) of these models decay exponentially, which is pretty different from the ACF of real network traffic. Indeed, more complexity must be added to this kind of models to handle long-range dependent traffic.

As an effort to extend ARIMA in order to predict bursty and long-range dependent traffics, different models have been proposed, including FARIMA (N. Zhang et al., 2017), and a single-step traffic prediction model called ARIMA/GARCH (M. Zhang et al., 2017). FARIMA is the generalization of the ARIMA model in which non-integer values for the differencing parameter is allowed so that it can capture LRD traffic. FARIMA model has been employed for multi-step traffic prediction (N. Zhang et al., 2017). They reduced the complexity of fitting procedure in FARIMA and thus reduced its time complexity. The flexibility in describing both types of short-range dependent (SRD) and long-range dependent (LRD) traffic is the main advantage of their method compared to former ARMA model (Brockwell & Davis, 2013). However, to initialize parameters of their traffic forecasting algorithm, it is required to firstly determine Hurst exponent (Willinger et al., 2003) value of traffic using one of the known Hurst estimation methods (e.g. periodogram, the variance of residuals, R/S, etc (Karagiannis et al., 2004)). Therefore, their performance is highly dependent on the accurate value of Hurst exponent, which cannot be calculated definitively and only can be estimated (Karagiannis et al., 2004). Unfortunately, even

well-known Hurst estimators may still produce conflicting results, and there is no consistent robust Hurst estimator (Karagiannis et al., 2004).

Unlike time-series models which are based on assumptions on the nature of network traffic (e.g. stationarity or self-similarity), artificial neural networks (ANN) do not require prior knowledge about properties of data. ANN models are widely used for traffic prediction, which can handle real data characteristics such as non-linearity, non-stationarity, and non-Gaussianity. Among all types of ANNs, feedforward neural network (also called multilayer perceptron) was one of the first tools for developing traffic predictors. The simple implementation and well-understood learning algorithm (i.e. backpropagation) of Multilayer perceptron (MLP) network, as well as its approximation capability of unknown function, make it an attractive choice for prediction. However, MLP neural network is a very simplified version of the biological neuron, and it does not capture dynamic features of its biological counterpart, due to the utilization of static scalar weights in synapses of MLP networks. Consequently, MLP provides an only static mapping between input and outputs and it cannot handle the temporal behaviour of real data.

To address the disadvantages of MLP, two different approaches have been followed in recent works. The first approach focuses on the ANN architectures and their combinations. For example, adaptive finite impulse response (FIR) linear filters have been used instead of static weights in MLP neural networks to introduce time delays into the synaptic structure of the ANNs. FIR-based neural networks (FIRNN) use a modified version of the backpropagation algorithm called temporal backpropagation to determine weights of FIR filters. It has been reported that FIR neural networks gain better performance in time-series prediction compared to standard recurrent neural networks, linear predictors, Wiener filters, and MLPs. An ANN architecture has been proposed combining two individual ANNs to enhance prediction accuracy (Y. Wang et al., 2020). They observed their architecture outperforms autoregressive (AR) model in one-step network traffic prediction with regards to four types of traffic (i.e. MPEG, and JPEG video, Ethernet and Internet traffic traces).

The second approach in ANN traffic prediction targets improving learning algorithms instead of working on the architecture. A famous example is multiresolution learning, which significantly improves the neural network's generalization and robustness (Hardegen et al., 2020). Wavelet-Domain methods can reveal traffic features that have direct network interpretation (e.g. round-trip time) (Willinger et al., 2003). Multiresolution learning paradigm and FIR neural network have been exploited to develop a time-series prediction algorithm. Their results showed that the multiresolution learning paradigm improves neural network prediction capability. Performance of different ANN architectures and different learning algorithms for network traffic prediction have been studied (Hardegen et al., 2020). They applied the Real-Time Recurrent Learning (RTRL), and Extended Kalman Filter (EKF) based training algorithm on MLP and also Radial Basis Function (RBF) and recurrent networks and compared their performance.

A suggested algorithm based on the neural networks has been tested to target the traffic on network links (Morales et al., 2017). A combination of the wavelet analysis and neural networks has been used to form an algorithm for predicting the inter-data-center network traffic (Y. Li et al., 2016). Peng *et. al.* (Peng et al., 2016) developed an algorithm called HadoopWatch to predict the traffic demands of big data applications. LSTM is the result of recent advances in deep learning algorithms (Xingjian et al., 2015). It is a type of Recurrent Neural Networks (RNN) and is famous for its performance in time-series prediction. To the best of our knowledge, all existing traffic models and predictors are based on an individual learner. A disadvantage of ANN is a large number of training samples they require to train. This issue is not a major problem when the time interval is small. However, when the time scale is around second or more, this drawback leads to time-consuming training phase and reduces the applicability of ANN in real-time traffic prediction. Another disadvantage of ANN predictors is that they are analytically intractable.

1.2.1 Gaussian Process Regression

In a previous publication, we proposed a GPR-based traffic predictor by designing a covariance function to handle different traffic characteristics such as LRD/SRD, self-similarity, and

periodicity (Bayati et al., 2016). It achieved remarkable results compared to other time-series forecasting methods. However, the standard GPR has two main limitations. First, the computational requirements of GPR scale cubically with the number of training samples (P. Li & Chen, 2016), which is the consequence of the inversion of the covariance matrix. Second, GPR lacks the flexibility for modeling the nonstationary data (S. Sun & Xu, 2010). Both issues are fixed in this work. The problem of cubic time complexity is fixed because each GPR expert is trained over a limited subset of the samples. Thus, the inversion of a large covariance matrix in the standard GPR is substituted with the multiple inversion of small matrices. Also, using multiple GPR experts can easily handle the nonstationarity (S. Sun & Xu, 2010). These improvements are the result of using a group of GPR learners in an ensemble model instead of an individual GPR.

A model based on Mixtures of Gaussian Process (MGP) has been developed (Tresp, 2001). MGP has been inspired by the Mixture of Experts (ME) (Jacobs et al., 1991) which is a divide and conquer approach to address the shortcomings of standard GP. The MGP model is based on the introduction of a gating network to divide the input space into regions within which separate experts make the prediction. In another research, the experts and the gating network have been implemented using three sets of GPs (Tresp, 2001). Although that model affords the input-dependent scale parameter, its main drawback is that it requires the number of GP experts (in each set) to be determined a priori. The authors of (Rasmussen & Ghahramani, 2002) addressed this issue (i.e. the number of experts) by proposing a nonparametric model of the infinite mixture of GPs (IMGP) in which the gating network was a Dirichlet Process (DP). Their DP-based gating network enabled the model to infer the number of GP components from the data automatically. The Pitman-Yor process (PYP) prior has been employed instead of the DP process to build an IMGP (Chatzis & Demiris, 2012). PYPs, as a generalization of DP can comprise a high number of clusters with few data points, and a low number of clusters with a large number of data points.

While the mentioned MGP methods are based on the conditional models, there is another category of MGPs which uses a full generative model. Meeds and Osindero introduced an

alternative IMGP in which a joint mixture distribution over the input and output space has been extracted (Meeds & Osindero, 2006). Using such a joint distribution allows the model to accommodate partially observed data. The inference in this nonparametric model is made using Markov Chain Monte Carlo (MCMC) sampling. The alternative MGP proposed to utilize a finite number of GPs and employed a variational inference to approximate the posterior distribution using an iterative optimization technique (Yuan & Neubauer, 2009). In another work, an IMGP has been proposed which considered an infinite number of mixtures of GP experts (S. Sun & Xu, 2010). The main drawback in MGP methods is the time-complexity of the inference method, which is required for estimating the posterior distribution of the latent variables. In this thesis, we follow the ensemble learning paradigm to eliminate the necessity for inference methods. This improvement is the main difference between our models and the existing MGP models.

1.2.2 Ensemble Learning

Ensemble methods are learning algorithms (either in classification or regression problems) that generate a set of models that are combined by taking a (weighted) vote of their predictions (Polikar, 2012). Recently, the machine learning community is increasingly paying attention ensemble methods to address different learning problems such as confidence estimation (Tekin et al., 2016), feature selection (Seijo-Pardo et al., 2017), online learning (Y. Sun et al., 2016), data stream analysis (Krawczyk et al., 2017), and learning concept drift (Minku et al., 2009).

Recently, ensemble learning has drawn the attention of researchers thanks to its promising ability to resolve complex problems. Existing ensemble models employed different base learners including neural network (Yang et al., 2013), support vector machine (SVM) (Cheng et al., 2017), nearest neighbors classifier (Khan & Ahmad, 2018), and also hybrid of various classifiers (Lu et al., 2015). There are well-known ensemble models based on decision trees, including Gradient Tree Boosting (GTB), Random Forest (RF), and Extremely Randomized Trees (ERT). GTB is an ensemble of decision trees based on boosting approach. RF algorithm is an ensemble of decision trees and achieves diversity using a random split of the dataset. ERT is an extension of RF in which the randomness of the subsets has been improved. Ensemble models combine

a set of learners to solve different problems such as classification (Khan & Ahmad, 2018), regression (J. Liu et al., 2015), online learning (Y. Sun et al., 2016), and learning concept drift (Minku et al., 2009). A small portion of studies on ensemble learning focused on the time-series prediction. An ensemble of support vector regression (SVR) has been designed for prognostics of time-series data (J. Liu et al., 2015). A comparison of popular AdaBoost algorithms for time-series prediction has been conducted (Barrow & Crone, 2016).

Accuracy and diversity are two essential criteria to decide the performance of ensemble learning. Unfortunately, the majority of existing time-series ensemble models consider either accuracy or diversity only (Rahman et al., 2016). Few models consider both criteria. Recently, a layered ensemble architecture (LEA) has been proposed to address this problem (Rahman et al., 2016). In LEA, accuracy is obtained by finding an appropriate time lag, and diversity is mainly achieved using different training sets for learners. Unfortunately, LEA does not model a trade-off between accuracy and diversity as two conflicting objectives. Instead, it uses separate techniques to increase both factors independently. This approach cannot guarantee the optimal trade-off between accuracy and diversity of individual learners.

In a recent study, accuracy and diversity have been formulated to create the multiobjective deep belief networks ensemble (MODBNE) (C. Zhang et al., 2017). Each learner in MODBNE refines the error of other learners in the cost of reducing its accuracy. In MODBNE, the objective is a function of the output of all the learners for all the training samples. MODBNE is appropriate for small training sets because its complexity increases significantly by the rise in the number of samples and number of learners. The objective function in MODBNE measures accuracy and diversity, considering all the training samples and base learners. For large data sets, it leads to a complicated function with too many local optimums and prevents the ensemble from finding the optimal balance. In our time-series ensemble model, the accuracy-diversity trade-off is calculated locally for each learner, which allows avoiding the complexity issues. Each learner considers the samples in a small region of the feature space during accuracy-diversity optimization. Since the complexity of objective function (for training each learner) is not affected by the size of the training set, our approach is appropriate for large datasets.

A necessary condition for an ensemble model to be more accurate than any of its members is the degree of diversity within the experts. Combining different types of models is a common approach to obtain diversity. For example, the Gaussian process dynamical model has been combined with K-nearest neighbour (KNN) model to create an ensemble regression algorithm (Zhao & Sun, 2016). Another approach is to combine experts of the same type but with different configurations. For example, the method presented in a research employed several deep neural networks whose inputs have different window length (X.-L. Zhang & Wang, 2016).

CHAPTER 2

OBJECTIVES AND GENERAL METHODOLOGY

2.1 Objectives

Internet Protocol (IP) networks are traditionally built with resource over-provisioning to guarantee the availability and reliability of services. They are designed with the redundancy of network devices and dimensioned to maintain critical applications at peak traffic load and recover from failures quickly. This design principle is opposed to the objectives of green networking and energy awareness. Therefore, it is essential to revise network design strategies to reduce energy consumption while still ensuring that requirements on network performance are met. In 2011, it was estimated that information and communication technology (ICT) is responsible for 4% of global carbon emissions (Vereecken et al., 2011). Energy consumption during the active phase of the equipment represents about 40% to 60% of the carbon emissions. These emissions are expected to increase unless actions are taken to reduce this footprint (Manjate et al., 2018). A major part of these emissions, about 20%, is attributed to telecommunication networks.

In this thesis, the *principal goal* is to maintain the minimum network energy consumption presuming the fluctuated traffic load in successive time intervals. Since traffic is varying, the network configuration (i.e., paths of flows and link rates) must be adjusted to achieve energy conservation. The changes in the network configuration cause instability and decrease QoS and network performance. Therefore, our *second objective* is to preserve stability in network configuration by minimizing the number of changes in consecutive intervals.

To achieve the second objective, we need to integrate the traffic-awareness into the optimization model. To understand the role of traffic awareness in fulfilling the second objective, we need to look into the network optimization mechanism. Let us discuss this role by a simple example. Assume the network is in an optimal state regarding its power consumption (i.e., minimum consumption). The traffic demand on a route disappears in the next time slot, leaving a few unutilized links (i.e., no traffic load on those links). The network is no longer in the optimal

state because unutilized links consume energy but do not carry traffic. We may turn off the unutilized links to reduce consumption. This action is a reconfiguration to the network (i.e., in this example, reconfiguration means to turn off the links) occurred in response to the traffic load fluctuations. In general, **the frequency of reconfigurations and the number of changes (in each reconfiguration) depend on traffic fluctuations**. This conclusion introduces the role of traffic awareness in network power optimization.

We can imagine two approaches to incorporating traffic awareness into network optimization: the reactive and proactive approaches. First, we can use the observed traffic values to feed the optimization model and tune the network configuration according to the traffic's current state. This approach leads to a reactive solution. It means we change the network configuration when we notice a change in the traffic loads. This "adjust as traffic-change" solution may not guarantee the optimality of energy consumption and reconfigurations. When we follow the traffic fluctuations to adjust the network, we may end up in a situation where we pay a massive cost (in terms of energy or network changes) to reach the optimal state. Thus, it is better to utilize a traffic model instead of the raw observed traffic values. A traffic model allows us to understand the traffic behaviour and patterns, which is not obtainable using raw observed values. We can perform traffic prediction by applying a traffic model and use the predicted values in our optimization model instead of raw observations. This approach leads to a proactive solution (instead of a reactive system).

This thesis aims to provide a proactive model, so our third objective is to build a traffic model that allows us to perform accurate traffic predictions. The forecast horizon is a crucial element in the prediction algorithms. A long horizon shows traffic variations over the long-term future and provides ample time to make the right choices. There are several circumstances in network management where immediate network improvements are costly or not feasible. For example, the time taken to set the wavelength (or lambda) in optical networks is often within minutes, so it cannot be done instantly. Multi-step traffic forecasting provides sufficient time for constructive management in such cases. This factor leads us to the next objective, which is designing a multiple-step-ahead traffic prediction algorithm.

To sum up, the main objective is optimizing network energy consumption. The specific objectives are as follows:

1. Maintain minimum network energy consumption in consecutive time-slots;
2. Decreasing number of changes in network configuration during the repeated optimizations;
3. Network traffic modelling and data analysis;
4. Designing a multiple-step-ahead prediction algorithm.

The first objective (i.e., minimum energy consumption) is the main objective of the thesis. The following three items are the sub-objectives that we need to achieve to accomplish the primary goal. The following figure shows our thesis objectives and the connection between them.

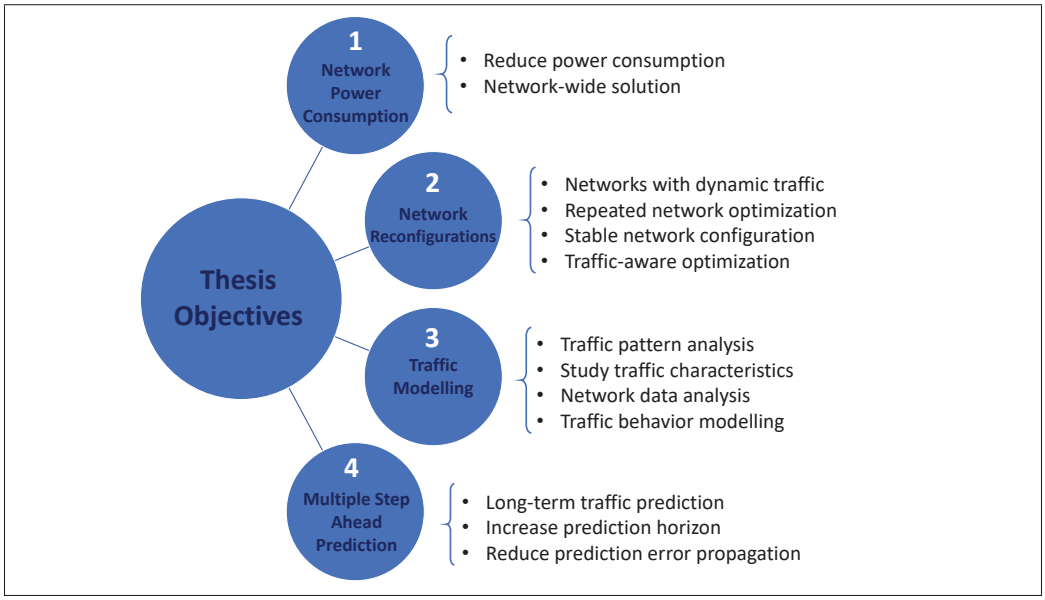


Figure 2.1 The thesis objectives and sub-objectives

2.1.1 Reducing Energy Consumption

This is the main objective of this thesis. We model the network energy consumption and consider the role of traffic and network configuration in this problem. We approach this problem from a

network-wise perspective instead of working on individual network elements. In this approach, we change the traffic route according to the network resources to optimize consumption. There is a combination of two main methods that we utilize in this research: Energy-Aware Routing (EAR) and Adaptive Link Rate (ALR).

2.1.2 Minimizing Network Reconfigurations

The second objective is focused on the network reconfiguration during repeated optimizations. The network configuration needs to be adjusted when the traffic demand is changed. As the traffic flows fluctuate, the globally optimum network configuration changes over time, and the network requires reconfiguring to maintain the minimum power consumption. In other words, the network needs to be re-optimized to minimize the energy consumption based on the demands in each time slot. Re-optimizing the network in consecutive intervals causes an excessive number of changes in the network configurations (e.g., the routes). Unfortunately, these reconfigurations degrade the network performance. A link rate transition leads to packet loss and delay because it requires a considerable amount of time (Gunaratne et al., 2008), and the link is not functional during the transition time (B. Zhang et al., 2008). Also, traffic flow rerouting affects packets' order, and they can be received out of order. The number of reconfigurations must be reduced to prevent network performance degradation.

2.1.3 Traffic Modelling and Characterization

We aim to design a traffic-aware energy optimization model. The study and understanding of traffic behaviour and characteristics are required assets in this method. The goal of the traffic characterization (in our research) is to build a customized forecast algorithm adjusted to the network traffic behaviour. Thus, we keep an eye on prediction algorithms during the study of traffic characterizations. In other words, we need to investigate traffic characteristics in order to integrate them into our prediction algorithm (to improve the forecast precision). Traffic time-series have a highly evolving behaviour since it includes different patterns at different times, links, and various time scales. Meanwhile, dominant characteristics of traffic process

such as self-similarity and long-range (short-range) dependency are definitely associated with its behaviour (Willinger et al., 2003). Therefore, we need to study and understand traffic characteristics for building an accurate traffic model. First, we investigate traffic characteristics, including Long/Short range dependency (LRD and SRD), periodicity, and self-similarity. Then we propose a traffic model based on a machine learning algorithm that can handle traffic characteristics. We propose a network traffic predictor based on Gaussian Process Regression (GPR) which considers traffic characteristics to improve the prediction accuracy. Our traffic prediction algorithm is utilized in the energy-aware optimization model to achieve the objectives.

2.1.4 Traffic prediction

Traffic prediction is an important objective in this thesis. As mentioned earlier, our approach is to select a network configuration in each time slot that reduces the network energy consumption in the current time slot and needs the minimum number of changes to adapt to the traffic demands in the future time slots. A prediction algorithm must forecast the traffic demands in future time-slots and eliminate the temporary reconfigurations in this approach. An appropriate traffic prediction method must respond to several requirements in order to be suitable for working in a real network environment. Firstly, it must be highly accurate to reduce the risk of consequence decisions. Clearly, an inaccurate prediction algorithm deteriorates the overall performance of the network and reduces resource utilization. The second is time constraint. In computer networks, decisions must be made almost in real-time. Another requirement is the computational resources required for the prediction algorithm. Traffic predictors are usually implemented in network devices such as routers and switches, which are limited in available computational resources (e.g. CPU, memory). All these restrictions make traffic prediction a challenging subject for researchers. We design a multiple-step-ahead prediction algorithm based on the Gaussian Process Regression (GPR) algorithm. We design a multiple-step-ahead prediction algorithm based on the Gaussian Process Regression (GPR) algorithm. Our GPR-based algorithm is customized for the network traffic characteristics that we investigated as part of the previous objective.

2.2 General Methodology

As mentioned above, we have four objectives in this thesis: reducing energy consumption, avoiding network reconfigurations during repeated network optimization, traffic modelling, and reducing error propagation in the multiple-step-ahead traffic forecast. In this section, our methodology for achieving these objectives is described. Our methodology is depicted in figure 2.2, summarizes six steps to build our framework. In the rest of this section, we explain our methodology in bottom-up order.

2.2.1 Traffic Characterization and Modelling

We start our methodology by exploring and study of traffic characterization. This step will lead to the design of a forecast algorithm that fits the traffic behaviour. By focusing on traffic features and characteristics, we can understand its behaviour in different networks and applications. We examine traffic patterns at different time scales under different circumstances. This study will be done on the real traffic data. We use the result of this study to build a kernel (i.e., covariance function) for the Gaussian Process Regression (GPR) algorithm. The covariance function must be designed and customized according to the traffic characteristics. The outcome of this step is an initial prediction algorithm adjusted to the network traffic characteristics.

2.2.2 Enhanced Traffic Model

This step focuses on using machine learning approaches to enhance the initial traffic prediction model achieved in the previous step. The initial GPR-based algorithm is the result of analyzing and study of traffic behaviour. We aim to improve its prediction performance using machine learning techniques such as the ensemble model. Ensemble modelling is a machine learning approach in which a set of learners are trained for a learning problem so that each learner rectifies the prediction error of other learners. Although individual learners' prediction performance is low, the overall (or final) prediction is precise because other learners correct each learner's error.

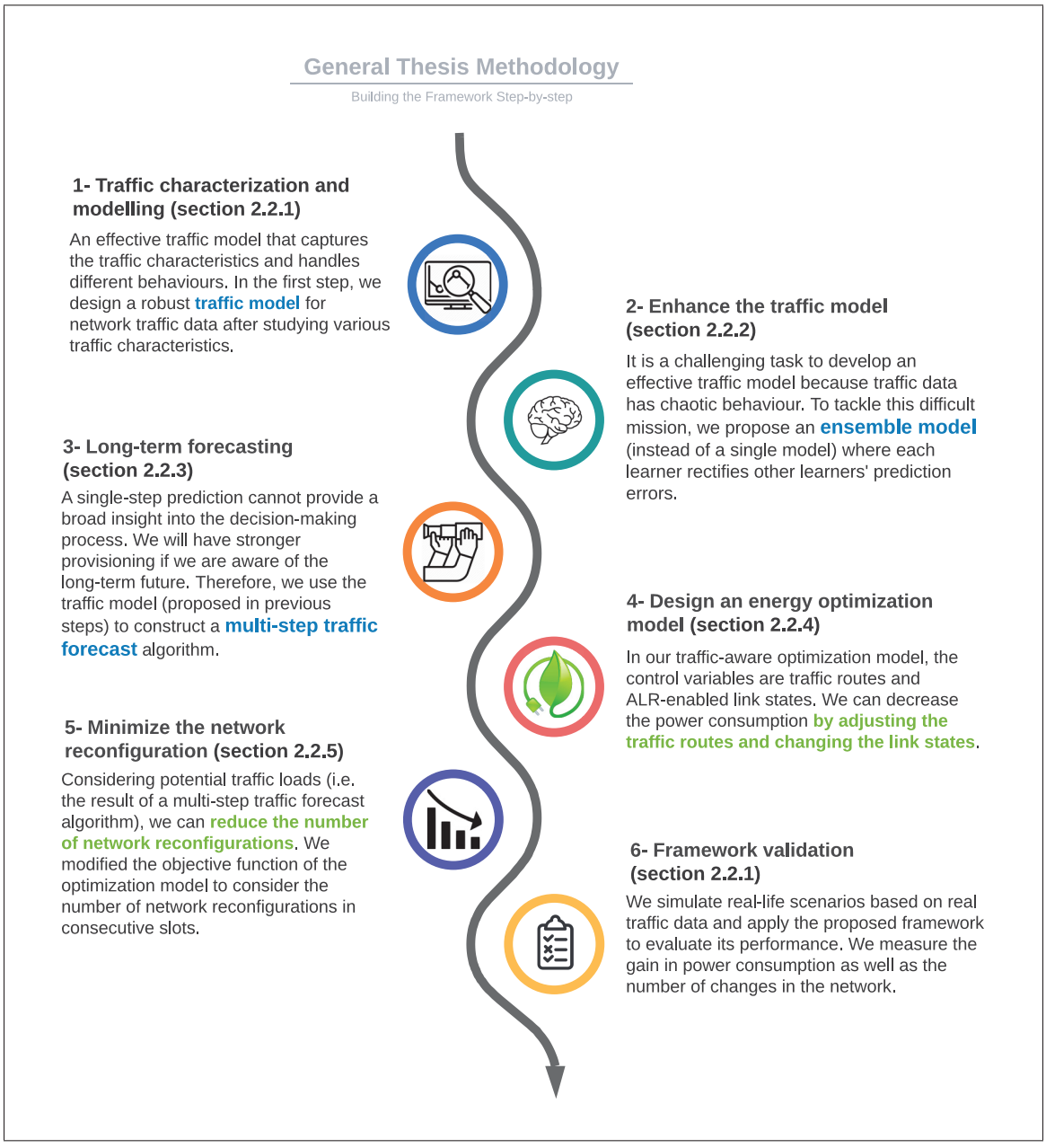


Figure 2.2 General methodology in this thesis

We build an Ensemble of GPR models while each learner is the same as the initial learner we created in the previous step.

2.2.3 Long-term Forecasting

The prediction module is responsible for offering a view of traffic loads in future time slots. The predicted values feed the network optimization model to allocate resources in order to minimize consumption. There is, however, a gap between the time the prediction information is available and the time the action is triggered. It may not be possible in a network to perform instance actions because of the consequences on the network performance and QoS. We need to apply changes at a reasonable pace. It means we need a long-term prediction that informs us about traffic fluctuations in multiple steps ahead of time. We use the algorithm built in the previous stage to design a multiple-step-ahead traffic predictor. The traffic characteristics investigated in the first stage are applicable to reduce error propagation in the multiple-step-ahead algorithm.

Figure 2.3 illustrates the prediction algorithm design in three steps. In the first step, we designed a learner based on the Gaussian Process Regression (GPR) algorithm. This step's main activity is to build a covariance function (i.e., a kernel function) that fits the traffic characteristics. The second step consists of designing an ensemble prediction algorithm based on the GPR learner built in the first step. It must be optimized to achieve the accuracy-diversity balance for the ensemble of learners. The second step's outcome is a single-step-ahead (ensemble) prediction algorithm. We use this algorithm to build a model for a multiple-step-ahead prediction forecast in the third step.

2.2.4 Design the Energy Optimization Model

This step aims to design an initial model for network energy optimization. It builds an Energy-Aware Routing (EAR) model in which traffic is routed to minimize power consumption. This initial model considers traffic load in energy saving. Nevertheless, it is important to note that it only examines the current traffic values (traffic values observed at the moment). In other words, it does not consider future traffic fluctuations (i.e., it is not a proactive optimization model). This EAR-based model's objective function is energy consumption, and its control variables are the traffic routes. The traffic load on different links is monitored through the network controller and

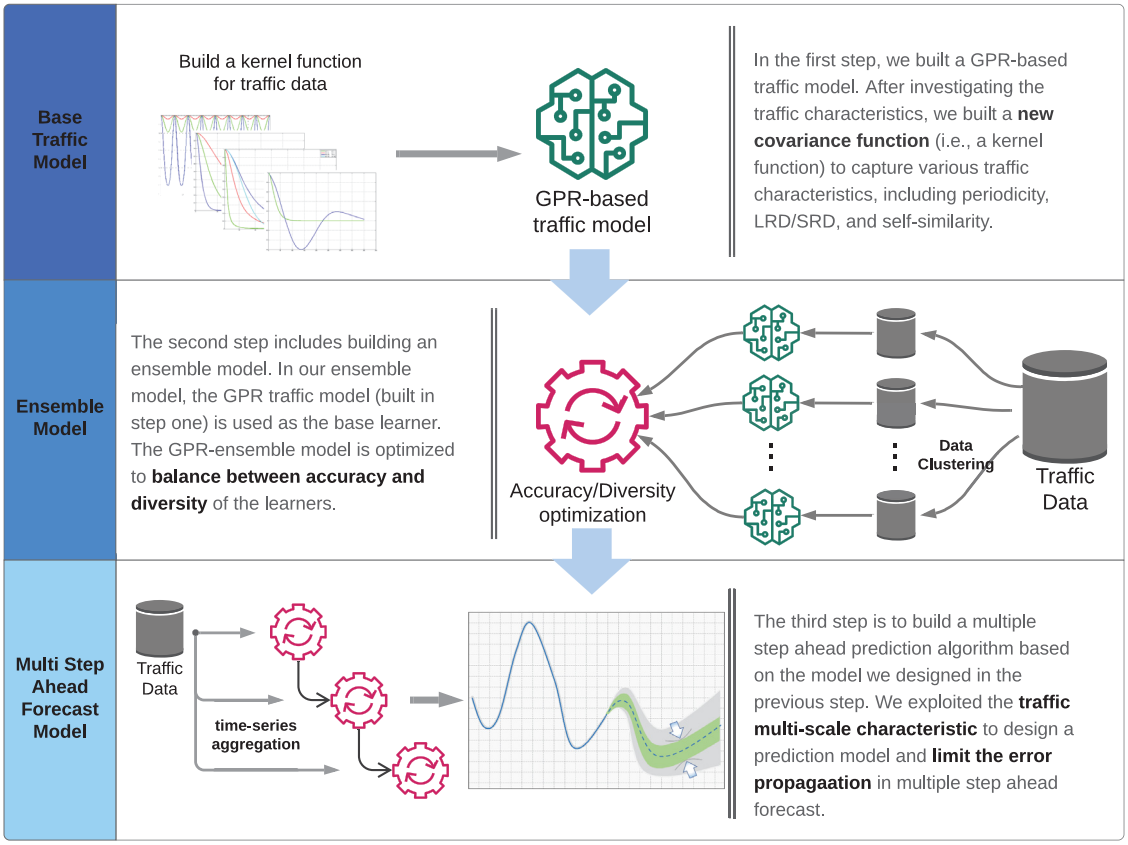


Figure 2.3 The structure of our prediction algorithm that we deigned for traffic modelling

fed to the model as input data. When the traffic loads on links change, the model responds to this change and adjusts the routes to maintain the minimum consumption.

2.2.5 Minimize the Network Reconfiguration

The model built in the preceding step has a single objective function: energy consumption. Considering production network requirements, such a model may need some improvements to be ready for real-life scenarios. This model reacts to the traffic variations, and it adjusts the traffic routes from time to time according to the traffic load fluctuations. It preserves the minimum energy consumption, but it may need to change many traffic routes repeatedly. Apparently, it is not feasible to apply such a model in a production network because the frequent traffic rerouting

leads to network performance degradation. One approach to address this issue is integrating traffic prediction into energy optimization and building a proactive model that considers future variations in traffic demand.

In this step, we integrate our multi-step-ahead prediction algorithm into the energy optimization model and develop a proactive optimization model. The main differences between the initial model and the proactive model are as follows:

- The objective function: while the initial model only considers the energy consumption as its objective function, the proactive model has a two-fold objective function: energy consumption and the number of changes in the network (e.g., traffic rerouting). The proactive model minimizes the energy consumption while changing only a few traffic routes to preserve the minimum consumption.

- The input data: in the initial model, observed traffic values are given as the input data. However, the proactive model needs the predicted values (the outcome of the multiple-step-ahead prediction algorithm) to view future traffic demands. The predicted values of traffic loads enable the proactive model to select a solution (i.e., traffic routes) for the current state of the network that requires the minimum number of changes to adapt to future demands.

2.2.6 Framework Validation

The last step is to validate the proposed energy optimization model. We use real traffic data to validate the proposed framework. In our validation step, we create scenarios based on popular traffic networks and benchmark traffic data. The validation phase includes the comparison between the proposed solution and existing solutions. We prepare two sets of experiments. First, we examine our framework's application in a backbone network and measures the achievements. In the second set of experiments, we investigate our framework's performance when applied to a data center network.

2.3 Proposed Energy Saving Framework Based on Traffic Prediction

Different modules in the methodology (defined in previous sections) have been organized in our energy-saving framework called Rate Adaptation using Prediction (RAP). Fig. 2.4 illustrates RAP. As shown, traffic data are collected from the SDN network. Traffic sampling and flow analysis are the first preprocessing steps before prediction and optimization. After performing the flow analysis on the observed traffic data, feature vectors are extracted and stored in a database. The historical data in the database is used for initializing and training the predictor, which estimates the future steps of traffic given a new sample. The outcome of the predictor is used in the optimization to find the optimal paths for flows.

In Fig. 2.4, the SDN controller is responsible for two tasks. First, it supports network monitoring and data collection. In our approach, we need traffic data and flow information on the network and links to train the model and predict the future states of the network. Second, the SDN controller provides the ability to apply the network configuration (re-routing the traffic flows and switching the link states). The optimization process changes the network configuration (i.e., re-route the traffic flows and change the link states) to implement the minimum energy consumption. The SDN controller is the means to re-configure the network and execute the optimal solution.

2.3.1 Traffic Sampling and Flow Analysis

The traffic sampling and the flow analysis modules monitor and collect the traffic samples for each flow. The definition of traffic flow depends on the type of network. For example, in an IP network, all the packets with the same protocol type, source/destination IP, and source/destination port belong to the same traffic flow. In networks where there are many traffic flows, it is not applicable to track and monitor all the flows. Instead, to reduce the monitoring and prediction overhead, it is possible to observe only the big flows. In (Y. Li et al., 2016), prediction of elephant flows has been employed to manage inter-data-center traffic. The energy consumption can be optimized by managing the big flows inside the network. In contrast, the small flows can

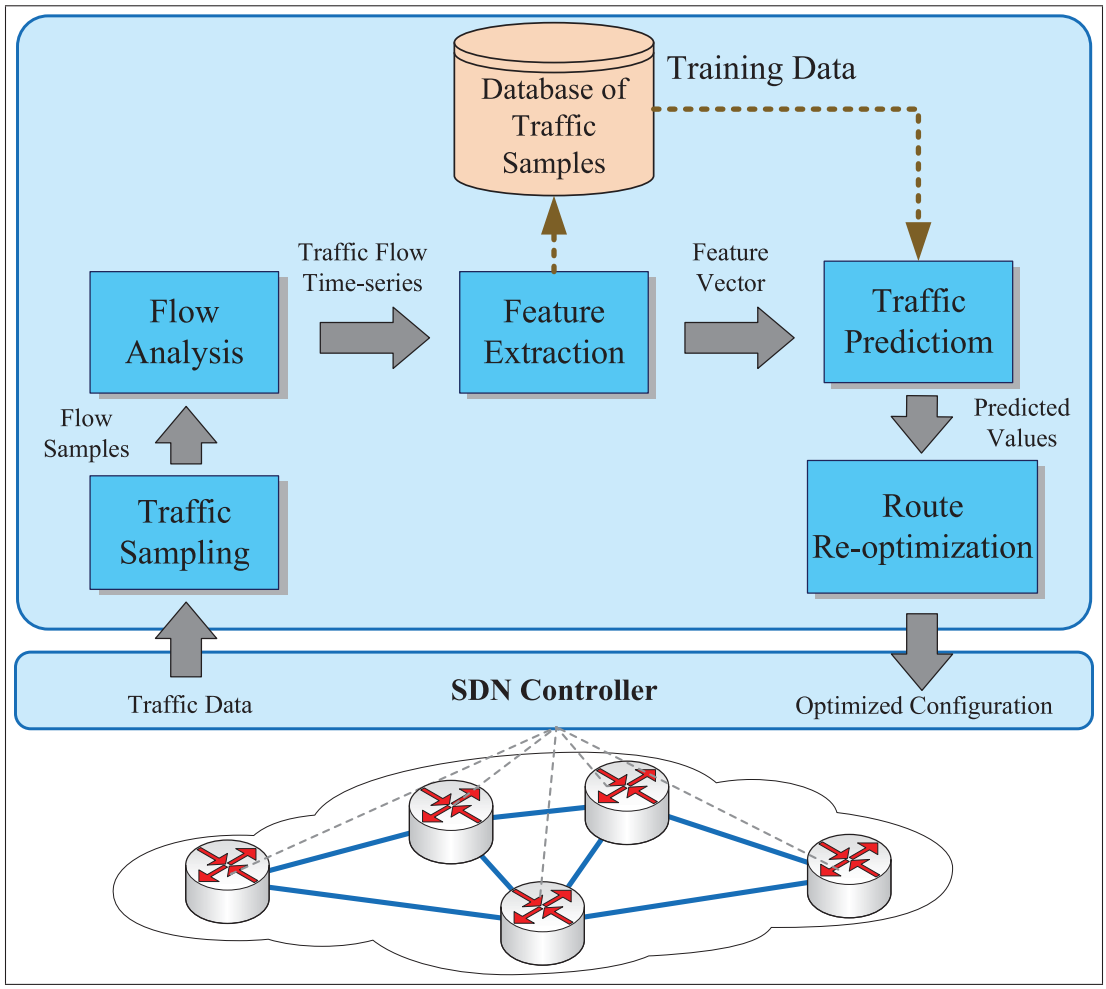


Figure 2.4 The proposed Framework for Rate Adaptation using Prediction (RAP)

be routed using a general-purpose routing algorithm (such as a shortest-path algorithm). Traffic sampling and flow analysis are required to improve data quality and eliminate the noise of small flows.

2.3.2 Feature Extraction and Traffic Prediction

These modules form a major part of studies. They include investigation of traffic characteristics, modelling, single-step-ahead traffic prediction, and multiple-step-ahead prediction. Actually, we designed a multiple-step-ahead traffic predictor in three steps:

1. First, we designed a kernel based on specific traffic characteristics and employed our proposed kernel in GPR. Our GPR predictor can handle LRD/SRD, periodic, and self-similar traffic patterns. This single-step-ahead predictor has been used as the core of our prediction algorithm in the next two steps.

2. We improved the GPR-based traffic predictor using ensemble learning. Although Gaussian Process Regression (GPR) has proven advanced prediction performance in prior research, modelling the complex behaviour of traffic using a single predictor may not be flexible enough to handle varying traffic patterns and limit prediction accuracy. The proposed GPR traffic has been optimized using ensemble learning which resolves the time-complexity issues in GPR and improves its prediction accuracy. It has been achieved by introducing a new likelihood function for GPR.

3. In the third step, our single-step-traffic predictor has been used in a hierarchical model to build a multiple-step-ahead prediction algorithm. In multiple-step-ahead traffic prediction, the goal is to predict the traffic performance measures (e.g., bandwidth, packet loss, and latency) forward in time, up to a particular horizon. In our multiple-step-ahead prediction algorithm, we focused on the multi-scale behaviour of traffic to reduce error propagation. We showed and proved that considering traffic behaviour significantly impacts preventing error propagation in our multiple-step-ahead predictor.

Briefly, we created a single-step-ahead traffic predictor based on GPR. Then we improved our single-step-ahead predictor using ensemble learning. Finally, we used our enhanced single-step-ahead predictor to create a multiple-step-ahead traffic predictor. In those three steps, we investigated and considered traffic characteristics and behaviour in the design of our algorithms to improve prediction accuracy.

2.3.3 Route Re-optimization

This module is responsible for solving the network optimization problem in order to reduce energy consumption. It finds an optimal configuration that reduces energy consumption and

prevents network changes. This module repeatedly solves the optimization problem (e.g., every 15 minutes) because the traffic demands are evolving, and link rates must be adjusted based on traffic demands. The optimization problem has two objectives: minimizing energy consumption and reducing repeated optimization changes. The traditional ALR optimization is an NP-hard problem. We formulated our optimization problem, which is mentioned as Rate Adaptation using Prediction (RAP) as an Integer Linear Programming (ILP) model. We showed that RAP is an NP-hard problem. We designed a heuristic algorithm based on simulated annealing (SA) to solve RAP. The route re-optimization module uses the predicted values (provided by the traffic prediction module) and information of the network to solve RAP and find the optimal solution.

The proposed optimization model manages the traffic routing task. In other words, it works as a routing algorithm to determine the routes for every traffic flows in the network in the network section that we want to reduce energy consumption. In this optimization model (or routing algorithm), the goal is to assign routes in order to reduce power consumption. There are a few assumptions behind this schema. First, we assume the network consists of high-speed links that can support peak traffic. So, the traffic congestion and packet loss is not a major concern. Second, the delay difference between routes is minimal and neglectable. This assumption is satisfied in many networks, such as data center networks. Also, in the optimization model, we set an upper bound on the link utilization, tuned according to the limitations. Our experiments set the link utilization upper bound equal to 70% to ensure there are no over-utilized or congested routes in the final solution. This upper bound also enhances the traffic delay and helps to avoid packet loss.

2.4 The structure of the thesis

In the rest of this thesis, we tackle the problem of network energy efficiency step-by-step. FIGURE summarizes the following chapters. In Chapters 3 to 5, we design our traffic model. Chapter 3 is dedicated to kernel design and building the GPR-based learner. We study the traffic characteristics and design a kernel function that fits the traffic data. Chapter 4 includes the study and design of the GPR ensemble model. We optimize the ensemble of GPR learners

Chapter	Task	Summary
3	Kernel Design	Chapter 3 includes the first step for prediction algorithm design. It includes our research for designing a covariance function for GPR learner that fits the traffic characteristics.
4	Ensemble Model	The GPR learner built in Chapter 3 is used as the base learner in an optimized ensemble model. Chapter 4 discusses the design of our GPR ensemble model, and the accuracy-diversity balance.
5	Multi-step forecast	Chapter 5 explains how we used our GPR ensemble model to design a prediction algorithm to provide accurate multiple-step-ahead prediction. Our design is based on the multi-scale nature of the network traffic time-series.
6	Energy Optimization Model	We built an optimization model in Chapter 6 and integrated the multiple-step-ahead prediction into the optimization model. It resulted into a proactive optimization model and provides energy saving in a network with dynamic traffic.
7	Model Evaluation	The energy optimization model is evaluated using real-traffic data and well-known network topologies. Chapter 7 includes the experimental setup and results.
8	Discussion	We will discuss the different aspects of this research and the future works.

Figure 2.5 Summary of the next chapters

by making a balance between their accuracy and diversity. We finalize our ensemble model's design and creation in Chapter 5, where the main goal is to provide the multiple-step-ahead prediction. We exploit the traffic data's multi-scale nature to eliminate the error propagation in our multiple-step-ahead prediction to achieve that goal. Chapter 6 consists of energy optimization modelling, where we integrate our prediction algorithm into the network optimization to achieve our goals. The goal is to provide an optimization model for the networks with dynamic traffic. The optimization model takes the prediction algorithm's output and allocates the resource accordingly to ensure energy efficiency and network stability. Chapter 7 includes the evaluation of the model. Finally, Chapter 8 discuss the various aspect of this thesis, and the future works and improvements.

In each chapter, we explain the sub-problem that we solve in the chapter. Also, we introduce the notation that we used. The literature review for all the sub-problems is explained in Chapter 2, and we avoid repeating them. Chapters 3 to 5 include the experimental analysis for evaluating each traffic model proposed in those chapters. However, the evaluation of the final optimization model is done in Chapter 7.

2.5 Summary

In this thesis, we investigated and addressed a two-fold problem in network energy saving. Our primary goal is to reduce energy consumption in the network using the ALR approach. A negative effect of this approach is network instability and frequent changes in network configuration. Therefore, our second goal is to prevent network changes in repeated network optimization. We employed machine learning and traffic prediction to forecast future demands and adjust the network based on predicted values to prevent sudden changes.

The organization of the thesis in the remaining chapters is as follows. In Chapter 3, we present our GPR single-step-ahead traffic predictor, which is a kernel-based machine learning algorithm. We embedded the LRD/SRD, periodicity, and self-similarity as three main traffic characteristics in the proposed kernel for GPR. In Chapter 4, we discuss our single-step-ahead traffic predictor based on ensemble learning. It is an enhancement of the GPR algorithm in Chapter 3 which trains several learners (instead of a learner) to reduce prediction error. Chapter 5 illustrates our multiple-step-ahead traffic predictor. It uses our single-step-ahead predictor (demonstrated in Chapter 4). It forms a hierarchical model based on the multi-scale behaviour of traffic to limit the error propagation in the multiple-step-ahead prediction algorithm. The RAP problem is discussed and formulated in Chapter 6. We used our multiple-step-ahead prediction algorithm in RAP to address the problem.

CHAPTER 3

GAUSSIAN PROCESS REGRESSION-BASED TRAFFIC MODELING AND PREDICTION IN HIGH-SPEED NETWORKS

A. Bayati, V. Asghari, K.H. Nguyen, M. Cheriet

Department of Automated Production Engineering, École de Technologie Supérieure, University of Quebec, Montreal, Quebec, Canada H3C 1K3

Paper published in *IEEE GLOBECOM*, December 2016

3.1 Introduction

Network traffic modeling has widely been used for developing traffic characterization (Adas, 1997), performance modeling, traffic generation (Yin et al., 2015), and etc. An important application of traffic modeling is traffic prediction that plays an important role in quality of service (QoS) enhancement in computer networks. Traffic prediction is an efficient tool for making decisions to improve network performance. For example, it has been proposed to utilize traffic prediction for predictive congestion control, proactive resource provisioning and allocation (Krithikaivasan et al., 2007), and etc. Traditionally, in this type of applications, decisions are made based on current state of network traffic which result in unstable solutions. To address this issue, traffic forecasting techniques are exploited to reduce risk of decisions by using predicted state of network traffic instead of its current state.

An appropriate traffic prediction method must response to several requirements in order to be suitable for working in a real network environment. Firstly, it must be highly accurate to reduce the risk of consequence decisions. Clearly, an inaccurate prediction algorithm deteriorates the overall performance of the network and reduces resource utilization. The second is time constraint. In computer networks, decisions must be made almost in real-time. Another requirement is the amount of computational resources required for the prediction algorithm. Traffic predictors are usually implemented in network devices such as routers and switches which

are limited in available computational resources (e.g. CPU, memory). All these restrictions make traffic prediction a challenging subject for researchers.

A traffic predictor estimates next values of a time series corresponding to network traffic given a history of previous samples. Traffic samples can be a sequence of instances of packets arrival times, or the number of arriving packets (or the total number of bytes) in successive, non-overlapping time intervals of unit length (e.g. millisecond, second, etc). In this work, we focused on the prediction of load on high-speed links. Let $X = \{X(t_0), X(t_1), X(t_2), ...\}$ byte count process where $X(t_i)$ describes the total number of bytes in the time interval t_i. Whereas process X represents a traffic trace, it might have a highly evolving behavior since it includes different patterns at different times, links, and various time-scales. Meanwhile, dominant characteristics of traffic process such as self-similarity and long-range (short-range) dependency are definitely associated with its *autocorrelation* function (Willinger et al., 2003). These two clues led us to Gaussian Process (GP) framework (Rasmussen & Williams, 2006) as a kernel-based method which allows to use particular kernels for treating certain patterns and, simultaneously enables direct access and manipulation of process autocorrelation using covariance functions. Hence, we used GPR framework to model and predict byte count process X. The contributions of our work are: 1) adapting GPR framework for traffic modeling and prediction by proposing self-similar covariance functions; and 2) using machine learning techniques to estimate Hurst parameter as a part of our prediction method.

The remaining of this chapter is organized as follows. Section 3.2 defines Gaussian Process Regression and its connection to self-similarity. Experimental results are reported in section 3.3.

3.2 Gaussian Process Regression Framework

Gaussian Process Regression (GPR) (Rasmussen & Williams, 2006) is a supervised machine learning technique that provides mapping function between input and (continuous) output data. Gaussian process framework has been used in different areas extended from classification to regression problems including: rural traffic prediction, time-varying systems (Hu et al., 2014) and

etc. Consider n pairs of input and noisy output observations, $\mathcal{D} = \{(t_i, X(t_i))|i = 0, 1, 2, ..., n-1\}$, and an unknown mapping function $f(t_i)$ while:

$$X(t_i) = f(t_i) + \varepsilon_i, \tag{3.1}$$

where ε_i is independent Gaussian noise with zero mean and variance σ^2, i.e. $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. In GPR framework, the mapping function is assumed to be a *stochastic Gaussian process* (GP) (Rasmussen & Williams, 2006). According to the definition, a Gaussian process (GP) is a set of random variables that any subset of them has joint Gaussian distribution. Gaussian process $f(t)$ is completely specified by its mean and covariance functions:

$$f(t) \sim \mathcal{GP}(m(t), k(t_i, t_j; \theta)), \tag{3.2}$$

where $m(t)$ is the mean function and $k(t_i, t_j; \theta)$ is the arbitrary covariance function and θ is the set of hyperparameters. Without loss of generality, we can assume that the mean function is equal to zero for stationary processes (i.e., $m(t) = 0$). Covariance function $k(t_i, t_j; \theta)$ plays an important role in GPR framework which will be discussed in the remaining of this section. According to Equation (3.1) and GP prior distribution of mapping function $f(t)$, the covariance function of target values $X = \{X(t_i),\ i = 0, 1, ..., n-1\}$ can be formulated as:

$$k_X(t_i, t_j; \theta) = k(t_i, t_j; \theta) + \sigma^2 \delta_{ij}, \tag{3.3}$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ in other cases.

In a regression problem, the goal is prediction of the target $X(t_*)$ for new input data t_* which does not belong to the dataset $\mathcal{D}$. To achieve this, GPR framework uses the GP prior distribution on the mapping function $f(t)$ as well as the knowledge provided by dataset $\mathcal{D}$ to draw the posterior distribution over the mapping function and then to make inferences about the conditional distribution of the function value at t_*. The GP assumption implies that joint distribution of the observed target values X and the function value at t_* is a Gaussian distribution (Rasmussen &

Williams, 2006):

$$\begin{bmatrix} X \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K + \sigma^2 I & K_* \\ K_*^\top & k(t_*, t_*; \theta) \end{bmatrix}\right), \tag{3.4}$$

where $f_* = f(t_*)$ and $t = \{t_i,\ i = 0, 1, ..., n-1\}$ is the input process in dataset $\mathcal{D}$. The element K is called *covariance matrix* of input data X and denotes a $n \times n$ matrix of covariances evaluated for all pairs in the input process, i.e., $[K]_{ij} = k_X(t_i, t_j; \theta)$, and the element K_* is $n \times 1$ matrix whose ith element is calculated as $[K_*]_{i,1} = k(t_i, t_*; \theta)$.

The conditional distribution of the function value f_* will be:

$$f_*|t, X, t_*, \theta \sim \mathcal{N}\left(\bar{f}_*, \quad cov(f_*)\right), \tag{3.5}$$

$$\bar{f}_* = K_*^\top (K + \sigma^2 I)^{-1} X, \tag{3.6}$$

$$cov(f_*) = k(t_*, t_*; \theta) - K_*^\top (K + \sigma^2 I)^{-1} K_*. \tag{3.7}$$

Equation (3.6) provides predicted value for the f_* and is called mean predictor. Also, Equation (3.7) is an estimate of its variance. Clearly, the prediction given in Equations (3.6) and (3.7) depend on the covariance function $k(t_i, t_j; \theta)$ and noise variance (or noise power value) σ^2.

3.2.1 Model selection

In each GP-based modeling, the selected covariance function has to be able to properly reflect features of data in that particular context. In this framework, covariance function $k(t_i, t_j; \theta)$ is a positive semidefinite (PSD) function that determines the nearness or similarity of input pairs t_i and t_j. The accuracy of a model highly depends on the selected covariance function. Selecting appropriate covariance function is known as *model selection* problem. Despite existence of numerous number of covariance functions and the infinite number of their combinations, it is not possible to select the best one considering all of them in the model selection phase. In fact, we first have to select a small number of covariance functions which are better for traffic modeling and prediction.

In this work, our selected model has to be able to reflect two important behavior of traffic: *self-similarity*, and *periodicity*. We exploited stationary covariance functions for this purpose because based on our assumption, byte count process X_t is a stationary process especially in long duration time-scales (e.g. 5 minutes or longer). According to the definition, a stationary process is a function of $r = t_i - t_j$. Therefore, covariance function of such a stationary process can be denoted as $k(r;\theta)$ instead of $k(t_i, t_j;\theta)$.

3.2.1.1 Self-similarity

Network traffic exhibits self-similar or fractal-like behavior (Willinger et al., 2003). That is, the network traffic looks statistically similar at different time scales. The major reason of self-similarity in network traffic is heavy-tailed distribution corresponding to large file transfer over the network (Karagiannis et al., 2004). By definition, a stochastic process $X = (X_0, X_1, X_2, ...)$ is called self-similar with self-similarity parameter H if (Tsybakov & Georganas, 1998):

$$X \overset{dis}{=} m^{1-H} X^{(m)}, \quad \forall m, \quad m \in \mathbb{N} \tag{3.8}$$

where $\overset{dis}{=}$ means equality in the sense of finite-dimensional distribution and $X^{(m)}$ is the aggregated process of X at aggregation level m:

$$\begin{aligned} X^{(m)} &= (X_0^{(m)}, X_1^{(m)}, X_2^{(m)}, ...), \\ X_i^{(m)} &= \frac{1}{m}(X_{im-m+1} + ... + X_{im}), \forall i \in \mathbb{Z}, i >= 0. \end{aligned} \tag{3.9}$$

The scalar parameter $H \in (0, 1)$ is called Hurst exponent and it quantifies level of self-similarity. Calculating Hurst parameter is not straightforward, it can only be estimated (Karagiannis et al., 2004).

The autocorrelation function (ACF) of traditional traffic models such as Poisson and Markov models decays exponentially fast. These models are called short-range dependent because each sample in the time-series generated (or modeled) by them depends only on a few previous

samples. However, real traffic measurements and analysis revealed that in most cases real traffic is a long-range dependent process. It means that ACF of real traffic traces decays hyperbolically which implies a non-summable ACF:

$$\sum_{r} \rho(r) = \infty, \tag{3.10}$$

where $\rho(r)$ is ACF of the traffic. Therefore, traditional short-range dependent models are not appropriate tools for modeling real traffic.

In addition to self-similarity, we also need to define long-range dependency (LRD) and short-range dependency (SRD). By definition, a stationary process $X = (X_0, X_1, X_2, ...)$ is called a long-range dependent process if its autocorrelation function (ACF) $\rho(r)$ satisfies (Karagiannis et al., 2004):

$$\rho(r) \sim cr^{-\beta}, \tag{3.11}$$

for parameter $0 < \beta < 1$ where c is a constant (here, two functions $f(t)$ and $h(t)$ assumed to be approximately equal ($f(t) \sim h(t)$) if $f(t)/h(t) \to 1$ when $t \to \infty$). Long-range dependency means a process exhibits significant correlations across large time scales. Also a process is short-range dependent (SRD) if $1 < \beta < 2$ which implies a summable ACF (Tsybakov & Georganas, 1998).

LRD (or SRD) corresponds to the shape of autocorrelation function while self-similarity evaluates preserving statistical properties among different time scales. It has been proved that the following linear relationship holds between Hurst exponent H of a *stationary* self-similar process and parameter β of a LRD/SRD process (Tsybakov & Georganas, 1998):

$$H = 1 - \frac{\beta}{2}. \tag{3.12}$$

For this reason, these two characteristics (self-similarity and LRD/SRD) might have been assumed the same.

The goal of our traffic modeling is capturing both types of LRD and SRD traffic. This improves generalization of proposed model. Our approach for obtaining such a goal is to select covariance functions which can lead to both strong dependent GPs with slowly decaying ACF as well as weak dependent GPs with fast decaying ACF by changing values of their hyperparameters. ACF of a LRD (or SRD) process must satisfy the condition given in (3.11) for certain values of β. Since the relation between covariance function and ACF is:

$$k(r) = \frac{\rho(r)}{k(0)}, \tag{3.13}$$

the condition given in (3.11) can be represented as the following condition for covariance functions:

$$k(r) \sim cr^{-\beta}. \tag{3.14}$$

In order to capture LRD or SRD, the covariance function in GPR has to satisfy (3.14) for $\beta \in (0, 1)$ or $\beta \in (1, 2)$ respectively. A Gaussian process with such a covariance function is able to model a self-similar traffic process X_t. Therefore, we call those covariance functions as *self-similar covariance functions*. An example of self-similar covariance functions is rational quadratic covariance function which is given by:

$$k_{RQ}(r; l, \alpha) = s^2.\left(1 + \frac{r^2}{2\alpha l^2}\right)^{-\alpha},\ \alpha > 0, l > 0, \tag{3.15}$$

where s is variance parameter, l is known as length-scale parameter and α is magnitude parameter (or shape parameter) (Rasmussen & Williams, 2006). It can be shown that rational quadratic covariance function satisfies (3.14):

$$\lim_{r\to\infty} \frac{k_{RQ}(r)}{cr^{-\beta}} = 1, \tag{3.16}$$

if:

$$\alpha = \frac{\beta}{2}. \tag{3.17}$$

Regarding (3.12) and (3.17) the relation between α, as a parameter of the model, and H, the Hurst parameter of the byte count process X_t, is elicited as:

$$\alpha = 1 - H. \tag{3.18}$$

This linear equality is very important because it proves that GPR-based modeling (using rational quadratic covariance function or any other self-similar covariance function) can capture traffic dependency and self-similarity while it does not rely on Hurst estimation algorithms. In fact, Hurst estimation is done as a part of training phase (i.e. selecting values of hyperparameters) using machine learning techniques. In compare to those previous regression-based models which relied on Hurst estimation algorithms, this is a significant advantage of our work.

3.2.1.2 Periodicity

Periodicity is the other constantly observable behavior of traffic at different time scales. In short time units, network traffic exhibits periodicity because its behavior is mostly affected by the network and its running protocols and devices in those time intervals (e.g., few seconds or shorter time scales) (Willinger et al., 2003). On the other hand, users' traffic demands seem to be cyclical with a 24-hour cycle. This phenomenon begets periodic variation in traffic patterns in long time unit (Krithikaivasan et al., 2007).

Cyclical behavior of network traffic can be exploited for improving results of the traffic predictor using a periodic covariance function. There are different periodic covariance functions. A prevalent instance is (Rasmussen & Williams, 2006):

$$k_{periodic}(r; l, p) = s^2.exp\left(-\frac{2sin^2(\pi r/p)}{l^2}\right), \tag{3.19}$$

with variance parameter s, period p and length scale parameter l. This periodic covariance function is created using squared exponential (SE) covariance function in u-space where SE

kernel is formulated as (Rasmussen & Williams, 2006):

$$k_{SE}(r;l) = s^2.exp\left(-\frac{r^2}{2l^2}\right), \tag{3.20}$$

and function u is:

$$u(r) = \Big(cos(r), sin(r)\Big). \tag{3.21}$$

3.2.1.3 Self-similarity and periodicity

Although $k_{periodic}$ captures periodicity, it can be shown that it is not a self-similar covariance function because it cannot satisfy condition in (3.14). We need a covariance function which handles both traffic characteristics (periodicity and self-similarity) simultaneously. To this end, we propose creating a *semi-periodic self-similar (SPSS)* covariance function by combining previous ones:

$$\begin{aligned} k_{SPSS}(r;\theta) &= k_{periodic}(r;s_1,l_1,p).k_{SE}(r;s_2,l_2) \\ &\quad +k_{RQ}(r;s_3,l_3,\alpha). \end{aligned} \tag{3.22}$$

where $\theta = \{s_1, l_1, p, s_2, l_2, s_3, l_3, \alpha\}$ is the set of hyperparameters. It can be shown that k_{SPSS} is a PSD and self-similar covariance function. This covariance function can handle periodic self-similar network traffic. For building such a SPSS covariance function, it is possible to use other periodic covariance functions instead of $k_{periodic}$ and also to employ other self-similar covariance functions instead of k_{RQ}. In section 3.3 we will compare results of GPR-based traffic prediction using k_{RQ}, $k_{periodic}$, and k_{SPSS}. Also we compare the performance our model with other traffic predictors.

3.2.2 Selecting values of hyperparameters

Each covariance function has a set of parameters θ known as *hyperparameters*. The values of these hyperparameters have great effect on the model accuracy, so they should be selected

carefully. We used Bayesian inference to estimate values of hyperparameters. This approach is based on maximizing the likelihood function (Rasmussen & Williams, 2006):

$$p(X|t,\theta) = \frac{1}{(2\pi)^{N/2}|K|^{1/2}} exp\left(-\tfrac{1}{2}X^T K^{-1} X\right) \tag{3.23}$$

This likelihood function simply obtained by considering $X \sim \mathcal{N}(0, K + \sigma^2 I)$. Equivalently, we can maximize *log-likelihood* function:

$$log\left(p(X|t,\theta)\right) = -\frac{1}{2}X^T K^{-1} X - \frac{1}{2}log(|K|) - \frac{N}{2}log(2\pi) \tag{3.24}$$

Search direction algorithms (e.g. gradient descent) are appropriate for maximizing log-likelihood function. This requires gradient (i.e. partial derivative with respect to hyperparameters) of log-likelihood function:

$$\frac{\partial}{\partial\theta_i} = \frac{1}{2}X^T K^{-1}\frac{\partial K}{\partial\theta_i}K^{-1}X - \frac{1}{2}tr\left(K^{-1}, \tfrac{\partial K}{\partial\theta_i}\right) \tag{3.25}$$

where $tr(A)$ is trace of (square) matrix A.

3.3 Experimental Results

Our experimental result is two-fold: 1) validation of relation between Hurst parameter H and parameter α given in (3.18), and 2) Comparison of the proposed model and previous work using the same benchmark.

3.3.1 Hurst Estimation

We evaluated the connection between Hurst exponent H and hyperparameter α in (3.18) as follows. First, we trained our GPR model (employing rational quadratic covariance function) using real traffic traces to estimate value of hyperparameter α. Then, we used well-known Hurst

estimation algorithms on the same traffic traces to estimate value of Hurst exponent. Finally, we compared H and $1 - \alpha$ to see the relation between them.

The training phase including maximum likelihood estimation (MLE) explained in section 3.2.2 uses 1500 traffic load samples in time-scale of 1 second (i.e., each sample represent number of byte transmitted on the link during 1 second) from two different sources. The first set of traces contains *Anonymized Internet Traces* from Center for Applied Internet Data Analysis (CAIDA) (*The CAIDA UCSD Anonymized Internet Traces*, 2008-2015). This set consists traffic traces collected from high-speed Internet backbone links at different days of different years (from 2010 to 2014). The average bit rates of those traces are in the range of Gbit/s and their duration is 1 hour. The second set includes traffic traces from Waikato VIII captured at network edge of the University of Waikato (*Waikato VIII 2011*, 2011). Each trace in the second set contains 24 hour of traffic and their average bit rates are in the range of Mbit/s.

Meanwhile, Hurst estimation algorithms have been applied on the same traffic data. There are two categories of Hurst estimators: time-domain estimators which investigate particular statistical properties in time domain (e.g., R/S, and variance of residuals), and frequency-domain estimators which operate in frequency or wavelet domain (e.g., periodogram, and Whittle) (Karagiannis et al., 2004). Regarding existing Hurst estimation methods, the corresponding results are often conflicting and no method is known as the most accurate and robust estimator. However, it has been shown in most cases, frequency-domain estimators (especially Whittle and Periodogram) seem to be more accurate (Karagiannis et al., 2004). We used the software package SELFIS to acquire results of different Hurst estimators (Karagiannis et al., 2004).

Table 3.1 shows estimated values for hyperparameter α and Hurst exponent. The first two columns contain estimated values of α and $1 - \alpha$ respectively. Other columns illustrate estimated value of H using different Hurst estimators. As shown $1 - \alpha$ is close to estimated values of H for most of the traffic sequences, especially to estimation resulted from Whittle. This shows $1 - \alpha$ is a good estimate for H. Therefore, the proposed GPR model with self-similar covariance function is able to handle self-similarity of traffic. Figure 3.1 illustrates the relation between

negative log-likelihood and value of the hyperparameter α during the process of MLE using the traffic trace Waikato-01. In fact, the minimum of negative log-likelihood (i.e. the maximum of log-likelihood) corresponds to the value of 0.22 for α.

Table 3.1 Comparison between estimated values for α and H

Traffic Data	α	$1-\alpha$	Estimated Hurst exponent (H)			
			R/S	Variance of Residuals	Perio-dogram	Whittle
CAIDA-01	0.06	0.94	0.79	1.06	0.99	0.97
CAIDA-02	0.13	0.87	0.8	0.97	0.92	0.85
CAIDA-03	0.02	0.98	0.8	1.16	1.13	0.97
CAIDA-04	0.22	0.78	0.77	0.9	0.84	0.87
CAIDA-05	0.15	0.85	0.77	0.97	0.79	0.79
CAIDA-06	0.38	0.62	0.79	0.87	0.74	0.92
CAIDA-07	0.13	0.87	0.82	1.08	0.99	0.88
CAIDA-08	0.2	0.8	0.81	1.0	1.06	0.88
CAIDA-09	0.1	0.9	0.82	1.03	0.98	0.97
CAIDA-10	0.01	0.99	0.7	1.19	1.11	1.0
Waikato-01	0.22	0.78	0.74	0.98	0.9	0.88
Waikato-02	0.11	0.89	0.78	1.04	1.04	0.97
Waikato-03	0.15	0.85	0.77	0.95	0.93	0.96
Waikato-04	0.08	0.92	0.8	1.13	1.13	0.99
Waikato-05	0.15	0.85	0.78	0.97	1.05	0.91
Waikato-06	0.05	0.95	0.82	1.04	0.85	0.87
Waikato-07	0.2	0.8	0.78	0.97	0.94	0.94
Waikato-08	0.13	0.87	0.84	1.0	1.07	0.94
Waikato-09	0.09	0.91	0.81	1.0	1.18	0.97
Waikato-10	0.24	0.76	0.73	0.88	0.74	0.9

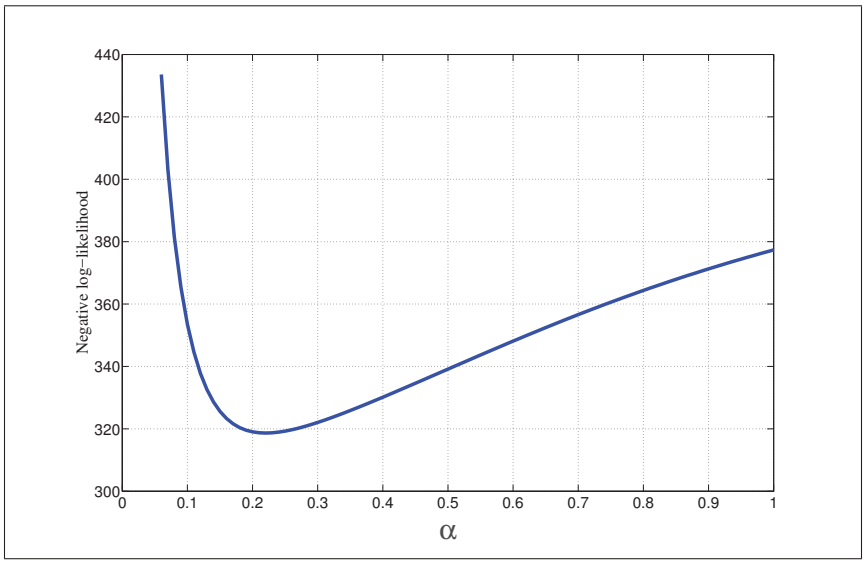


Figure 3.1 Estimation of the value of α in the MLE process using the traffic trace Waikato-01

3.3.2 Traffic Prediction

Like well-known regression-based methods, our proposed GPR model has been applied on real traffic traces to compare their prediction accuracy at various time-scales. We used bandwidth traffic traces monitored on different links of Abilene Internet2 Network in 5 minutes intervals over the period from 2005-10-16 to 2007-07-31 (*Internet2 Abilene Network*, 2007).

Four different traffic predictors have been employed for comparison. *Recent value prediction* is the simplest prediction method in which the recent observed value is used as the predicted value. The second model is *AR(4)* (i.e. autoregressive model of order 4) in which parameters are estimated using Burg's lattice-based method (Brockwell & Davis, 2013). *ARMA(4,1)* and *ARIMA(4,1,1)* are other models utilized in this experiment and MLE has been used for estimation of their parameters.

In this experiment, 10 traffic traces from Abilene Internet2 have been used. The experiment has been done with more than 30 randomly selected points of each traffic trace. In each run, 100

Method	NMSE	AIC
RecVal	0.24	-1.62
AR	0.19	**-1.81**
ARMA	0.21	-1.69
ARIMA	0.19	-1.70
GPR (SE)	0.21	-1.63
GPR (RQ)	**0.18**	-1.79
GPR (SPSS)	**0.18**	-1.54

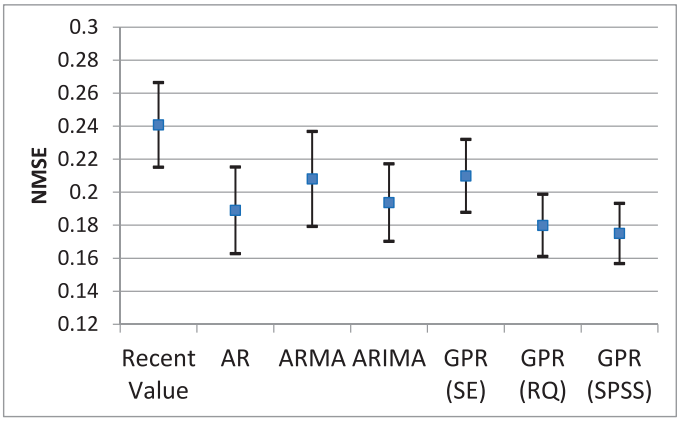


Figure 3.2 Prediction error of different models (time-scale of 5 minutes)

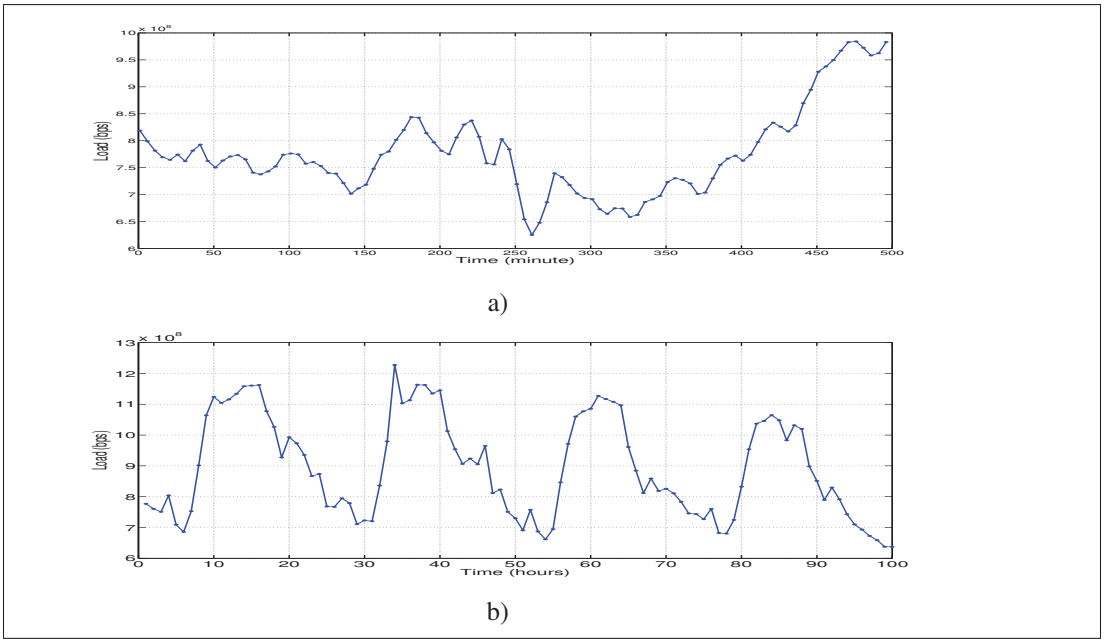


Figure 3.3 Two real traffic sequences with 100 data points monitored at time-scales of (a) 5 minutes and (b) 1 hour

data samples have been used for training (i.e. estimating model parameters) and 50 data points have been used as the observed samples for doing prediction.

The predictors have been evaluated using two different metrics: Normalized Mean Squared Error (NMSE), and Akaike Information Criterion (AIC) (Brockwell & Davis, 2013). NMSE (also known as average relative prediction variance) measures prediction error:

$$NMSE = \frac{1}{\sigma^2 N} \sum_{n=1}^{N} (x_n - \hat{x}_n)^2 \tag{3.26}$$

in which x_n is the bandwidth value of real traffic and $\hat{x}_n$ is the predicted value, and σ^2 is variance of the real traffic over N samples. A value of $NMSE = 0$ corresponds to the perfect predictor while $NMSE = 1$ is obtained when the average of data samples is used as the predicted value. On the other hand, AIC is a measurement for both prediction error and model complexity (Brockwell & Davis, 2013):

$$AIC = ln(NMSE) + 2 * \frac{p}{N} \tag{3.27}$$

where p is number of parameters in the model (i.e. indication of model complexity). The value of AIC is $-\infty$ for a perfect predictor and its value increases when prediction error or number of parameters increase.

Results shown in Figure 3.2 indicate GPR with self-similar covariance functions have higher prediction accuracy compared to others. However, in the term of AIC, AR has the best result which shows that AR(4) is a simple model (with only four parameters) while GPR (with SPSS covariance function) has 8 hyperparameters. Nevertheless, GPR (with RQ covariance function) has a small error and at the same time its AIC is acceptable. Figure 3.2 also illustrates variance of NMSE values for GPR method (using self-similar covariance functions) is smaller than others which is an indication of generalization ability and stability of GPR.

Traffic patterns are not the same at different time-scales. Figure 3.3 displays two traffic traces from the same link but monitored at different time-scales. Traffic has periodic pattern at time-scale of 1 hour while it is not periodic at time-scale of 5 minutes. Therefore we have to investigate performance of different models at different time-scales. One model has to be flexible

enough to handle different patterns and behaviors to have a proper performance at different time-scales.

Figure 3.4 presents prediction error of different algorithms at different time-scales. We obtained these results by applying traffic predictors on data from one link at time-scales of 5, 10, 15, 30, 45, 60, 75, 90, 105, and 120 minutes. As seen, some prediction algorithms are not stable as their accuracy varies highly at different time-scales. This outcome can be explained through variable traffic behavior at various time intervals depicted in Figure 3.3. Overall, GPR (with SPSS covariance function) outperforms other algorithms in the terms of NMSE accuracy because its structure combines self-similar and periodic covariance functions. Therefore, it treats both periodic and self-similar patterns in the same time, so it is highly accurate and stable in most of time-scales.

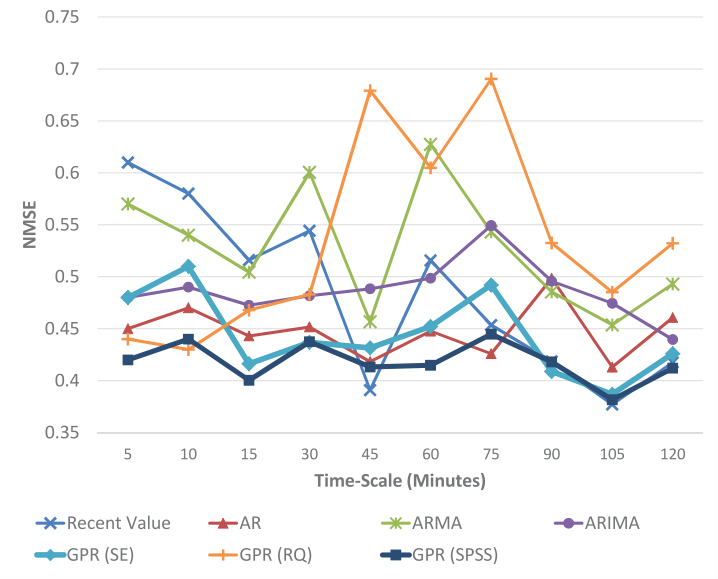


Figure 3.4 Prediction error (NMSE) of the models at different time-scales

3.4 Summary

Traffic exhibits variable behavior and various patterns in different time-scales analysis which challenges traffic models in making accurate prediction. The Kernel-based model (GPR) we proposed in this chapter addresses this issue by capturing each behavior using a particular kernel and employing kernels combination to handle complex behaviors. As its major challenge, this method requires a precise model selection approach (i.e. making decision about number and types of covariance functions). We addressed this issue by proposing self-similar and SPSS covariance functions. Experimental results show that SPSS covariance function which focuses on traffic self-similarity and periodicity, is more stable and outperforms existing models at different time-scales. In future, the proposed model can be extended to a traffic generator considering aforementioned traffic characteristics (i.e., periodicity and self-similarity) and other characteristics such as burstiness.

CHAPTER 4

GAUSSIAN PROCESS REGRESSION ENSEMBLE MODEL FOR NETWORK TRAFFIC PREDICTION

A. Bayati, K.H. Nguyen, M. Cheriet

Department of Automated Production Engineering, École de Technologie Supérieure, University of Quebec, Montreal, Quebec, Canada H3C 1K3

Paper published in *IEEE ACCESS*, September 2020

4.1 Introduction

Network traffic prediction is an efficient tool for improving proactive resource scheduling and traffic engineering. It has been utilized to solve various problems such as resource management, mobile data offloading, data-center traffic management, etc. Traffic prediction is challenging because the behaviour of network traffic is affected by many factors including users' behaviour, network protocols, topology, and management policies. Many predictive models have been proposed based on different algorithms including neural networks (Morales et al., 2017), kernel-based methods (Haghighat et al., 2015), time-series models (Chen, 2017), etc. They rely mainly on a single learner to train and forecast the traffic data. Although those models are efficient for specific types of network traffic, they lack flexibility and generalization to capture the complex and varying behaviour exhibited in traffic time-series. This work investigates a new approach based on ensemble learning to fulfill this shortcoming.

Ensemble learning (Polikar, 2012) is a recent direction of research in machine learning algorithms in which a group of learners are trained and combined to increase the accuracy and generalization of the prediction. The base learners in an ensemble model can be either the same type of machine learning algorithm or different types of algorithms. The learners are trained on the same training set or different subsets of the training set. The outcome of the ensemble model for an input sample is calculated by combining the prediction results of individual learners

on this sample. Combining techniques include voting (for classification), weighted sum (for regression), etc. Many ensemble models have been proposed for classification (Khan & Ahmad, 2018), regression (J. Liu et al., 2015), online learning (Y. Sun et al., 2016), and learning concept drift (Minku et al., 2009). Nevertheless, there are a limited number of ensemble models for time-series prediction (Rahman et al., 2016). In this work, we propose a new ensemble model for time-series prediction and apply it on the complex time-series of network traffic.

An ensemble of learners outperforms any of its members if the individual learners are both diverse and accurate (Polikar, 2012). The *accuracy* of a learner is measured based on the difference between the learner's prediction and the actual target value. Accuracy is a necessary condition to avoid poor learners to obtain the majority of votes. *Diversity* measures the discrepancy among the outputs of the learners. It is required to ensure the learners make uncorrelated errors and achieve better generalization performance (Polikar, 2012). In a diverse ensemble, the learners rectify the prediction errors of each other. Maintaining accuracy and diversity is the cornerstone of ensemble learning. Existing ensemble models employ various techniques to increase these two factors. For example, accuracy has been increased by exploiting effective features, finding appropriate lag (in time-series prediction) (Yan, 2012), and optimal parameters for base learners (Rahman et al., 2016). Diversity has been increased using various methods such as bagging, boosting, and employing different base learners (Polikar, 2012).

Accuracy and diversity are two conflicting objectives (C. Zhang et al., 2017). The increase in diversity among learners can lead to a decrease in their accuracy. Finding the optimal balance between accuracy and diversity improves the ensemble prediction performance. Despite this relationship, many ensemble algorithms employ separate techniques to enhance both objectives *independently*. They do not consider the relationship between accuracy and diversity as two conflicting objectives. Therefore, they cannot guarantee the optimal balance between accuracy and diversity to minimize the ensemble prediction error and maximize its generalization. To address this shortcoming, the trade-off between accuracy and diversity must be reflected in the training phase of the ensemble model. This idea forms the basis of our approach to ensemble modelling.

In this work, we propose an ensemble model for traffic time-series prediction which optimizes both the accuracy of learners and diversity between them. During the training phase of the proposed model, the accuracy of learners and diversity between their outputs are adjusted as two parameters to find the global optimum point which minimizes the ensemble prediction error. The control variables that optimize the balance between accuracy and diversity are the number of base learners, their parameters, and their training sets. The training phase determines the number of required learners in the ensemble, values of their parameters, and the samples that must be assigned to the training set of each learner.

We designed a divide-and-conquer approach to distribute the process of finding the optimal accuracy-diversity balance between the base learners. It reduces the computational requirements of the training phase and improves the training performance for large datasets. In this approach, each base learner provides a trade-off between accuracy and diversity in a small region of feature space. The feature space is firstly segmented into multiple subspaces, each is assigned to a base learner. Then, each base learner provides the balance between accuracy and diversity in a local area of the feature space. In other words, each base learner maintains the balance between two terms: accuracy on its corresponding segment, and diversity on the adjacent segments.

We employ Gaussian Process Regression (GPR) as the base learner in our ensemble model. GPR is a kernel-based Bayesian method and a powerful tool for regression and function approximation (Rasmussen & Williams, 2006). In prior work, we showed GPR can handle different traffic characteristics including long/short range dependencies (LRD/SRD), self-similarity, periodicity (Bayati et al., 2016), and multiscale behaviour (Bayati et al., 2018). The standard GPR is not appropriate to be used in the proposed ensemble model because the Gaussian likelihood in GPR considers only the prediction accuracy. We changed the training phase of GPR to make it compatible with the requirements of our ensemble model by introducing a new likelihood function which considers both accuracy and diversity.

The proposed model has two advantages over the existing time-series ensemble models. First, it optimizes the accuracy-diversity trade-off to improve the prediction performance. Second, it is appropriate for large datasets. The main contributions of the proposed model are as follows.

- Our model considers both accuracy and diversity as two conflicting objectives and finds the optimal balance between them to minimize the ensemble prediction error. This is the main focus of this work.

- We proposed a divide-and-conquer approach to optimizing the balance between accuracy and diversity during the training phase. In this approach, each learner considers a small portion of the training samples in a region of the feature space during the accuracy-diversity optimization. Comparing to the approaches when each learner enhances the accuracy-diversity balance considering the whole training set, our approach leads to an unsophisticated objective function for optimizing the parameters of the base learners. It resolves the issues related to complexity of the problem and allows the model to achieve accuracy-diversity balance for large datasets in a reasonable time.

- We proposed a new *GPR ensemble likelihood function* which improves the the standard GPR to satisfy the requirements of our ensemble model.

- We compared our ensemble model with well-known time-series prediction algorithm such as Long Short-Term Memory (LSTM), autoregressive integrated moving average (ARIMA), fractional autoregressive integrated moving average (FARIMA), Support Vector Regression (SVR), Least absolute shrinkage and selection operator (LASSO), Gradient Tree Boosting (GTB), Random Forest (RF), and Extremely Randomized Trees (ERT) using real traffic datasets.

The remainder of the chapter is organized as follows. Three main components used in the proposed model (i.e., the concept of ensemble learning, GPR, and clustering algorithm) are explained in Section 4.2. The phases of the proposed GPR ensemble model are detailed in Section 4.3. Section 4.4 presents our experimental results. Finally, the conclusion is drawn, and we outline the future work.

Table 4.1 Model notation

Variable	Description
x_i	feature vector of i-th sample
y_i	label (or target value) of i-th sample
X	the set of feature vectors
Y	the set of labels
$\mathcal{D}$	dataset of samples
d	length of feature vector (i.e., number of features in x)
$k(x_i, x_j, \theta)$	covariance function with hyperparameters θ
K	covariance matrix
o	occupation number in DP clustering algorithm
α	innovation parameter in DP clustering algorithm
$f_m(x_i)$	prediction of learner m for sample x_i
$\hat{f}_{ens}(x_i)$	the final prediction of GPR ensemble model for x_i
$\mathcal{P}$	standard Gaussian likelihood in GPR
$\mathcal{L}$	the proposed GPR ensemble likelihood function
$\mathcal{V}$	the measure of accuracy-diversity balance in $\mathcal{L}$
$\mathcal{E}$	the measure of accuracy in $\mathcal{V}$
$v_m(x_i)$	effect of expert m on the accuracy-diversity balance for x_i
$\mathcal{I}$	the measure of diversity in $\mathcal{V}$
M	total number of experts in the GPR ensemble model

4.2 Background

This section describes the main building blocks of the model including ensemble learning, Gaussian process regression (GPR), and Dirichlet process (DP) clustering.

4.2.1 Ensemble Learning

An ensemble learning system consists of a set of individual learners where each learner provides an estimate of the target variable. It predicts the target variable by combining the results of all the individual learners. There are three main steps in building an ensemble learning system:

ensemble generation, training each model, and combining the results (Ferreira & Figueiredo, 2012). The learners can be the same or different types of machine learning algorithms. The success of the ensemble learning system depends on the accuracy and diversity among the results of the learners (Minku et al., 2009).

Accuracy is measured based on different metrics such as mean squared error (MSE), normalized mean squared error (NMSE) (Bayati et al., 2016), etc. In an ensemble model with M learners (f_i), diversity over N data points (x_i) can be measured as:

$$\mathcal{I} = \sum_{m=1}^{M} \sum_{i=1}^{N} \left(f_m(x_i) - \bar{f}_{\neg m}(x_i) \right)^2, \tag{4.1}$$

$$\bar{f}_{\neg m}(x) = \frac{1}{M-1} \sum_{\substack{j=1 \\ j \neq m}}^{M} f_j(x) \ . \tag{4.2}$$

Equation (4.1) is the basis for *negatively correlated ensemble learning* (Ferreira & Figueiredo, 2012). It measures the difference between the prediction result of learner f_m and the average of outcomes of other experts in the ensemble.

4.2.2 Gaussian Process Regression (GPR)

We used GPR as the base learner in our ensemble model. According to the definition, a Gaussian process (GP) is a set of random variables that any subset of them has a joint Gaussian distribution. Gaussian Process Regression (GPR) (Rasmussen & Williams, 2006) is a supervised machine learning algorithm that provides the mapping function between the input $X = \{x_i\}$ and (continuous) output $Y = \{y_i\}$. Consider n pairs of input and noisy output observations, $\mathcal{D} = \{(x_i, y_i) | i = 0, 1, 2, ..., n-1\}$, and the unknown mapping function $f(x_i)$:

$$y_i = f(x_i) + \varepsilon_i, \tag{4.3}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ is the independent Gaussian noise, and $f(X) \sim \mathcal{GP}(\bar{m}(X), k(x_i, x_j; \theta))$. $\bar{m}(X)$ denotes the mean function and $k(x_i, x_j; \theta)$ is the arbitrary covariance function with

hyperparameters θ. We employ the *Rational Quadratic* (RQ) covariance function defined as (Bayati et al., 2016):

$$k_{RQ}(x_i, x_j; l, \alpha_c) = s^2.\left(1 + \frac{(x_i - x_j)^2}{2\alpha_c l^2}\right)^{-\alpha_c},\ \alpha_c > 0,\ l > 0, \tag{4.4}$$

with the set of hyperparameters $\theta = \{\alpha_c, l, s\}$. In (Bayati et al., 2016), we showed that RQ covariance function can handle traffic characteristics such as LRD/SRD and periodicity. Periodicity is a traffic pattern that is consistently observable at different time scales of traffic data. Thus, using periodic covariance functions to model such behaviour is a well-known approach. For example, in (Bayati et al., 2016), we proposed a *semi-periodic self-similar (SPSS)* covariance function for GPR, capturing LRD/SRD and patterns of periodicity in traffic data. In this work, however, the periodic pattern may not be perceptible by individual learners because the data samples are clustered into smaller subsets. The samples in a cluster are expected to have the same pattern, but they do not necessarily illustrate periodic behaviour. Therefore, we used the RQ covariance function for modelling.

The goal is to predict y_* for previously unobserved sample x_* which does not belong to $\mathcal{D}$. The conditional distribution of $\bar{f}_* = f(x_*)$ is:

$$\bar{f}_* | D, x_*, \theta \sim \mathcal{N}\left(\hat{f}_*,\quad V_*^2\right), \tag{4.5}$$

$$\hat{f}_* = K_*^\top (K + \sigma^2 I)^{-1} Y, \tag{4.6}$$

$$V_*^2 = k(x_*, x_*; \theta) - K_*^\top (K + \sigma^2 I)^{-1} K_*. \tag{4.7}$$

Equation (4.6) provides the predicted value. K is the $n \times n$ *covariance matrix* of X, i.e., $[K]_{ij} = k(x_i, x_j;\ \theta)$. K_* is a $n \times 1$ vector where $[K_*]_i = k(x_i, x_*; \theta)$.

Since the values of hyperparameters have a significant impact on the prediction accuracy, they have to be selected carefully. We used Bayesian inference to estimate the values of hyperparameters. This approach is based on the maximization of the Gaussian likelihood

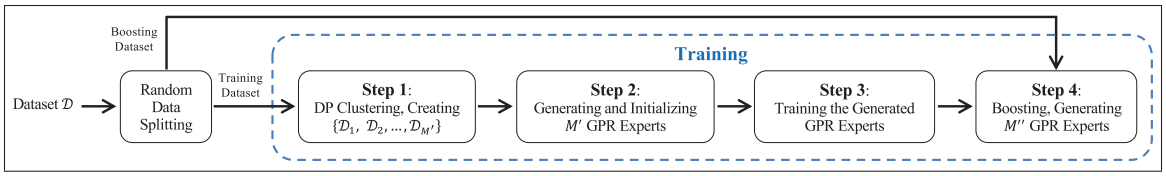


Figure 4.1 Creating and training the GPR ensemble model

function (Rasmussen & Williams, 2006):

$$p(Y|X,\theta) = \frac{1}{(2\pi)^{n/2}|K|^{1/2}} exp\left(-\tfrac{1}{2}Y^T K^{-1}Y\right) . \tag{4.8}$$

This likelihood function simply obtained by considering $Y \sim \mathcal{N}(0, K + \sigma^2 I)$. Equivalently, we can maximize the *log-likelihood function* (Bayati et al., 2016):

$$log\left(p(Y|X,\theta)\right) = -\frac{1}{2}Y^\top K^{-1}Y - \frac{1}{2}log(|K|) - \frac{n}{2}log(2\pi) \tag{4.9}$$

This log-likelihood function considers only the prediction accuracy. We designed a new likelihood function for GPR to satisfy the requirements of our ensemble model regarding the accuracy-diversity optimization.

4.2.3 Dirichlet Process Clustering

A clustering algorithm is required to partition the feature space into multiple subspaces. Any clustering algorithm that satisfies both following requirements can be used. First, the number of subspaces is not known as prior knowledge and depends on the training samples. The clustering algorithm has to determine the number of subspaces automatically. Second, the probability that a new data sample belongs to an existing cluster is required in our ensemble model (in the training and prediction) and the clustering algorithm must be able to estimate such a probability. Dirichlet process (DP) clustering is a natural choice to provide both requirements.

DP has been used in nonparametric Bayesian models (S. Sun & Xu, 2010). It is a distribution over distributions, i.e., each draw from a Dirichlet process is itself a distribution while the marginal

distributions are Dirichlet distributions. A $DP(\alpha, G_0)$ is described by a base distribution G_0 and a positive scalar α, usually indicated as the *innovation parameter*. Consider a sample distribution G drawn randomly from a DP, and random variables c_i sampled from G:

$$G|\alpha, G_0 \sim DP(\alpha, G_0)\,, \tag{4.10}$$

$$c_i|G \sim G, i = 1...n\,. \tag{4.11}$$

Random variable c_i indicates the cluster of sample x_i.

Assume the indicators (c_i) take M unique values. It can be shown the conditional probability of a single indicator c_i given other indicators when integrating out G and letting M tends to infinity exhibits a clustering effect (Rasmussen & Ghahramani, 2001):

$$p(c_i = m|c_{\neg i}, \alpha) = \frac{o_{\neg i,m}}{M - 1 + \alpha}\,, \tag{4.12}$$

$$p(c_i \neq c_j,\ \forall j \neq i|c_{\neg i}, \alpha) = \frac{\alpha}{M - 1 + \alpha}\,, \tag{4.13}$$

where $c_{\neg i}$ is the set of all indicators excluding the ith indicator c_i, and $o_{\neg i,m}$ is the *occupation number* of the cluster m ignoring the value of c_i. In the standard DP, the occupation number is calculated as:

$$o_{\neg i,m} = \sum_{j \neq i} \delta(c_j, m)\,. \tag{4.14}$$

According to the equations (4.12) and (4.13), the indicator c_i belongs to a new cluster (with probability $\frac{\alpha}{M-1+\alpha}$) or the existing cluster m (with probability $\frac{o_{\neg i,m}}{M-1+\alpha}$). The number of clusters is not predetermined, and it depends on the value of α and the number of sample points. A larger value of α induces a higher tendency of creating more clusters, and a smaller value of α leads to a lower number of clusters.

4.3 GPR Ensemble Model For Traffic Prediction

This section presents the proposed GPR ensemble model. The algorithm is explained in three phases: preprocessing, training, and prediction. The proposed model requires three steps for being used in real applications. First, the input data (i.e., traffic time-series) must be processed and formatted to create a dataset. Second, the model is trained using the dataset. Finally, the trained model is employed for prediction. For example, the proposed model may be used for proactive resource allocation in data centers. For such an application, we can employ an instance of the trained model in the network controller, where (i) it can access the traffic data on the links, and (ii) its outcome is accessible to the network controller for resource allocation. In preprocessing phase, traffic samples are prepared and structured in a dataset to be used in the training and prediction phases. It normalizes the time-series and eliminates their trends. In the training phase, the GPR learners are generated and trained considering learners' accuracy and their diversity. In the prediction phase, the outputs of base learners for a new sample are combined to create the ensemble prediction. The ensemble prediction is a weighted sum of individual outputs. The input of this model is the time-series of traffic bit-rates (presented as t_i in this section), and the output is the predicted traffic bit-rates (presented as $f_{ens}(x_i)$). The type and source of traffic bit-rates depend on the application. For example, it can be traffic bit-rates on the links (for predicting the link utilization) or traffic load between each pair of points in the network (for predicting the traffic matrix). In our experiments, both traffic types are considered (i.e., CAIDA and Waikato datasets consist of link utilization, but Abilene datasets consist of traffic matrix).

4.3.1 Preprocessing

In the preprocessing phase, two transformations are applied to the traffic time-series, and then the dataset is created using the transformed samples. The input data of this step includes time-series values shown as t_i. This step's output is a dataset (called $\mathcal{D}$) consisting of a feature vector (presented as x_i) and a target value (shown as y_i) for each data point. The first transformation is known as the *difference* operator which eliminates the nonstationarity and trend in the time-series

(Montgomery et al., 2015):

$$r_i = (1 - B)t_i = t_i - t_{i-1}\,, \tag{4.15}$$

where t_i is the original traffic sample, and B is the backshift operator. The second transformation is the normalization of input using a sigmoid function:

$$q_i = \frac{2}{1 + e^{-a_n r_i / l_n}} - 1\,, \tag{4.16}$$

where l_n is the link capacity, and a_n is a coefficient value that alters the shape of the normalization function. The function (4.16) maps the input value into the range of $[-1, 1]$.

The transformed traffic data, q_i, is used to create the training dataset $\mathcal{D} = \{(x_i, y_i)|i = 0, 1, 2, ..., n-1\}$ in which feature vector x_i (with length d) and label y_i are:

$$x_i = [q_{i-1}, q_{i-2}, ...q_{i-d}]\,, \tag{4.17}$$

$$y_i = q_i\,. \tag{4.18}$$

It means for data point i, the target value is q_i, and the feature vector is the d values observed before q_i (i.e., $q_{i-1}, q_{i-2}, ...q_{i-d}$) where d is the size of the feature vector.

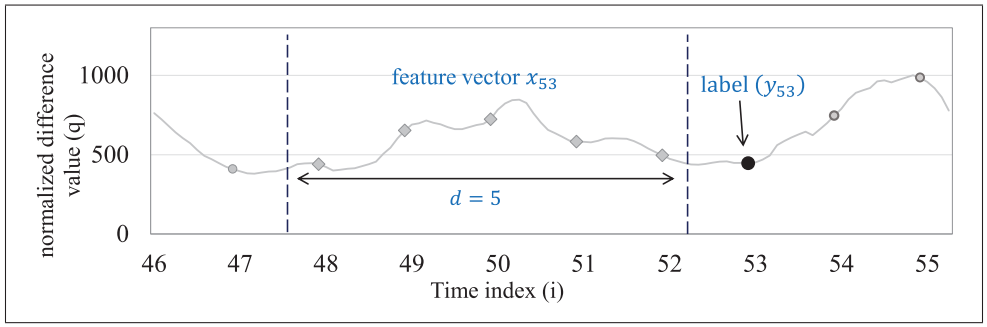


Figure 4.2 An example of creating the training sample $i = 53$ from processed time-series values (q_i). Sample $i = 53$ includes feature vector x_{53} (with length $d = 5$) and target value y_{53}

4.3.2 Training

The base learners have to be trained in such a manner that the whole ensemble achieves the optimal balance between the accuracy of base learners and the diversity of their outputs to minimize the prediction error. The ensemble training process must find the optimal balance by tuning two parameters at the same time: accuracy and diversity. In the proposed model, this optimization process is done using a divide-and-conquer approach. Each base learner finds the optimal balance between accuracy and diversity in a small region of the feature space. Hence, the feature space is segmented into smaller subspaces, and each subspace is used as the training set of a base learner. Then, each base learner is trained to provide a balance between the following targets: (i) accuracy on its corresponding segment (or training set), and (ii) diversity on the adjacent segments.

Database $\mathcal{D}$ is employed to train the ensemble model. The samples in $\mathcal{D}$ are divided (randomly) into two groups: *the training dataset*, and *the boosting dataset*. The training dataset is denoted as $\mathcal{D}^{'}$ and includes 80% of samples in $\mathcal{D}$. The training phase is done in four steps as shown in Fig. 4.1. In Step 1, the training dataset is segmented into $M^{'}$ segments using DP clustering algorithm while the value of $M^{'}$ is determined automatically in DP. In Step 2, $M^{'}$ base learners are generated and their hyperparameters are initialized by maximizing the standard Gaussian likelihood. In Step 3, the base learners are trained to provide the optimal accuracy-diversity balance. Instead of using the standard Gaussian likelihood function, we propose a likelihood function which considers both criteria (accuracy and diversity). In Step 4, $M^{''}$ base learners are generated and added to the ensemble using the boosting dataset as input. The total number of generated base learners during the four steps of the training phase is $M = M^{'} + M^{''}$. The steps are detailed in the remainder of this section.

4.3.2.1 Step 1

DP clustering algorithm (explained in Section 4.2.3) is used to partition the training dataset. DP algorithm generates $M^{'}$ clusters $\mathcal{D}_m$ ($m = 1, ..., M^{'}$) while each cluster represents a subspace

in the feature space. The number of generated clusters is determined automatically during the clustering and can be tuned by changing the innovation parameter α. The goal is to place similar samples (i.e., with similar feature vector) in the same segment. The probability that a new sample belongs to a cluster depends on the similarity of the new sample to the members of the cluster. It is calculated based on the occupation number. Since the occupation number in Equation (4.14) is not data-dependent (Rasmussen & Ghahramani, 2001) we replace it with the following kernel-based function:

$$o_{\neg i,m} = (M-1)\frac{\sum_{j\neq i} k(x_i, x_j;\theta)\delta(c_j, m)}{\sum_{j\neq i} k(x_i, x_j;\theta)}, \tag{4.19}$$

$$\delta(c_j, m) = \begin{cases} 1 & \text{if } c_j = m \\ 0 & \text{otherwise.} \end{cases} \tag{4.20}$$

The function $k(x_i, x_j;\theta)$ is an arbitrary covariance function. We used the *dot-product covariance function*:

$$k(x_i, x_j; l) = exp\left(l^{-1}\sum_{k=1}^{d}[x_i]_k \cdot [x_j]_k\right), \tag{4.21}$$

in which l is the hyperparameter of the covariance function. The element $[x_i]_k$ is the k-th component of the feature vector x_i. It has been shown that dot-product covariance function is effective in the classification problems where the feature values are in the range of $[-1, 1]$ on each dimension (Rasmussen & Williams, 2006).

The steps in DP clustering are shown in Algorithm 4.1. Function $pop(\mathcal{D}')$ in Line 1 removes one sample from $\mathcal{D}'$ and returns the sample. Function $size(\mathcal{D}')$ in Line 2 returns the number of sample in the set $\mathcal{D}'$. The occupation number is calculated using Function *occupationNumber* in Line 5 based on Equation (4.19). The probability of assigning x_i to the existing clusters is calculated in Line 7 using Function $clusteringProbability$ according to Equation (4.12). Function $newClusterProbability$ computes the probability of creating a new cluster and assigning x_i to the new cluster based on Equation (4.13). The output of Algorithm 4.1 is M' clusters of samples while M' is determined automatically.

Algorithm 4.1 Step 1 of training phase (DP clustering)

Input: $\mathcal{D}^{'}$, α
Output: $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_{M'}$

$M^{'} = 1$, $\mathcal{D}_1 \leftarrow pop(\mathcal{D}^{'})$
while $size(\mathcal{D}^{'}) > 0$ **do**
    $x_i \leftarrow pop(\mathcal{D}^{'})$
    $\hat{m} = 1$, $P_{max} = 0$
    for $m = 1$ ***to*** $M^{'}$ **do**
        $O_m = occupationNumber(m, x_i)$ Eq. (4.19)
        $P_m = clusteringProbability(O_m, \alpha)$ Eq. (4.12)
        if $P_m > P_{max}$ **then**
            $P_{max} = P_m$, $\hat{m} = m$
        end
    end
    $P_{new} = newClusterProbability(\alpha)$ Eq. (4.13)
    if $P_{max} > P_{new}$ **then**
        $\mathcal{D}_{\hat{m}} \leftarrow x_i$ Assign to an existing cluster
    else
        $M^{'} = M^{'} + 1$
        $\mathcal{D}_{M'} \leftarrow x_i$ Create a new cluster
    end
end

4.3.2.2 Step 2

Base learners f_m ($m = 1, ..., M^{'}$) are generated in this step. Cluster $\mathcal{D}_m$ is assigned as the training set of learner f_m. All the learners utilize the RQ covariance function defined in Equation (4.4) while the hyperparameters of f_m are denoted as θ_m. The hyperparameters of the learners are initialized using standard GPR training process. In standard GPR, the values of hyperparameters θ are determined by maximizing the Gaussian likelihood function $p(Y|X, \theta)$ defined in (4.8). The learners achieve the best fit on their training set. However, they cannot guarantee the accuracy-diversity balance.

4.3.2.3 Step 3

In this step, the ensemble is trained to provide the balance between accuracy and diversity. Each base learner optimizes the accuracy-diversity balance in a small region of the feature space. For each base learner, the goal is to compromise between the accuracy on its corresponding

subspace and diversity of predictions on its neighbor subspaces. This goal has two sides. On the one hand, it enhances the learner's accuracy which is independent of the samples in other segments and the results of other learners. On the other hand, it improves the diversity which depends on the outputs of other learners. To achieve this goal, a new likelihood function is introduced which considers the accuracy (on the learner's corresponding subspace) and diversity (on the learner's adjacent subspaces). The values of θ_m is selected considering the accuracy of f_m on $\mathcal{D}_m$ and diversity of the learners on the subspaces that are adjacent to $\mathcal{D}_m$.

The values of hyperparameters have been initialized in Step 2. In this step, they are evolved to maximize the proposed *GPR ensemble likelihood function* which considers accuracy and diversity as follows:

$$\mathcal{L}_m = \mathcal{P}_m + \mathcal{V}_m, \tag{4.22}$$

where $\mathcal{P}_m = log\left(p(Y_{\mathcal{D}_m}|X_{\mathcal{D}_m}, \theta_m)\right)$ is the Gaussian log-likelihood defined in Equation (4.9), and $\mathcal{V}_m$ measures the diversity:

$$\mathcal{V}_m = \frac{1}{|D_{\neg m}|} \sum_{x_i \in \mathcal{D}_{\neg m}} p_m(x_i) \cdot v_m(x_i)\,, \tag{4.23}$$

in which $\mathcal{D}_{\neg m}$ is the set of samples x_i in the clusters adjacent to $\mathcal{D}_m$ (excluding $\mathcal{D}_m$) for which the $p_m(x_i) \geq \epsilon$, and $|D_{\neg m}|$ is the number of samples in $D_{\neg m}$. Function $v_m(x_i)$ is calculated as:

$$v_m(x_i) = \frac{1}{2} \cdot \frac{\mathcal{I}_m(x_i) - \mathcal{E}_m(x_i)}{\mathcal{I}_m(x_i) + \mathcal{E}_m(x_i)} + \frac{1}{2}\,, \tag{4.24}$$

while $\mathcal{I}_m(x_i)$ is defined based on Equation (4.1) and measures the distance between estimation of expert m for x_i and $\bar{f}_{\neg m}(x_i)$:

$$\mathcal{I}_m(x_i) = \left(f_{m|\theta_m}(x_i) - \bar{f}_{\neg m}(x_i)\right)^2, \tag{4.25}$$

where $\bar{f}_{\neg m}(x_i)$ (the average of predicted values of other learners on x_i) is defined based on Equation (4.2):

$$\bar{f}_{\neg m}(x) = \frac{1}{M-1} \sum_{\substack{j=1 \\ j \neq m}}^{M} p_j(x) f_j(x) \,. \tag{4.26}$$

Also, $\mathcal{E}_m(x_i)$ is the estimation error of expert m on x_i:

$$\mathcal{E}_m(x_i) = \left(f_{m|\theta_m}(x_i) - y_i \right)^2 . \tag{4.27}$$

$p_m(x_i)$ is the probability of x_i belongs to $\mathcal{D}_m$ based on Equations (4.12) and (4.19). In other words, $p_m(x_i)$ measures the similarity of between x_i and $\mathcal{D}_m$ and is defined as:

$$p_m(x_i) = \frac{p(c_i = m | c_{\neg i}, \alpha)}{\sum_{j=1}^{M} p(c_i = j | c_{\neg i}, \alpha)} \,. \tag{4.28}$$

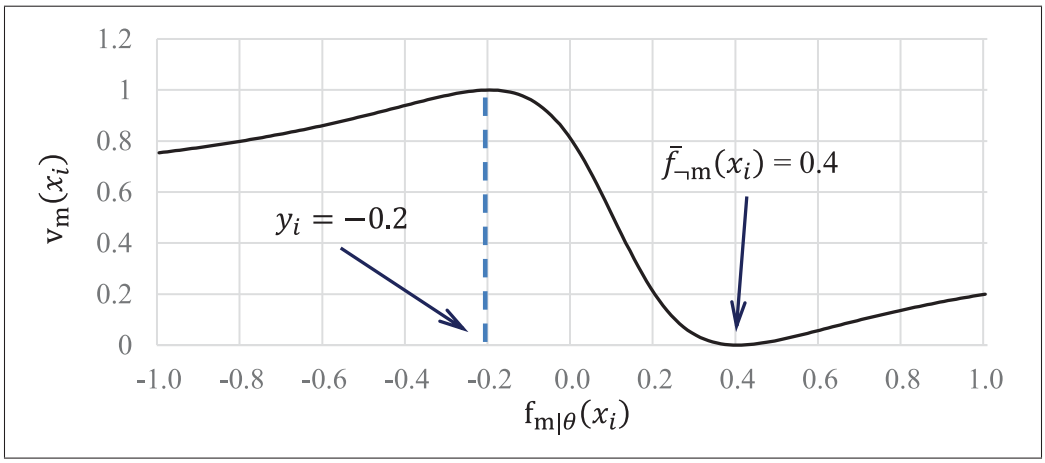


Figure 4.3 An example for function $v_m(x_i)$ for $x_i \notin D_m$ where $y_i = -0.2$ and $\bar{f}_{\neg m}(x_i) = 0.4$. The minimum value of $v_m(x_i)$ happens when $f_{m|\theta}(x_i) = \bar{f}_{\neg m}(x_i)$, and its maximum value occurs when $f_{m|\theta}(x_i) = y_i$

Function $p_m(x_i)$ can be assumed as a similarity score between x_i and samples in D_m. When $p_m(x_i)$ is less than a threshold (which means x_i is not similar to cluster m), x_i is excluded from the calculation of $\mathcal{V}_m$. This allows learner m to focus on its local neighbourhood instead of the

whole training dataset. The value of threshold is determined based on the minimum value of $p(x)$ for x in cluster m.

The likelihood function $\mathcal{L}_m$ is the sum of two terms. The first term ($\mathcal{P}_m$) is calculated using the samples of subspace m (i.e., $\mathcal{D}_m$) and shows the accuracy of f_m on $\mathcal{D}_m$. The second term ($\mathcal{V}_m$) is computed using the samples of surrounding subspaces ($\mathcal{D}_{m'}, m^{'} \neq m$). $\mathcal{V}_m$ measures the contribution of $f_m|\theta_m$ to the prediction diversity on other subspaces.

Function $v_m(x_i)$ is in the range of $[0, 1]$. It is minimized when $\mathcal{I}_m(x_i) = 0$, and maximized when $\mathcal{E}_m(x_i) = 0$. The outcome $\mathcal{E}_m(x_i) = 0$ occurs when there is no difference between the prediction of learner m for x_i and target value y_i (i.e., $f_{m|\theta}(x_i) = y_i$), which means the prediction error of learner m for x_i is zero. The term $\mathcal{I}_m(x_i) = 0$ happens when the prediction of learner m for x_i is equal to the average prediction of other learners (i.e., $f_{m|\theta}(x_i) = \bar{f}_{\neg m}(x_i)$). This is the worst case for the diversity of the learner m when it generates the same outcome as other learners.

We can analyze $v_m(x_i)$ using the following example. Consider x_i as a sample in the neighbourhood of cluster m (but it is not in cluster m) with target value $y_i = -0.2$. The average prediction of other learners for x_i is $\bar{f}_{\neg m}(x_i) = 0.4$. Based on this assumptions, function $v_m(x_i)$ is calculated as:

$$v_m(x_i) = \frac{1}{2}\frac{\Big(f_{m|\theta}(x_i) - 0.4\Big)^2 - \Big(f_{m|\theta}(x_i) + 0.2\Big)^2}{\Big(f_{m|\theta}(x_i) - 0.4\Big)^2 + \Big(f_{m|\theta}(x_i) + 0.2\Big)^2} + \frac{1}{2}. \tag{4.29}$$

which is illustrated in Figure 4.3. As shown, v_m has a minimum value at $f_{m|\theta}(x_i) = \bar{f}_{\neg m}(x_i)$, and it has a maximum value at $f_{m|\theta}(x_i) = y_i$.

Probability $p_m(x_i)$ limits the calculation of $\mathcal{V}_m$ to the subspaces that are adjacent to $\mathcal{D}_m$. When the distance between x_i and $\mathcal{D}_m$ is high (i.e., x_i is not similar to samples in $\mathcal{D}_m$ which means x_i is not in the surrounding subspaces), $f_{m|\theta_m}(x_i)$ does not affect $\mathcal{V}_m$. Since $p_m(x_i)$ takes small values (close to zero) for samples that are far from $\mathcal{D}_m$, it limits the number of samples that affect the likelihood function of f_m. Since f_m optimizes the accuracy-diversity trade-off in its local neighborhood, the complexity of objective function $\mathcal{V}_m$ (and $\mathcal{L}$) is decreased significantly

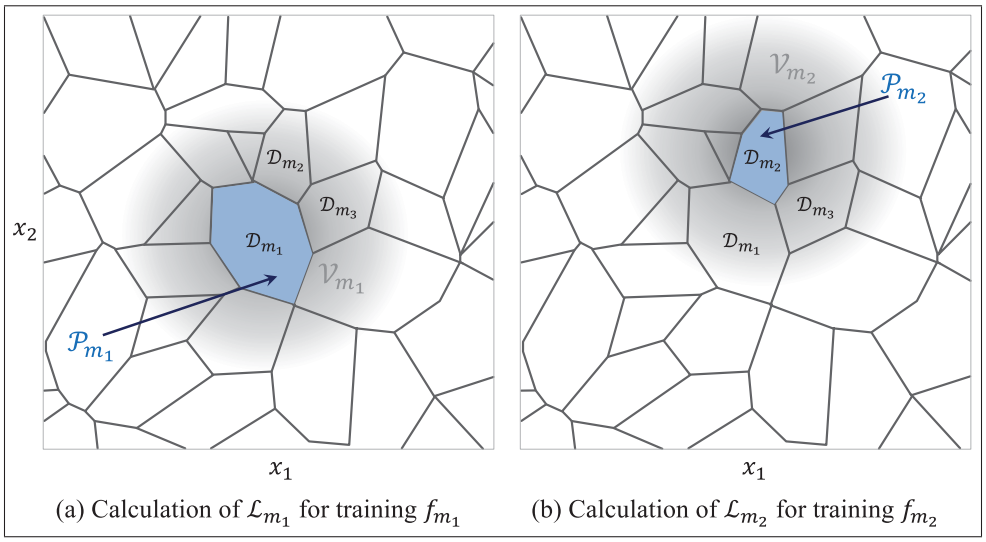


Figure 4.4 The process of calculating the likelihood function. Likelihood function $\mathcal{L}_m$ is sum of $\mathcal{P}_m$ and $\mathcal{V}_m$ while $\mathcal{P}_m$ is calculated using the samples in $\mathcal{D}_m$, and $\mathcal{V}_m$ is computed using the prediction results of other learners on the adjacent subspaces $\mathcal{D}_{\neg m}$ (the grey area)

(compared to an objective function which considers the whole dataset). Also, the complexity of training process for f_m does not depend on the size of the dataset $\mathcal{D}$. Fig. 4.4 shows this property of our ensemble training. During the training of f_m, $\mathcal{P}_m$ is calculated using the samples in the subspace $\mathcal{D}_m$, and $\mathcal{V}_m$ is calculated based on samples in adjacent subspaces $\mathcal{D}_{\neg m}$. The samples that are not in the adjacent subspaces and the predictions of experts for those samples do not affect $\mathcal{L}_m$.

Fig. 4.5 gives an intuitive illustration of the accuracy-diversity balance. The samples of one cluster ($\mathcal{D}_m$) are given in a one-dimensional feature space x. The goal is to train f_m (and find the optimal θ_m) in a manner that it provides an optimal accuracy-diversity balance in the regions around $\mathcal{D}_m$. In this example, (x_i, y_i) is a data point in a subspace adjacent to $\mathcal{D}_m$ ($x_i \notin \mathcal{D}_m$). GPR expert f_m has to diminish the prediction error of other learners on x_i by providing diversity. There are three learners ($f_{m_1'}$, $f_{m_2'}$, and $f_{m_3'}$) that have been trained on their subspaces ($\mathcal{D}_{m_1'}$, $\mathcal{D}_{m_2'}$, and $\mathcal{D}_{m_3'}$). So, the values of their hyperparameters and their predicted values for x_i are known. Consider two possible sets of hyperparameters $\bar{\theta}$ and $\hat{\theta}$ for f_m where the outcome of standard Gaussian likelihood function on $\mathcal{D}_m$ is almost the same and maximized for both sets (e.g., they

are two local optimums of Equation (4.8), and $p(Y_{\mathcal{D}_m}|X_{\mathcal{D}_m}, \bar{\theta}) \simeq p(Y_{\mathcal{D}_m}|X_{\mathcal{D}_m}, \hat{\theta})$). It means they result in the same level of accuracy for f_m, so both $\bar{\theta}$ and $\hat{\theta}$ have equal chances to be selected in the standard GPR training. However, $\hat{\theta}$ is more likely to be selected when we employ our GPR ensemble likelihood function. Consider the prediction results of f_m, $f_{m'_1}$, $f_{m'_2}$, and $f_{m'_3}$ for x_i. It can be shown $f_{m|\hat{\theta}}(x_i)$ leads to higher diversity compared to $f_{m|\bar{\theta}}(x_i)$ according to Equation (4.1). So, $\hat{\theta}$ is preferred because it provides a higher diversity on x_i. In this example, we considered only the outcome of three learners on sample x_i. In a real scenario, we need to consider all the training samples and base learners in the neighbor segments.

The accuracy-diversity balance is improved after one round of optimizing the experts' hyper-parameters within Step 3. It is possible to enhance the balance by multiple iteration of Step 3.

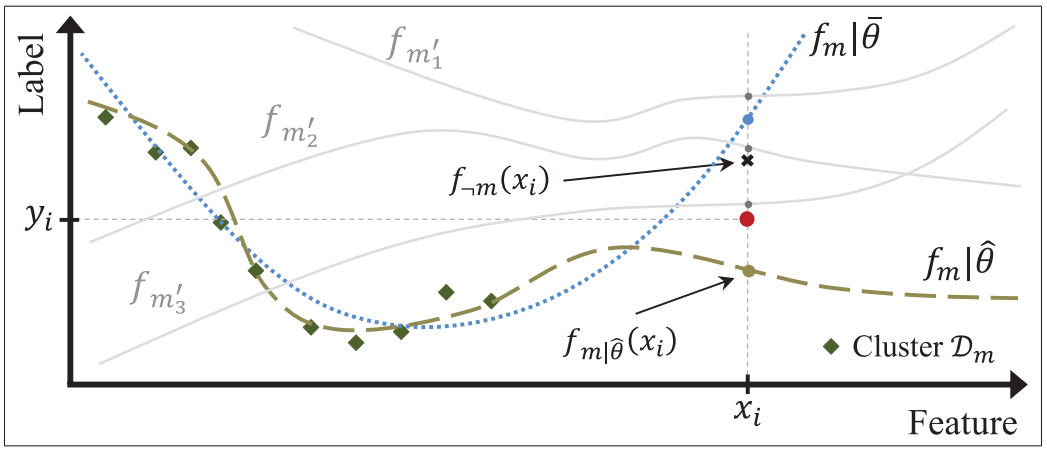


Figure 4.5 Example of hyperparameter optimization for function f_m. The prediction results of other learners (e.g., $f_{m'_1}$, $f_{m'_2}$, $f_{m'_3}$) on the surrounding samples (e.g., x_i) must be considered in the hyperparameter optimization of learner f_m

4.3.2.4 Step 4

In this step, the samples in the boosting dataset are used to enhance the ensemble performance. First, the generated (and trained) learners are employed to predict the samples in the boosting dataset. The prediction algorithm is explained in Section 4.3.3. The goal is to select the samples

in the boosting dataset that their corresponding prediction error are significant. In other words, the samples that cannot be predicted accurately by the existing learners are selected to form a subset called $\mathcal{D}''$. The prediction error is measured using NMSE (Bayati et al., 2016):

$$NMSE = \frac{1}{\sigma^2 \, |\mathcal{D}''|} \sum_{(x_i, y_i) \in \mathcal{D}''} (y_i - \hat{f}_{ens}(x_i))^2, \tag{4.30}$$

where (x_i, y_i) is a sample in $\mathcal{D}''$, $|\mathcal{D}''|$ is the number of samples in $\mathcal{D}''$, σ^2 is the variance of $\{y_i\}$, and $\hat{f}_{ens}(x_i)$ is the voted prediction for x_i (defined in 4.3.3). The smaller values of NMSE are preferred, and $NMSE = 0$ corresponds to a perfect predictor with no error. $(x_i, y_i) \in \mathcal{D}''$ is added to the boosting dataset if its corresponding prediction error is greater than threshold e_{thr} (i.e., $\frac{1}{\sigma^2}(y_i - \hat{f}_{ens}(x_i))^2 \geq e_{thr}$). The value of e_{thr} is set to 50 percentile of the NMSE of the boosting dataset. Therefore, 50% of the boosting samples (with significant prediction error) will be selected.

The prediction error of generated experts is high for the samples in $\mathcal{D}''$. It will be used to generate and train M'' new learners which will be added to the ensemble to improve the overall prediction accuracy. The new learners are generated as explained in Step 1, Step 2, and Step 3. First, the $\mathcal{D}''$ is clustered using DP algorithm to create M'' clusters (while the value of M'' is determined in DP). Then, M'' learners are generated and initialized using the new clusters. Finally, the learners are trained to optimize the accuracy-diversity by maximizing the likelihood function.

Algorithm 4.2 implements steps 2, 3, and 4. In Step 2, the hyperparameters of GPR experts are initialized using function $fitStandardGPR$ (line 2). It takes the samples in one subspace and optimizes the standard Gaussian likelihood in Equation (4.8). In Step 3, the accuracy-diversity balance is optimized using function $fitAccuracyDiversity$ (line 4). In Step 4, the ensemble model is used to predict the samples in boosting dataset and create $\mathcal{D}''$ (lines 5 to 11). Then, M'' clusters are created by applying DP clustering to $\mathcal{D}''$ (line 12). Each cluster is employed to initialize and train a new GPR expert (lines 13 to 16). Note the difference between inputs of $fitAccuracyDiversity$ in lines 4 and 16. In line 4, the input consists of training dataset $\mathcal{D}'$. In

Algorithm 4.2 Steps 2, 3 and 4 of training phase

1 **Input:** $\mathcal{D}_1, ..., \mathcal{D}_{M'}$, *boosting data*
2 **Output:** $f_1, f_2, ..., f_M$

3 **for** $m = 1$ *to* $M^{'}$ **do**
4 | $f_m = fitStandardGPR(\mathcal{D}_m)$ **Step 2**: Maximize Eq. (4.8)
5 **end**
6 **for** $m = 1$ *to* $M^{'}$ **do**
7 | $f_m = fitAccuracyDiversity(f_m, \mathcal{D}^{'})$ **Step 3**: Maximize Eq. (4.22)
8 **end**
9 **for** $x_j \in$ *boosting* **do**
10 | $e_j = NMSE(\hat{f}_{ens}(x_j),\ y_j)$ **Step 4**: Prediction error in Eq. (4.30)
11 **end**
12 $threshold = percentile(e, 70)$
13 $\mathcal{D}^{''} = \{\}$
14 **for** $x_j \in$ *boosting* **do**
15 | **if** $e_j \geq threshold$ **then**
16 | | $\mathcal{D}^{''} \leftarrow x_j$
17 | **end**
18 **end**
19 $\{\mathcal{D}^{''}_1, ..., \mathcal{D}^{''}_{M''}\} \leftarrow DP(\mathcal{D}^{''})$ DP clustering for $\mathcal{D}^{''}$
20 **for** $m = 1$ *to* $M^{''}$ **do**
21 | $f_{m+M'} = fitStandardGPR(\mathcal{D}^{''}_m)$
22 **end**
23 **for** $m = 1$ *to* $M^{''}$ **do**
24 | $f_{m+M'} = fitAccuracyDiversity(f_{m+M'}, \mathcal{D}^{'} \cup \mathcal{D}^{''})$
25 **end**

line 16, both $\mathcal{D}^{'}$ and $\mathcal{D}^{''}$ are required for the accuracy-diversity balance because the new experts must consider the results of $M^{'}$ previously generated experts on the whole dataset.

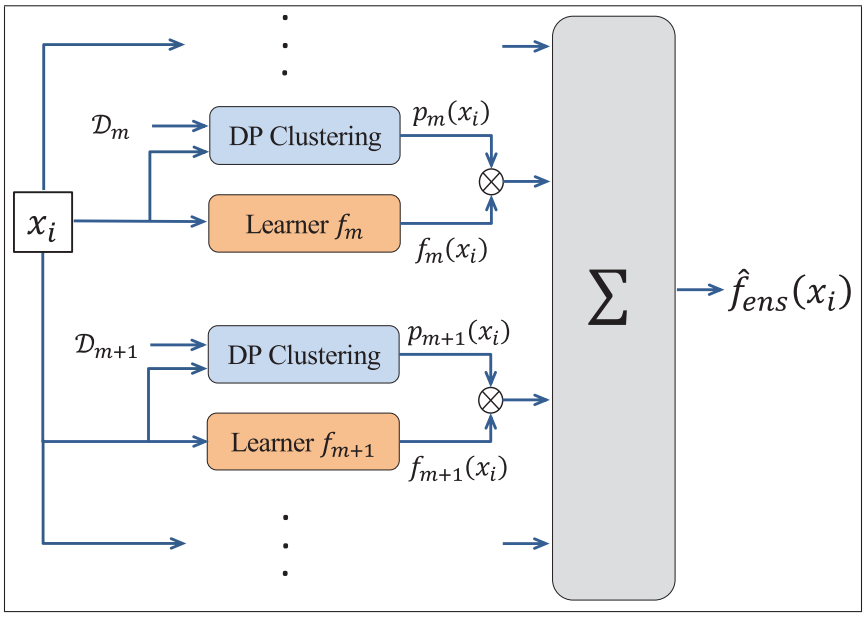


Figure 4.6 The final predicted value is the weighted sum of the individual prediction. The weight of each learner is proportional to the similarity of x_i to the corresponding cluster of the learner

4.3.3 Prediction

The final voted prediction for a new sample x_i is the weighted sum of predictions of the GPR components:

$$\hat{f}_{ens}(x_i) = \sum_{m=1}^{M} p_m(x_i) f_m(x_i), \tag{4.31}$$

where M is the number of generated experts, and $p_m(x_i)$ is defined in Equation (4.28). The contribution of expert m in the final voted predicted value, $\hat{f}_{ens}(x_i)$, is proportional to $p_m(x_i)$ (i.e., the similarity of x_i to the the samples in the cluster m). The term $p_m(x_i)$ gives strong weights to the experts that have been trained on samples similar to x_i. It reduces the effects of learners that have been trained on clusters which are far from x_i in the feature space. The ensemble prediction is shown in Figure 4.6. As shown, the outcome of each learner is multiplied by the probability of sample belongs to the training set of the learner which is calculated using DP clustering algorithm.

4.3.4 Computational Complexity

In standard GPR, the inverse of covariance matrix K is required to optimize the likelihood function in Equation (4.8). The time complexity of matrix inversion is $O(N^3)$ where N is the number of training samples. Thus, the standard GPR has a cubic computational requirement.

In our algorithm, two parts must be considered for time complexity analysis: DP clustering (in Step 1 of training), and hyperparameter optimization (in Step 2 and Step 3). In DP clustering, a covariance matrix of size $N \times N$ is required for calculation of the occupation number in Equation (4.19). Each element of the covariance matrix is the result of Equation (4.21). The time complexity for calculation of such a matrix in DP clustering is in the order of $O(N^2)$.

In steps 2, 3 and 4, the hyperparameters of M learners must be optimized. For expert m, the inverse of covariance matrix of $\mathcal{D}_m$ is needed. The size of $\mathcal{D}_m$ is not predetermined and depends on the DP clustering, training set, and length of feature vector. Assuming $|\mathcal{D}_m| \sim \frac{N}{M}$ (on average), the time complexity of hyperparameter optimization in our algorithm is $O(\frac{N^3}{M^3})$. For small values of M, the time complexity of algorithm is close to the standard GPR. This can be avoided by choosing appropriate values for α. As the number of clusters increases, the average size of clusters decreases. In Section 4.4, the number and size of clusters for different values of α are investigated. It shows that for a wide range of values for α, the time complexity of GPR ensemble model is significantly smaller than standard GPR.

4.3.5 Gradient Descent Optimization

The proposed likelihood function is optimized using the gradient descent method, which requires the likelihood function derivative. In this section, we provide the derivative of the likelihood function in Equation (4.22) with respect to hyperparameters θ:

$$\frac{\partial \mathcal{L}_m}{\partial \theta_m} = \frac{\partial \mathcal{P}_m}{\partial \theta_m} + \frac{\partial \mathcal{V}_m}{\partial \theta_m}. \tag{4.32}$$

The first part (i.e., $\frac{\partial \mathcal{P}_m}{\partial \theta_m}$) is the derivative of the standard GPR log-likelihood function, and has been investigated in existing GPR models (Bayati et al., 2016):

$$\frac{\partial \mathcal{P}_m}{\partial \theta_m} = \frac{1}{2} Y^\top K^{-1} \frac{\partial K}{\partial \theta_m} K^{-1} Y - \frac{1}{2} Tr\left(K^{-1} \frac{\partial K}{\partial \theta_m}\right), \tag{4.33}$$

where Tr is the trace of the matrix, and $\frac{\partial K}{\partial \theta_m}$ is the covariance matrix derivative:

$$\left[\frac{\partial K}{\partial \theta_m}\right]_{l,k} = \frac{\partial k(x_l, x_j, \theta_m)}{\theta_m}. \tag{4.34}$$

The second part of the derivative (i.e., $\frac{\partial \mathcal{V}_m}{\partial \theta_m}$) depends on $f_{m|\theta_m}$, $\mathcal{I}_m$, and $\mathcal{E}_m$. Function $f_{m|\theta_m}(x_*)$ is the prediction of learner m for input x_*, and it is defined in Equation (4.6):

$$f_{m|\theta_m}(x_*) = \hat{f}_* = K_*^\top (K + \sigma^2 I)^{-1} Y. \tag{4.35}$$

Its gradient is calculated as:

$$\begin{aligned} \frac{\partial f_{m|\theta_m}(x_*)}{\partial \theta_m} &= \frac{\partial K_*^\top}{\partial \theta_m}(K + \sigma^2 I)^{-1} Y + \\ & \quad K_*^\top (K + \sigma^2 I)^{-1} \frac{\partial K^\top}{\partial \theta_m}(K + \sigma^2 I)^{-1} Y. \end{aligned} \tag{4.36}$$

Accordingly, the derivative of $\mathcal{I}_m(x_i)$ and $\mathcal{E}_m(x_i)$ are:

$$\begin{aligned} \frac{\partial \mathcal{I}_m(x_i)}{\theta_m} &= \frac{\partial \left(f_{m|\theta_m}(x_i) - \bar{f}_{\neg m}(x_i)\right)^2}{\theta_m} \\ &= 2\left(f_{m|\theta_m}(x_i) - \bar{f}_{\neg m}(x_i)\right) \frac{\partial f_{m|\theta_m}}{\partial \theta_m}, \end{aligned} \tag{4.37}$$

$$\begin{aligned} \frac{\partial \mathcal{E}_m(x_i)}{\theta_m} &= \frac{\partial \left(f_{m|\theta_m}(x_i) - y_i\right)^2}{\theta_m} \\ &= 2\left(f_{m|\theta_m}(x_i) - y_i\right) \frac{\partial f_{m|\theta_m}}{\partial \theta_m}, \end{aligned} \tag{4.38}$$

where $\frac{\partial f_{m|\theta_m}}{\partial \theta_m} = \frac{\partial f_{m|\theta_m}(x_i)}{\partial \theta_m}$. The derivative of $\frac{\partial v_m(x_i)}{\theta_m}$ is:

$$\begin{aligned}\frac{\partial v_m(x_i)}{\theta_m} &= 2\,\frac{\mathcal{E}_m(x_i)\left(f_{m|\theta_m}(x_i) - \bar{f}_{\neg m}(x_i)\right)}{\left(\mathcal{I}_m(x_i) + \mathcal{E}_m(x_i)\right)^2}\frac{\partial f_{m|\theta_m}}{\partial \theta_m} \\ &- 2\,\frac{\mathcal{I}_m(x_i)\left(f_{m|\theta_m}(x_i) - y_i\right)}{\left(\mathcal{I}_m(x_i) + \mathcal{E}_m(x_i)\right)^2}\frac{\partial f_{m|\theta_m}}{\partial \theta_m}.\end{aligned} \tag{4.39}$$

Finally, we can calculate the derivative of $\frac{\partial \mathcal{V}_m}{\theta_m}$:

$$\frac{\partial \mathcal{V}_m}{\theta_m} = \frac{1}{|D_{\neg m}|}\sum_{x_i \in \mathcal{D}_{\neg m}} p_m(x_i)\frac{\partial v_m(x_i)}{\theta_m}. \tag{4.40}$$

In calculating the derivative, those terms that do not depend on θ are considered constant (e.g., $p_m(x_i)$, $\mathcal{D}_{\neg m}$, $\bar{f}_{\neg m}(x_i)$, and Y).

Table 4.2 Datasets of traffic time-series created from different sources at different time-scale

Dataset	**Source of data**	**Time-scale (minute)**
CAIDA-01	CAIDA Anonymized Internet Traces	0.5
CAIDA-02	CAIDA Anonymized Internet Traces	1
Abilene-01	Internet2 Abilene Network	5
Abilene-02	Internet2 Abilene Network	10
Waikato-01	Waikato VIII	15
Waikato-02	Waikato VIII	30

4.4 Experimental Results

4.4.1 Setup

We carried out experiments on six different datasets which have been created using traffic samples from three well-known sources of traffic data. The list of datasets and data sources are shown in Table 4.2. The first two datasets (CAIDA-01, and CAIDA-02) have been created from the traffic data monitored on 10GigE links from 2008 to 2015 provided by the Center for Applied Internet Data Analysis (CAIDA) (*The CAIDA UCSD Anonymized Internet Traces*, 2008-2015). Abilene Internet2 Network traffic data (*Internet2 Abilene Network*, 2007) (from 2007-01-01 to 2007-10-14) has been used to build the datasets Abilene-01 and Abilene-02. Abilene Internet2 Network is a high-performance backbone network for research and education institutes in the United States. Waikato traffic traces (*Waikato VIII 2011*, 2011) collected from 2011 to 2013 have been exploited to form datasets Waikato-01 and Waikato-02. The time-scales of the traffic samples in the datasets varies from 30 seconds (in CAIDA-01) to 30 minutes (in Waikato-02) as determined in Table 4.2. These datasets are created based on the traffic bit-rate time-series (in Mbps). Abilene datasets consist of traffic bit-rates between pairs of points (i.e., traffic matrix), but Waikato and CAIDA datasets include traffic loads on the links in the network (i.e., link utilization). The process of preparing the dataset is the same for all these datasets, as explained in Section 4.3.1. The main difference between these datasets is the time-scale of traffic samples, as illustrated in Table 4.2. Therefore, the prediction task for these datasets is to estimate target values (i.e., y_i) according to the feature vector (i.e., x_i).

We compared our model with different algorithm including traditional time-series algorithms (i.e., ARIMA (Adas, 1997), FARIMA (N. Zhang et al., 2017)), supervised regression methods (i.e., standard GPR (Bayati et al., 2016), SVR (J. Liu et al., 2015), LASSO (Choudhury et al., 2018)), ensemble learning methods (i.e., GTB, RF, ERT), and a deep learning time-series predictor (i.e., LSTM (Xingjian et al., 2015)). We performed separate experiments on the datasets in Table 4.2. In the experiments, each dataset is divided into two non-overlapped subsets. The first subset is used for the model selection (i.e., selecting the optimal size of the training set,

the optimal number of features d, and the optimal values for parameters) for each algorithm using a cross-validation process. The second subset is divided into 100 portions, and each portion is employed to create a pair of the non-overlapped train and test sets (random train-test split). For each pair, the models are fitted on the train set (using the training process explained in Section 4.3.2) and then, evaluated on the test set. The reported results are the average of 100 prediction error measurements which have been achieved from this process. The prediction error is evaluated using NMSE in Equation (4.30).

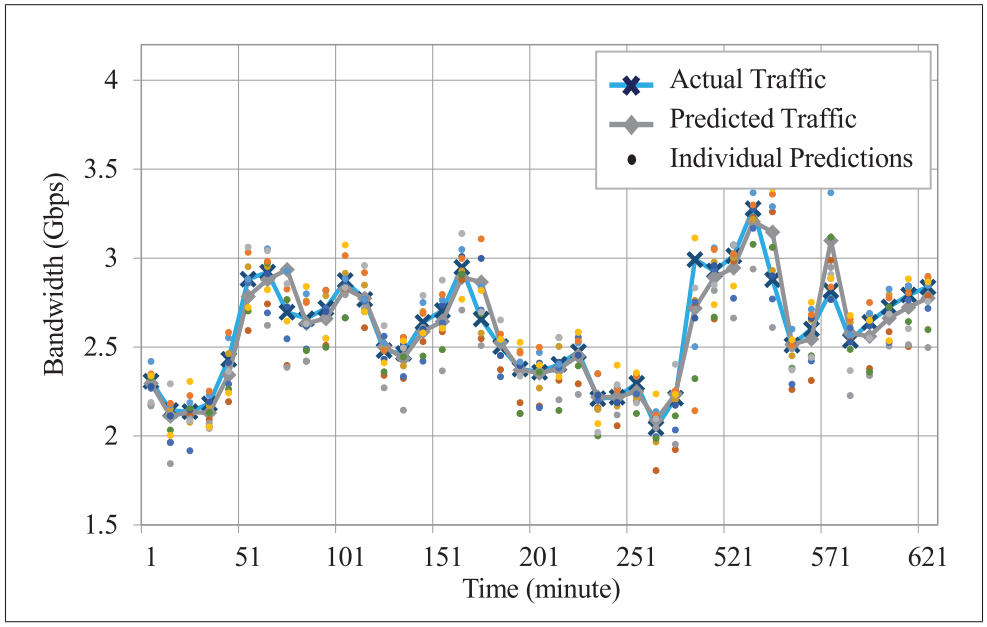


Figure 4.7 Prediction results of the proposed ensemble model (including the individual prediction results and the final predicted values) for traffic time-series at time-scale of 10 minutes (Abilene-02)

4.4.2 Results

In this section, we present the results of our experiments carried out on the datasets. In Fig. 4.7, an example of the outcome of the GPR ensemble model for predicting the traffic samples in Abilene-02 is presented. Each point is the outcome of one GPR expert. Also, the final predicted values (weighted sum of individual predictions) are illustrated. The individual experts have different errors (diversity), so the error of each expert is corrected by other experts. Thus, the

Table 4.3 Average of prediction error of different models on traffic datasets

	ARIMA	FARIMA	GPR	SVR	LASSO	GTB	RF	ERT	LSTM	GPR Ens.
CAIDA-01	**0.31**	0.37	0.43	0.49	0.51	0.43	0.41	0.39	0.38	0.33
CAIDA-02	0.33	0.31	0.39	0.47	0.47	0.41	0.39	0.35	0.33	**0.30**
Abilene-01	0.30	0.32	0.33	0.44	0.42	0.35	0.31	0.30	0.26	**0.19**
Abilene-02	0.32	0.35	0.30	0.33	0.48	0.35	0.30	0.26	0.24	**0.21**
Waikato-01	0.28	0.28	0.31	0.25	0.22	0.26	0.18	0.30	**0.16**	0.18
Waikato-02	0.32	0.31	0.29	0.34	0.35	0.31	0.26	0.25	0.15	**0.14**

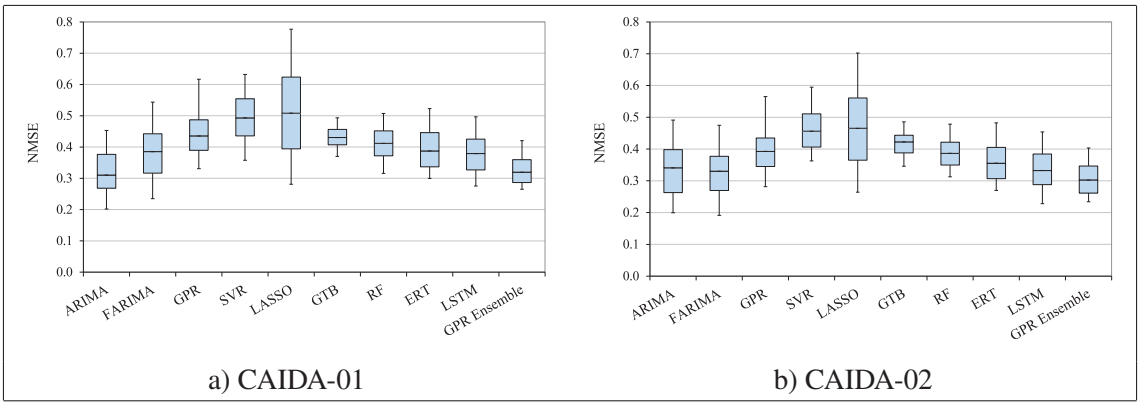


Figure 4.8 Prediction results for CAIDA datasets: (a) CAIDA-01 at time-scale of 30 seconds, (b) CAIDA-02 at time-scale of 1 minute

GPR ensemble model was able to predict most of the samples presented in Fig. 4.7 accurately. For few samples such as $t = 581$, the individual errors are not diverse around the actual value. Therefore, the final prediction error is high compared to other samples. It shows that diversity of individual predictions has an important role in reducing the final prediction error in ensemble model.

The average prediction error of different algorithms are illustrated in Table 4.3. As shown, the GPR ensemble model outperforms other models in the experiments on CAIDA-02, Abilene-01, Abilene-02, and Waikato-02. In two cases (CAIDA-01 and Waikato-1) ARIMA and LSTM have the smallest prediction error respectively. Nevertheless, GPR ensemble error is close

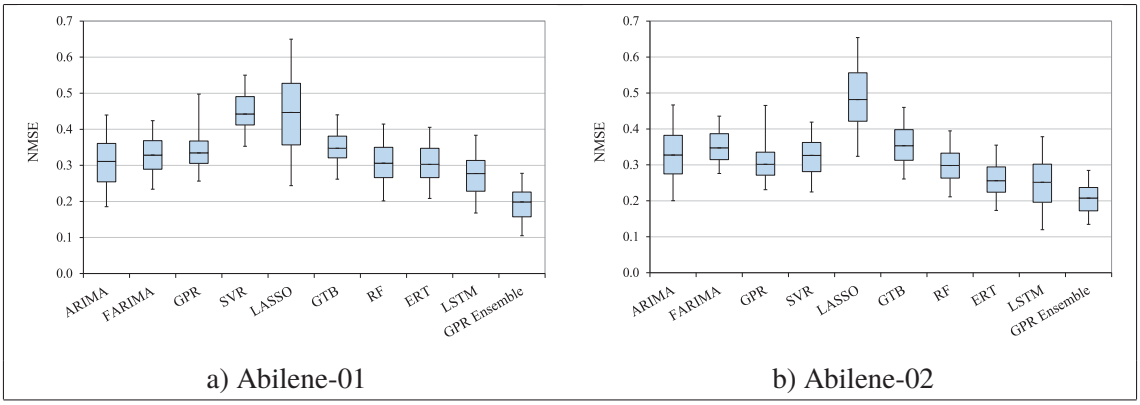


Figure 4.9 Prediction results for Abilene datasets: (a) Abilene-01 at time-scale of 5 minutes, (b) Abilene-02 at time-scale of 10 minutes

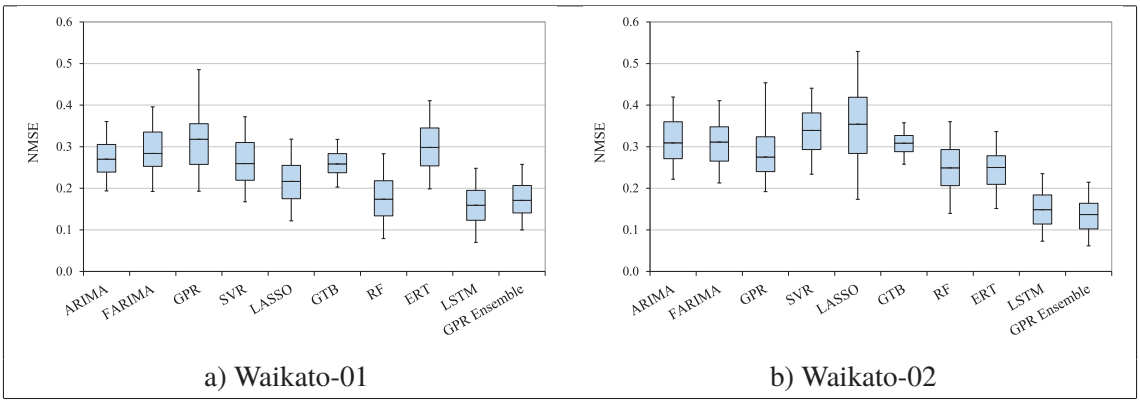


Figure 4.10 Prediction results for Waikato datasets: (a) Waikato-01 at time-scale of 15 minutes, (b) Waikato-02 at time-scale of 30 minutes

to the winner algorithms in these two cases. The details of results are illustrated in Fig. 4.8, 4.9, and 4.10. The prediction error of different models for datasets CAIDA-01 and CAIDA-02 are shown in Fig. 4.8. On average, ARIMA has the lowest prediction error for CAIDA-01 compared to other algorithms. GPR ensemble model has the second smallest prediction error. Also, its variance is less than ARIMA. For dataset CAIDA-02, GPR ensemble model has the best prediction results (in term of average prediction error). The performance of FARIMA is close to GPR ensemble model. In both cases, the GPR ensemble model has the minimum variance in prediction which shows it is more stable compared to other models. The average of

prediction error of the algorithms for CAIDA datasets is higher than other datasets (see Table 4.3). It is because of the traffic behaviour at small time-scales. The time-scale of samples in CAIDA-01 and CAIDA-02 are 30 seconds and 1 minute respectively. It has been shown that traffic exhibit short-range dependency (SRD) at small time-scales (Willinger et al., 2003). The traffic fluctuations are notable and temporal changes in traffic patterns are more frequent at small time-scales. Therefore, the average of the errors on these datasets is higher for all the algorithms.

The prediction results for datasets Abilene-01, and Abilene-02 are presented in Fig. 4.9. The time-scales of traffic samples are 5 and 10 minutes for these datasets respectively. In both cases, the GPR ensemble model outperforms other models with average NMSE of 0.19 for Abilene-01, and 0.20 for Abilene-02. Also, the variance of its prediction results is lower than other models. The second smallest prediction error belongs to LSTM with average NMSE of 0.26 and 0.24 for these datasets. Generally, the average of prediction error of all algorithms is smaller for Abilene datasets compared to the results for CAIDA datasets. The results of predictions for Waikato datasets are demonstrated in Fig. 4.10. In the case of Waikato-01, the average NMSE for LSTM is 0.16 (which is less than other competitors), and the average prediction errors of GPR ensemble model and RF are equal to 0.18. For Waikato-02, the GPR ensemble model has the best prediction result with average NMSE equal to 0.14.

The results of our experiments show the GPR ensemble model has a higher prediction accuracy compared to other ensemble models for many traffic datasets. Also, our model experiences a small variance in its prediction results which shows it is able to handle traffic characteristics and varying patterns. Unlike other models which have different results at various traffic time-scales, GPR ensemble model has a satisfactory and stable performance at all the traffic time-scales. The high prediction accuracy in GPR ensemble model is achieved by optimizing the balance between accuracy and diversity between prediction results of individual experts during the training phase.

4.4.3 Likelihood optimization

We monitored and analyzed the results for likelihood function convergence during the hyper-parameter optimization. An example learner is selected, and its likelihood values (for $\mathcal{P}_m$ and $\mathcal{V}_m$) are tracked. This learner is assigned to a cluster with 166 samples during the training process, and it has 237 samples in its neighbourhood (i.e., samples that $p_m(x)$ is greater than the threshold for them).

The changes in $\mathcal{P}_m$ and $\mathcal{V}_m$ as two terms of the likelihood function are shown in Figure 4.12. As shown, the value of $\mathcal{V}_m$ decreased during the first iterations (before iteration 41), while there is a significant improvement for $\mathcal{P}_m$. In the middle of optimization, there is a period that both terms are increased together (between iterations 41 to 51). After iteration 51, it seems $\mathcal{P}_m$ reaches its optimum point and stays stable, while there is a notable gain for $\mathcal{V}_m$. We noted the increase in $\mathcal{V}_m$ (in the final iterations) causes a minor decrease in $\mathcal{P}_m$, but the total likelihood is improving. In general, we observed similar behaviour in other learners. In the first optimization iterations, the hyperparameters are changed to improve $\mathcal{P}_m$ (i.e., improve accuracy on the assigned cluster), and in the final iterations, the hyperparameters are tuned to gain $\mathcal{V}_m$ (i.e., to improve diversity on adjacent samples).

4.4.4 Execution Time

We compare the execution time of different algorithms. As mentioned in Section 4.3.4, the computational complexity of the proposed algorithm can be affected by the number and size of the clusters. In the worst case, the time complexity of the GPR ensemble model is comparable to the standard GPR model if a cluster's size is close to N (size of the training set). On the other hand, the size and number of clusters depend on the innovation parameter α in DP clustering. Therefore, we need to investigate the size and number of clusters for different values of α. Figure 4.11 shows the average size and number of clusters created by applying the DP algorithm on dataset of size $N = 10000$. DP clustering algorithm has been executed 100 times for each dataset in Table 4.2 while 10000 samples were selected randomly from the dataset in each run. As

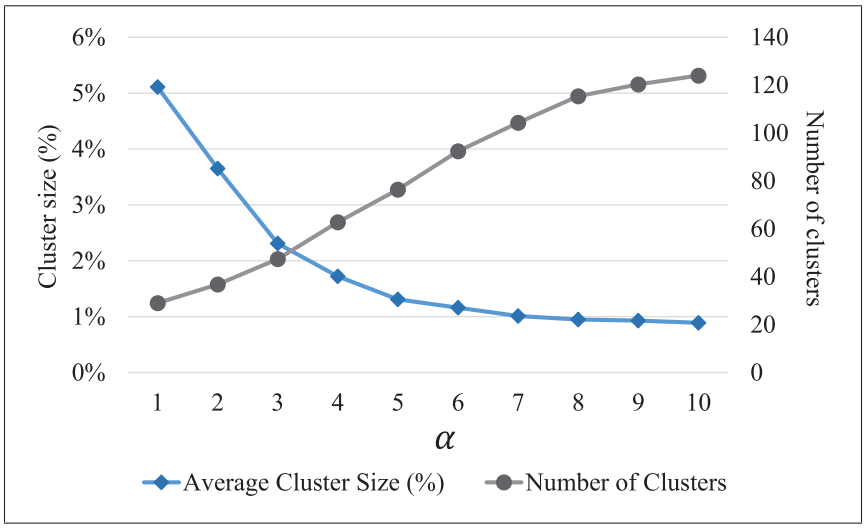


Figure 4.11 Average cluster size and number of clusters in DP clustering algorithm for different values of α when training set of size $N = 10000$

shown, the number of clusters is small when the value of α is close to zero, and it increases as α increases. At the same time, the average cluster size decreases as α increases. Even for small values of α, the cluster size is much smaller than the whole training set. For example, the average cluster size is around 5% of the training set (i.e., ~ 500 samples) for $\alpha = 0.5$.

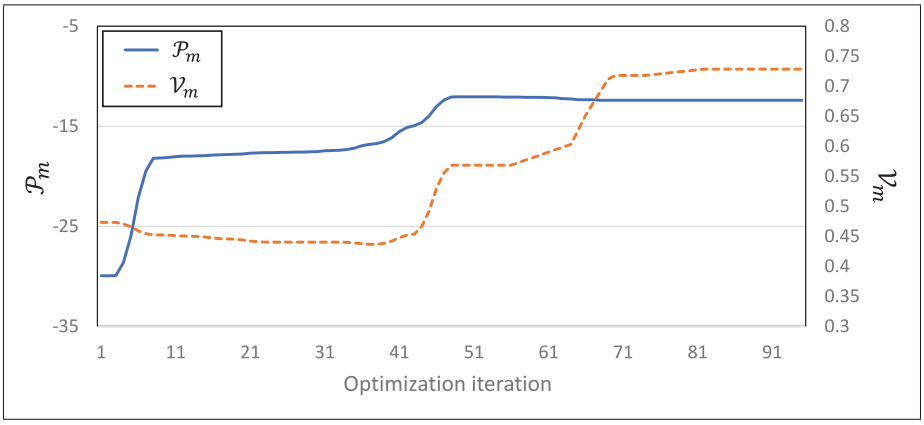


Figure 4.12 The optimization of likelihood function for an example learner that illustrates the changes in $\mathcal{P}_m$ and $\mathcal{V}_m$ during hyperparameter tuning

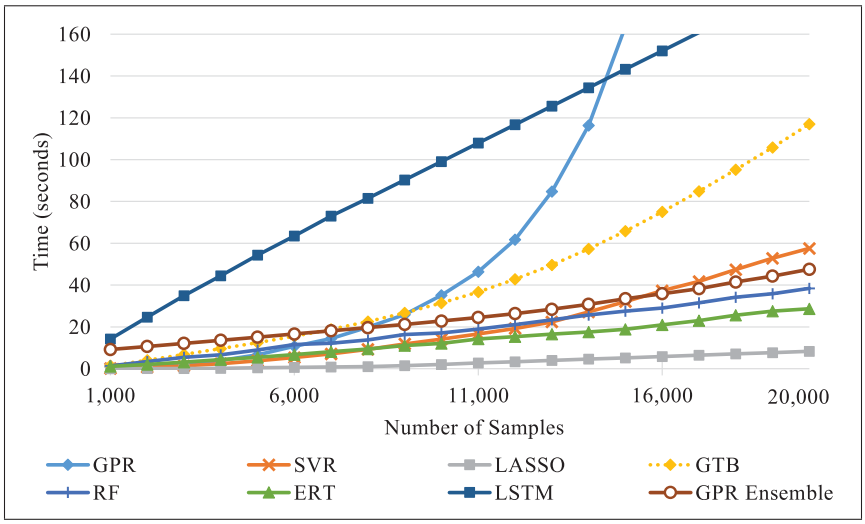


Figure 4.13 Training time of different models with varying training size

Figure 4.13 shows the execution time of different algorithms on varying size training sets. All the experiments are conducted in Python 3.7 on a workstation (Ubuntu Server 18.04.2 LTS, 8 Intel 2.40 GHz processors, 32 GB of RAM). As shown, the training time of standard GPR increases very fast (i.e., cubically) as the training size increases. For dataset of size 2000, it takes 981 seconds to train standard GPR. The training time of LSTM grows very fast (but linearly). LASSO has the shortest training time compared to other models as it is a low performance linear regression model (with high prediction error). The effect of training size on the execution time of the GPR ensemble model is significantly smaller than standard GPR model (and also LSTM and GPT). Since each expert in GPR ensemble model is trained on a segment of feature space, the size of training set has a small effect on the training time of the model. This is the result of local optimization of accuracy-diversity balance in our ensemble training.

4.5 Summary

While network traffic prediction is a powerful means to improve the network performance, it is a very challenging task. The traffic prediction model must handle different traffic patterns at various time-scales. In this chapter, we presented an algorithm for traffic prediction based on the

ensemble of GPR experts, and proposed an optimization framework to optimize the balance between accuracy and diversity of these experts. The proposed framework reduces the complexity through a divide-and-conquer approach where each learner optimizes accuracy-diversity balance in a segment of the feature space, and each segment is used to train a base learner. The proposed GPR ensemble model has been applied on real traffic traces. The experimental results show our GPR ensemble model outperforms other models in traffic prediction. In the future work, we will investigate the performance of our model regarding various traffic characteristics such as burstiness, and LRD/SRD. We intend to integrate the model into existing network controllers to improve the resource and traffic management in networks. We will also compare our model with new deep learning algorithms and graph-based neural networks to show its efficiency.

CHAPTER 5

MULTIPLE-STEP-AHEAD TRAFFIC PREDICTION IN HIGH-SPEED NETWORKS

A. Bayati, K.H. Nguyen, M. Cheriet

Department of Automated Production Engineering, École de Technologie Supérieure, University of Quebec, Montreal, Quebec, Canada H3C 1K3

Paper published in *IEEE Communications Letters*, December 2018

5.1 Introduction

In multiple-step-ahead traffic prediction, the goal is to predict the traffic performance measures (e.g., bandwidth, packet loss, and latency) forward in time, up to a particular horizon. A long horizon reveals traffic fluctuations in future steps and gives enough time to take proper decisions. However, it may also lead to traffic fluctuation in future steps. In the high-speed network management, there are many situations in which immediate changes in the network are expensive or not feasible. For example, the time required for establishing a wavelength (or lambda) in optical networks is often in the order of minutes, so it cannot be done instantly. Multiple-step-ahead traffic forecasting provides sufficient time for proactive management in such situations.

Two common strategies for multiple-step-ahead time-series prediction are *iterative* (or naive) and *direct* (or parallel) methods (Taieb et al., 2012). In the iterative approach, multiple-step-ahead time-series forecasting is achieved by making a repeated one-step-ahead prediction where the outputs in consecutive steps are used as the input for the next forecasting step (Palm, 2007). The recursive one-step-ahead prediction can be repeated up to a required *time horizon*, and only a single forecasting model is used (Girard et al., 2003). As its main drawback, the accumulation of prediction errors in the prior steps raises as the forecast horizon increases. On the other hand, in the direct approach, H different models are trained in parallel to perform H-step-ahead

time-series forecasting where the learner h ($1 \leq h \leq H$) predicts the h-th step-ahead value. The direct method requires H separate models to be trained and its time horizon is limited to H.

Our experimental results reveal that the traditional strategies fail to achieve an accurate prediction of high-speed traffics because they do not consider the traffic characteristics. Indeed, traffic of a high-speed link exhibits different patterns at different time-scales (Xie et al., 2013). For example, at the small time-scales, traffic behavior is protocol dependent (e.g., TCP, UDP, HTTP) (Willinger et al., 2003). On the other hand, at the time-scale of hours, traffic samples represent the humans' daily activities. Also, traffic exhibits Long-Range Dependency (LRD) at large time-scales, but it has Short-Range Dependency (SRD) at small time-scales (Willinger et al., 2003). The traffic behavior at a time-scale is determined by a particular set of factors corresponding to that time-scale. This traffic characteristic is beneficial for multiple-step-ahead prediction. Consider three traffic samples which have been captured at time-steps 7:00 AM, 7:05 AM, and 8:00 AM when the sampling time-scale is 5 minutes. Now consider the factors that define the traffic behavior at the time-scale of 1 hour (e.g., humans' activity). The states of these factors are almost the same from 7:00 AM to 7:05 AM, however they change supposedly from 7:00 AM to 8:00 AM. In other words, for one-step-ahead prediction at 7:00 AM, the traffic behavior at higher time-scales can be ignored. However, for 12-step-ahead prediction at 7:00 AM (i.e., to predict 8:00 AM), the models of traffic behavior at higher time-scales are required. This traffic property should be taken into account in prediction algorithm.

Different algorithms have been used for traffic prediction including ARIMA (M. Zhang et al., 2017), Artificial Neural Network (Y. Li et al., 2016), etc. In (Bayati et al., 2018), Gaussian Process Regression (GPR) has been shown to be a powerful tool for single-step traffic prediction which can handle the traffic self-similarity and periodicity. This work enhances (Bayati et al., 2018) by introducing a new multiple-step-ahead traffic prediction algorithm based on GPR. GPR framework allows studying the error propagation in the multiple-step prediction (Girard et al., 2003). The proposed algorithm consists of H GPR experts where the expert f^h captures the traffic behavior at time-scale h ($1 \leq h \leq H$). The prediction results of expert f^h are used to correct the prediction of the models at the smaller time-scales.

The algorithm is explained in Section 5.2. Sections 5.3 and 5.4 present the results and conclusion respectively. Throughout this chapter, subscripts denote the index of variables (in vectors), and superscripts determine the time-scale. Also, the vectors are presented using boldface variables.

5.2 Algorithm

5.2.1 Gaussian Process Regression (GPR)

A Gaussian process (Bayati et al., 2018) is a set of random variables such that any subset of them has a joint Gaussian distribution. GPR provides the mapping function between the input $\mathbf{X} = \{\mathbf{x}_i\}$ and (continuous) output $\mathbf{Y} = \{y_i\}$. Given the traffic samples $\mathbf{t} = \{t_i | \ 0 \leq i < n\}$, the feature vector $\mathbf{x}_i$ is a d dimensional vector created from time-series data (i.e., $\mathbf{x}_i = [t_{i-1}, t_{i-2}, ..., t_{i-d}]$), and $y_i = t_i$. Consider n pairs of input and noisy output observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, 2, ..., n\}$, and the unknown mapping function $f(\mathbf{x}_i)$:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \tag{5.1}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian noise, and function $f(\mathbf{X}) \sim \mathcal{GP}(m(\mathbf{X}), \ k(\mathbf{x}_i, \mathbf{x}_j; \theta))$ is defined with mean $m(\mathbf{X})$, covariance $k(\mathbf{x}_i, \mathbf{x}_j; \theta)$, and hyperparameters θ.

The goal is the prediction of target value y_* for new input data $\mathbf{x}_*$ which does not belong to the dataset $\mathcal{D}$. The GP assumption implies that joint distribution of the observed target values $\mathbf{Y}$ and the function value at $\mathbf{x}_*$ is a Gaussian distribution:

$$\begin{bmatrix} \mathbf{Y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \ \begin{bmatrix} K + \sigma^2 I & K_* \\ K_*^\top & k_* \end{bmatrix}\right), \tag{5.2}$$

where $f_* = f(\mathbf{x}_*)$ and $k_* = k(\mathbf{x}_*, \mathbf{x}_*; \theta)$. The element K is called *covariance matrix* of $\mathbf{X}$ and denotes a $n \times n$ matrix of the covariance values evaluated for all pairs in the input data, i.e., $[K]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j; \ \theta)$. The element K_* is a $n \times 1$ matrix for which $[K_*]_i = k(\mathbf{x}_i, \mathbf{x}_*; \theta)$. The

conditional distribution $f_*|D, \mathbf{x}_*, \theta \sim \mathcal{N}\left(\hat{f}_*, \hat{v}_*\right)$ leads to the mean and variance:

$$\hat{f}_* = K_*^\top (K + \sigma^2 I)^{-1} \mathbf{Y}, \tag{5.3}$$

$$\hat{v}_* = k_* - K_*^\top (K + \sigma^2 I)^{-1} K_*. \tag{5.4}$$

In this work, $\mathbf{t}$ is defined as the first difference of the bandwidth time-series:

$$t_i = b_i - b_{i-1}, \tag{5.5}$$

where b_i is the traffic bandwidth (measured in Bps) at time i. So, each traffic sample shows the difference of the monitored traffic bandwidth at two consecutive intervals. As the difference operator in Equation (5.5) removes trends in the traffic time-series, the mean function $m(\mathbf{X})$ can be taken to be zero.

5.2.2 Multiple-Step-Ahead Prediction

Our algorithm analyzes the traffic data at H time-scales and builds a GPR model for each time-scale. The traffic samples at each time-scale are calculated using the aggregate operation. The *aggregated process* of $\mathbf{t} = \{t_i | \, 0 \leq i < n\}$ at the aggregation level h is called $\mathbf{t}^h = \{t_i^h | \, 0 \leq i < n_h\}$ which is calculated by partitioning $\mathbf{t}$ into non-overlapping blocks of size h and calculating the sum of the blocks (Willinger et al., 2003):

$$t_i^h = \sum_{j=ih+1}^{(i+1)h} t_j \,, \tag{5.6}$$

$$n_h = \left\lfloor \frac{n}{h} \right\rfloor. \tag{5.7}$$

Aggregated process $\mathbf{t}^1$ is the same as the original process $\mathbf{t}$.

5.2.2.1 training

Using aggregated versions of the traffic, H datasets are created where the dataset $\mathcal{D}^h = \{(\mathbf{x}_i^h, y_i^h)\}$ is employed to train the GPR model f^h. The $(d+1)$-dimensional feature vector and the output for $\mathcal{D}^h$ $(1 \leq h < H)$ are:

$$\mathbf{x}_i^h = [w_i^h, t_{i-1}^h, t_{i-2}^h, ..., t_{i-d}^h], \tag{5.8}$$

$$y_i^h = t_i^h, \tag{5.9}$$

$$w_i^h = \sum_{j=ih+1}^{(i+1)h+1} t_j, \tag{5.10}$$

where w_i^h is the next step of the traffic at aggregation level $h+1$. Figure 5.1 illustrates the traffic aggregation and the feature selection at aggregation level 4. The only exception is the dataset $\mathcal{D}^H$ for which:

$$\mathbf{x}_i^H = [t_{i-1}^H, t_{i-2}^H, ..., t_{i-d}^H] \tag{5.11}$$

which does not include any sample from the higher aggregation level.

For creating the dataset $\mathcal{D}^h$ with size n, $(n+d-1) \times h$ traffic samples are required. Since $\mathcal{D}^H$ requires the maximum number of traffic samples, the total number of traffic samples needed for creating the training data is $(n+d-1) \times H$.

The time complexity of standard GPR algorithm is $O(n^3)$ where n is the number of the training samples (Girard et al., 2003). The proposed algorithm consists of H GPR experts that are trained separately; so, its time complexity is $O(H \times n^3)$. Since, H is a constant and $H \ll n$, the complexity of the proposed algorithm is the same as standard GPR.

5.2.2.2 prediction

The prediction phase includes two steps as presented in Figure 5.2: (i) single-step prediction at all the aggregations levels, and (ii) iterative predictions using f^1.

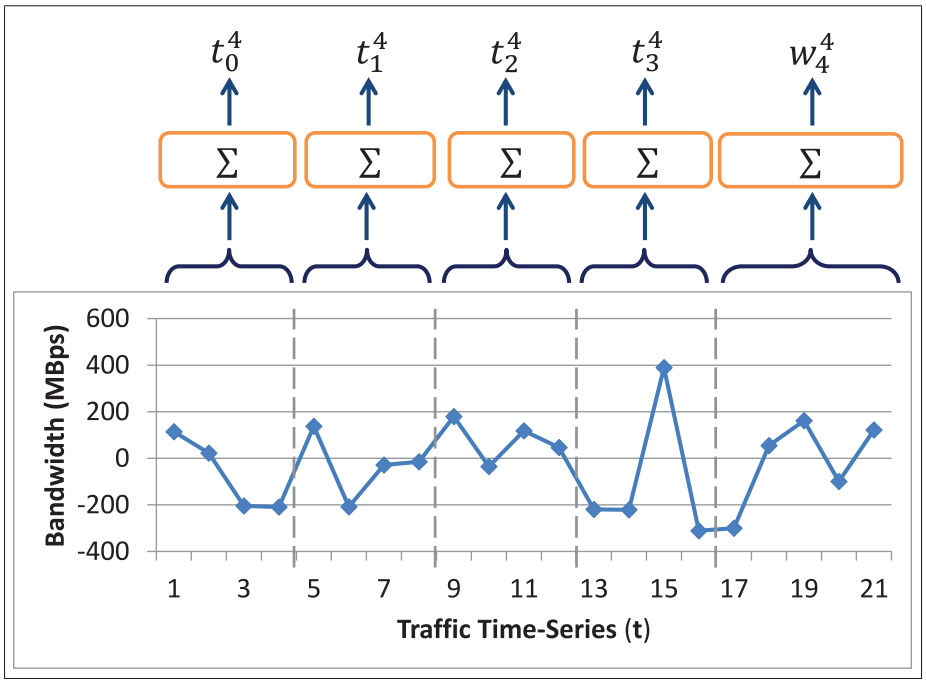


Figure 5.1 Traffic aggregation at level 4. The time-series **t** is the first difference of the traffic bandwidth. The time interval between samples is 5 minutes

In step 1, model f^h predicts value y_*^h given the input $\mathbf{x}_*^h$. However, the input $\mathbf{x}_*^h$ includes w_*^h which is unknown at the time of prediction. Considering Equations (5.8) and (5.10), w_*^h and t_*^{h+1} include the sum of the same traffic samples t_i at the prediction time, so they have the same value. According to Equation (5.9), y_*^{h+1} is the predicted value for t_*^{h+1}. Therefore, the value w_*^h can be estimated as:

$$w_*^h = y_*^{h+1}. \tag{5.12}$$

The feature vectors in $\mathcal{D}^H$ do not include w_i^H. Thus, the model f^H does not require the prediction results at the higher aggregation level, and its prediction is based on only the samples from aggregation level H. Model f^H provides y_*^H, the single-step-ahead prediction at aggregation level H. This predicted value is used instead of w_*^{H-1} in $\mathbf{x}_*^{H-1}$ which is the input to f^{H-1} to predict y_*^{H-1}. This process is repeated in step 1 from model f^{H-1} to f^1. The results is $\mathbf{Y}_* = \{y_*^h \mid h = 1, 2, ..., H\}$ which is utilized in step 2.

Algorithm 5.1 Multiple-Step-Ahead Prediction

Input: $\mathbf{t}^h$, f^h
Output: y^1_{s+h}

$\mathbf{x}^H_* \leftarrow [t^H_i, t^H_{i-1}, ..., t^H_{i-d+1}]$ **> Step 1**
$y^H_* \leftarrow f^H(\mathbf{x}^H_*)$
for $h := H - 1 to 1$ **do**
  $\mathbf{x}^h_* \leftarrow [y^{h+1}_*, t^h_i, t^h_{i-1}, ..., t^h_{i-d+1}]$
  $y^h_* \leftarrow f^h(\mathbf{x}^h_*)$
end
$y^1_{s+1} \leftarrow y^1_*$ **> Step 2**
for $h := 2 to H$ **do**
  $w^1_{s+h} = y^{h+1}_* - y^{h-1}_*$
  $\mathbf{x}^1_{s+h} \leftarrow [w^1_{s+h}, y^1_{s+h-1}, y^1_{s+h-2}, ..., y^1_{s+h-d}]$
  $y^1_{s+h} \leftarrow f^1(\mathbf{x}^1_{s+h})$
end

In step 2, the multiple-step-ahead predictions are achieved by iterative single-step estimations using f^1. For t_{s+h} , the $h-1$ previous predicted values and $\mathbf{Y}_*$ are required. The feature vector $\mathbf{x}^1_{s+h}$ is defined as:

$$\mathbf{x}^1_{s+h} = [w^1_{s+h}, y^1_{s+h-1}, y^1_{s+h-2}, ..., y^1_{s+h-d}], \tag{5.13}$$

where w^1_{s+h} is estimated based on the values in $\mathbf{Y}_*$:

$$w^1_{s+h} = y^{h+1}_* - y^{h-1}_*. \tag{5.14}$$

The value w^1_{s+h} has a crucial role in the algorithm. It conveys the knowledge about the traffic behavior at longer time-scales to f^1. In the following section, the importance and role of w^1_{s+h} have been analyzed.

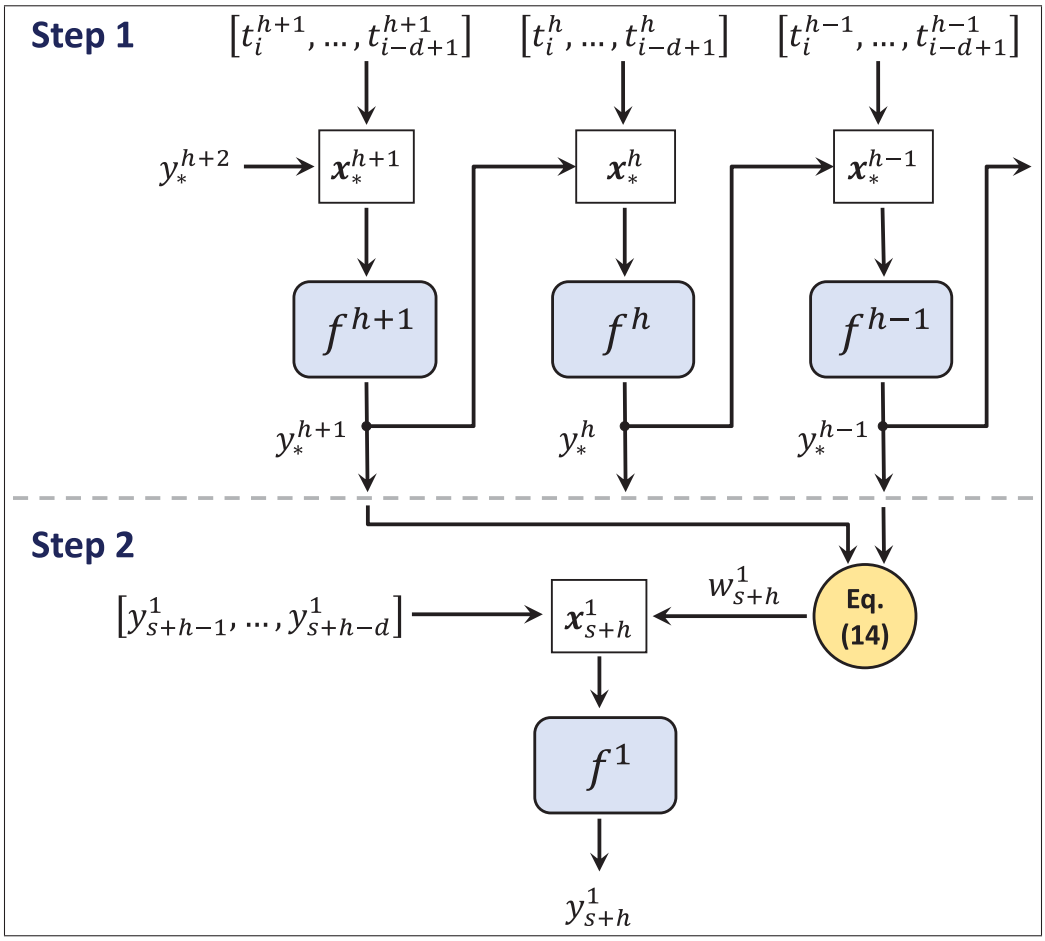


Figure 5.2 The workflow model of the prediction algorithm

5.2.3 Feature Importance

The features in $\mathbf{x}_{s+h}^1$ do not contribute evenly to the prediction results. In a machine learning problem, *feature importance* measures the role of a feature in the prediction accuracy. Generally, the correlation between a feature and the target value in a prediction problem reveals the feature importance. According to the traffic autocorrelation function (ACF), it can be shown y_{s+h-1}^1 and w_{s+h}^1 are the most important features in $\mathbf{x}_{s+h}^1$. By definition, the traffic ACF satisfies (Karagiannis et al., 2004):

$$\rho(r) \sim cr^{-\beta}, \tag{5.15}$$

where c is a constant. Traffic shows LRD if $0 < \beta < 1$, and SRD if $1 < \beta < 2$. In both cases, ACF decreases as the time lag between traffic samples increases:

$$0 \leq r \leq q \;\Rightarrow\; cov(t_s, t_{s+r}) \geq cov(t_s, t_{s+q}). \tag{5.16}$$

According to (5.16), the importance of t_{s+r} as a feature for the prediction of t_s is greater than t_{s+q}. Equation (5.15) indicates the importance of a feature in $\mathbf{x}^1_{s+h}$ drops exponentially as the time lag between the element and the target value increases.

5.2.4 Effects of Feature Vector on Reducing Error Propagation

In the proposed algorithm, the predicted value at time-step h is used as one of the input features for forecasting the next time-steps. So, the error in prediction at time-step h (or the uncertainty in the feature vector) is propagated through the forecasts at next time-steps. It means the error propagation is reduced as the input uncertainty is minimized. This section illustrates w^1_{s+h} reduces the uncertainty of the input feature vector and thus, lowers the error propagation.

This is achieved mainly by controlling the *predictive variance* which is a function of the uncertainty of $\mathbf{x}^1_{s+h}$. Thus, the predictive variance and accordingly, the error propagation is decreased by reducing the features' uncertainty. Accumulation of errors in the iterative predictions causes the uncertainty of the feature vector. Therefore, during the iterative prediction, $\mathbf{x}^1_{s+h}$ must be considered as a random variable.

First, the prediction uncertainty has to be formulated which can be done based on the *predictive variance*. Assume $\mathbf{x}^1_{s+h}$ as a random point with distribution $\mathcal{N}(m(\mathbf{x}^1_{s+h}), v(\mathbf{x}^1_{s+h}))$ where $v(\mathbf{x}^1_{s+h})$ is a $(d+1)\times(d+1)$ matrix. The Gaussian assumption for $\mathbf{x}^1_{s+h}$ allows the analytical approximation for the predictive variance of y^1_{s+h} (Girard et al., 2003):

$$v(y^1_{s+h}) = v_{m(\mathbf{x}^1_{s+h})} + V_h, \tag{5.17}$$

$$V_h = Tr\left\{v(\mathbf{x}^1_{s+h})\left(\frac{1}{2}\frac{\partial^2 \hat{v}_{\mathbf{x}^1_{s+h}}}{\partial \mathbf{x}^1_{s+h}\partial \mathbf{x}^{1\top}_{s+h}} + \frac{\partial \hat{f}_{\mathbf{x}^1_{s+h}}}{\partial \mathbf{x}^1_{s+h}}\frac{\partial \hat{f}_{\mathbf{x}^1_{s+h}}}{\partial \mathbf{x}^1_{s+h}}^{\top}\right)\right\}. \tag{5.18}$$

The predictive variance $v(y^1_{s+h})$ is the sum of two terms (i) $v_{m(\mathbf{x}^1_{s+h})}$, and (ii) V_h. The first term is the GPR prediction uncertainty for input $m(\mathbf{x}^1_{s+h})$ shown in (5.4). This term is larger at the inputs that are not similar (or close) to the training data compared to the point which are nearby the training data. The second term (in predictive variance) contains the variance (or uncertainty) of $\mathbf{x}^1_{s+h}$. This term is equal to zero when the input is not a random point. As the number of random elements in $\mathbf{x}^1_{s+h}$ raises, the value of V_h increases. So, the uncertainty of $\mathbf{x}^1_{s+h}$ is expected to be more than uncertainty of $\mathbf{x}^1_{s+h-1}$. Accordingly, $v(y^1_{s+h})$ is expected to be greater than $v(y^1_{s+h-1})$. This shows the errors are accumulated in y^1_{s+h}s as iterative prediction goes further ahead in time.

The level of uncertainty of w^1_{s+h} is independent of the prediction time-step. Because, w^1_{s+h} is the result of single-step predictions the second term of its predictive variance is equal to zero. Therefore, employing w^1_{s+h} as a feature decreases the uncertainty of the feature vector.

5.3 Experimental Results

We performed our experiments on two well-known traffic datasets: (i) CAIDA Anonymized Internet Traces from 2008 to 2015 (*The CAIDA UCSD Anonymized Internet Traces*, 2008-2015), and (ii) Abilene Network traffic data from $2007-01-01$ to $2007-10-14$ (*Internet2 Abilene Network*, 2007). Abilene dataset has been used for prediction on the long time-steps (i.e., 5 minutes), and CAIDA dataset has been used for prediction on short time-steps (i.e., 30 seconds). In the experiment, each dataset has been divided into two non-overlapped subsets. The first subset has been used for the model selection (i.e., selecting the optimal size of the training set, and the optimal number of features d) for each algorithm using a cross-validation process. The second subset has been divided into 100 portions, and each portion has been employed to create a pair of the non-overlapped train and test sets (random train-test split). For each pair, the models have been fitted on the train set and then, evaluated on the test set. The reported results are the average of 100 prediction error measurements which have been achieved from this process.

We compared our model with four multiple-step-ahead prediction algorithms: ARIMA, FARIMA, Long Short-Term Memory (LSTM) network, and Convolutional LSTM network (Xingjian et al., 2015). ARIMA and FARIMA are two well-known time-series models. LSTM and Convolutional LSTM are results of recent advances in the deep learning algorithms. LSTM is a type of Recurrent Neural Networks (RNN), and Convolutional LSTM is based on the fully connected LSTM (Xingjian et al., 2015). In Convolutional LSTM the redundant connections have been reduced to improve the prediction performance. Since both models have been proven to be powerful tools for time-series forecasting, they are proper candidates for comparisons with our algorithm. We employed the Rational Quadratic covariance function in our model because of its ability to capture traffic characteristics (Bayati et al., 2018). LSTM and Convolutional LSTM have been implemented as multivariate predictors. The number of units (or neurons) in the LSTM layer, and the order of ARIMA have been determined through the model selection. The order of FARIMA has been determined by maximum likelihood estimation.

The prediction error has been measured based on normalized mean-square error (NMSE):

$$NMSE = \frac{1}{\sigma^2 N} \sum_{i=1}^{N} (t_i^1 - y_i^1)^2 \tag{5.19}$$

where N is the size of the test set and σ^2 is the variance of $\mathbf{t}$. A value of $NMSE = 0$ corresponds to the perfect predictor.

Figure 5.3 illustrates the results of 10-steps ahead prediction on traffic data from Abilene network. The time difference between prediction steps is 5 minutes. As shown, the prediction accuracy of the iterative and direct method is less than others. In the first 3 steps, LSTM, Convolutional LSTM, and the proposed algorithm perform almost the same. However, the increase in the prediction error of the proposed algorithm is remarkably less than other models for the next steps.

Figure 5.4 represents the results of the traffic forecasting at the time-scale of 30 seconds using the traffic data from CAIDA. It is clear that the prediction errors at time-scale 30 seconds are

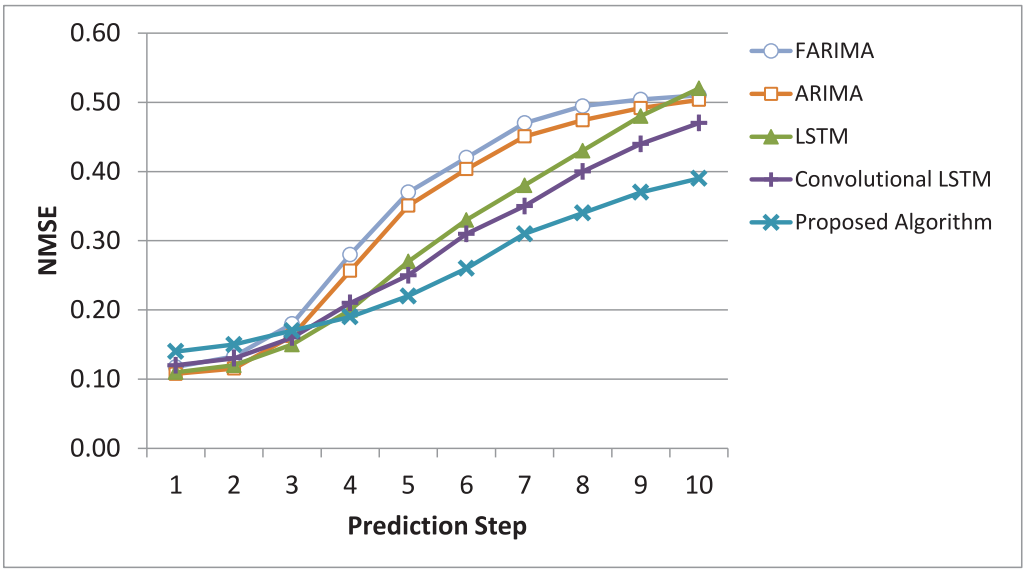


Figure 5.3 10-steps ahead prediction on the Abilene network traffic data (the time lag between steps is 5 minutes)

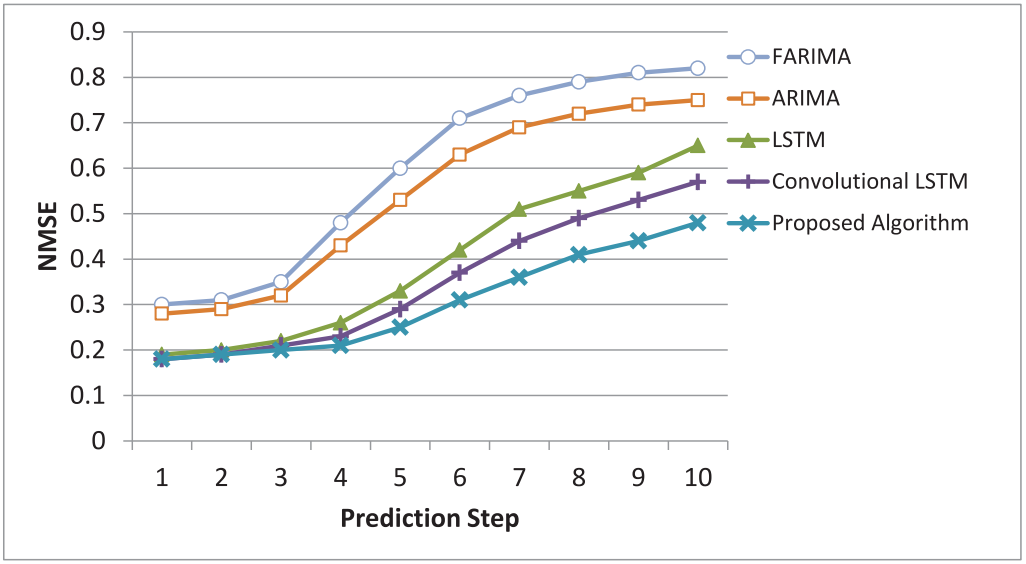


Figure 5.4 10-steps ahead prediction on the CAIDA traffic data (the time lag between steps is 30 seconds)

bigger for all the algorithms compared to the prediction at time-scale of 5 minutes. In both cases, the prediction accuracy of the proposed model is higher than any other algorithm.

5.4 Summary

In this work, we presented an algorithm for multiple-step-ahead traffic prediction based on the GPR in which the multi-scale traffic behavior is exploited to improve traffic prediction and to reduce the error propagation. We analyzed the error propagation in the proposed algorithm and showed that the traffic modeling at higher time-scales is essential and effective for an accurate multiple-step-ahead prediction. In the future work, we will investigate to use the traffic data from the same period of previous days. Also, we will employ the algorithm to improve the resource and traffic management in networks.

CHAPTER 6

REDUCING NETWORK ENERGY CONSUMPTION USING ADAPTIVE LINK RATE AND TRAFFIC PREDICTION

A. Bayati, C. Pham, K.H. Nguyen, M. Cheriet

Department of Automated Production Engineering, École de Technologie Supérieure, University of Quebec, Montreal, Quebec, Canada H3C 1K3

Paper submitted for publication, July 2020

6.1 Introduction

Network energy consumption causes a considerable portion of communication costs and environmental-related concerns (G.-R. Liu et al., 2018). It is independent of network traffic load and remains high even during underutilized periods (Manjate et al., 2018). Power proportionality in network devices is a technique to address this problem (Vitturi & Tramarin, 2015). A network element is energy-proportional if its power consumption is proportional to its carried traffic (Christensen et al., 2010). There are two main approaches to provide energy proportionality: (i) deactivating (or powering off) the device during idle periods (Addis et al., 2014), and (ii) using adaptive link rate (ALR) (L. Wang et al., 2013),(Andrews et al., 2012). In the first approach, data is sent faster in the active interval to have a longer sleep interval. In the second approach, link speed is decreased when the link is underutilized. It is based on the fact that the interface cards with lower transmission rates consume smaller amounts of power (Tang et al., 2012).

ALR has been employed in energy-aware routing (EAR) methods to optimize the network energy consumption (Manjate et al., 2018). EAR solutions have been investigated for different networks, including SDN networks (Destounis et al., 2018), data centers (Huin et al., 2018), and edge computing (Trinh et al., 2018). In ALR-based EAR, the network resources and link capacities are adapted to the traffic load to ensure energy conservation. It forms a mixed integer programming (MIP) optimization problem with energy consumption as its objective function

(Tang et al., 2012). The control variables are the paths of the flows and states of the links. It examines the possible flow allocations to minimize power consumption while guarantees the network performance. It is a centralized approach controlled by the network controller with a global view of resources and traffic.

In networks with static nonvarying traffic, the instance of ALR-based EAR problem does not change over time, and it needs to be solved just once the flows are established. Thus, the *network configuration* is stable under this assumption. The network configuration is defined as the set of selected paths for traffic flows and the link states. These two pieces of information (traffic routes and links states) are required for network management and power consumption calculation. Unlike the static traffic example, in networks with time-varying flows, the instance of optimization problem evolves across time due to changes in traffic demands. In this case, we need to solve a sequence of ALR-based EAR problems repeatedly, and constantly reconsider the network configuration. At each iteration of optimization, there might be some differences between the optimal solution and the actual network configuration. Since the optimum configuration changes over time in response to the dynamically changing traffic demands, the network must be reconfigured at each iteration to have the minimum energy cost. The *network reconfiguration* includes the traffic flow reroutings and the link state transitions. In other words, the network reconfiguration is defined as any changes in the link-state or traffic paths.

The network reconfigurations introduce instability to the system and decrease the QoS (Destounis et al., 2018). A link rate transition leads to packet loss and delay because it requires a considerable amount of time (Gunaratne et al., 2008), and the link is not functional during the transition time (B. Zhang et al., 2008). Also, flow rerouting affects the order of packets, and they can be received out of order. The number of reconfigurations, however, needs to be minimized to avoid their negative impacts (Destounis et al., 2018). The network reconfiguration has recently been studied as an issue within the networking community. For example, the costs of reconfiguration in the Software-Defined Networking (SDN) network were investigated in (Destounis et al., 2018). They studied the effects of reconfiguration on the network and QoS and proposed a minimum cost optimization model to reduce the frequency of reconfigurations. They did not, however, consider

the network power consumption and ALR technology, which are investigated and pointed out as the main topic of our work. To the best of our knowledge, previous ALR-based EAR studies did not consider the effects of network reconfigurations in their optimization approaches.

This work aims to reduce the number of network changes in the periodic ALR-based EAR. In particular, our approach is to select a network configuration in each time-slot, which can achieve the following objectives: (i) it reduces the network energy consumption in the current time-slot, and (ii) it needs the minimum number of changes to adapt to the traffic demands in the future time-slots. Unlike existing ALR-based EAR approaches which only considered the first objective, our proposed approach does not focus only on the energy consumption in the current time-slot. Our approach also considers the future changes in traffic demands and proactively plans to avoid significant network reconfigurations in future time-slots. In addition, our approach stabilizes the network by preventing the *temporary reconfigurations* in iterative network optimization. Temporary reconfigurations are defined as the set of short-lived actions (i.e., flow rerouting or link state transitions) that help to reduce the energy costs in a short period; however, they have to be reverted shortly after they are applied to the network because they hurt the energy optimality in next time-slots. The temporary reconfigurations can be detected and avoided in advance if the future values of traffic are given. Thus, our approach needs the (estimation of) future values of traffic demands.

A prediction algorithm is required in this approach to forecast the traffic demands in future time-slots and to eliminate the temporary reconfigurations. Any *multiple-step-ahead* time-series predictor can be exploited to achieve the predicted values. However, in order to fix a model, we employed our multiple-step-ahead prediction algorithm (Bayati et al., 2018). Our traffic prediction algorithm models the traffic multiscale behaviour using a set of Gaussian Process Regression (GPR) learners. GPR is a kernel-based learning algorithm which can handle traffic characteristics such as short/long range dependency, self-similarity, and periodicity (Bayati et al., 2016). In (Bayati et al., 2018), we showed that our multiple-step-ahead prediction algorithm outperforms powerful time-series predictors.

Table 6.1 Model notation

Variable	Description
$\mathcal{L}$	the set of links in the network
L	number of links
$\mathcal{S}$	the set of possible states for a link
S	number of possible link states
a_l	state of link l
e_s	interface power consumption in state s
c_s	link capacity in state s
C_{max}	maximum capacity of a link
u_l	utilization of link l
$v_{l,s}$	a binary variable which is equal to 1 if link l operates in state s
$\mathcal{K}$	set of all the traffic flows in network
K	total number of traffic flows
d_k	bandwidth demand of flow k
P_k	set of precalculated paths for flow k
N	number of precalculated paths for each flow
r_k	the selected route for flow k
$w_{p,k}$	a binary variable which is equal to 1 if path p has been selected for flow k
G	network configuration including the state of links and selected paths for flows
$E(G)$	the power cost function
E_{max}	maximum power consumption of network
$R(G_i, G_j)$	reconfiguration cost for transition from G_i to G_j
I	available traffic information
T	prediction horizon (the number of future steps of traffic)
$\mathcal{T}$	range of time-slots that are considered in RAP
$Q(G)$	the objective function of RAP

Summarily, the contributions of this work are as follows:

- We design an innovative energy-efficient framework using the multiple-step-ahead prediction of the traffic load which reduces the network power consumption while prevents the QoS degradation. This framework is called *Rate Adaptation using Prediction (RAP)*.

- We pose an optimization formulation based on integer linear programming (ILP) where the objective function is comprised of energy cost and reconfiguration cost incurred in a long-term consideration.

- To find a solution for our problem, which challenges any solver in polynomial time of solving, we advocate a heuristic algorithm based on simulated annealing (SA) which efficiently solves the ILP problem.

- Finally, we executed intensive experiments to evaluate the proposed framework using real traffic data (from Abilene network (*Internet2 Abilene Network*, 2007)) and large network topologies (GEANT network (Uhlig et al., 2006), and a fat-tree topology (Dehury & Sahoo, 2020)) which mimic realistic scenarios.

The remainder of this chapter is organized as follows. The problem has been explained in Section 6.2. It discusses the background on ALR and EAR and formulates the problem. Section 6.3 describes different parts of the proposed approach for RAP. The experimental results are reported in Section 7.1, and finally, the conclusion is drawn in Section 6.4.

6.2 Problem Statement

The problem is explained and formulated in this section, where we discuss ALR and EAR and propose our approach. The model notation is summarized in Table 6.1. First, the basics of ALR and EAR are discussed in Section 6.2.1. Section 6.2.3 uses an intuitive example to show the benefits of incorporating prediction in the periodic ALR-based optimization. Finally, our proposed approach (RAP) is formulated in Section 6.2.4.

6.2.1 ALR and Related Technologies

ALR has been proposed to reduce network power consumption. Network interfaces consume a notable portion of energy. The power consumption of a regular network interface is independent of its utilization because the link operates with the full capacity regardless of the traffic load (Gunaratne et al., 2008). ALR adjusts the interface capacity according to the link utilization to minimize the required energy. There are many examples of multiple-rate network interfaces that are capable of changing the link speed without interrupting the traffic. For example, a prototype network adapter for ALR has been created and tested in (Gunaratne et al., 2008), which allows changing the link-rate between 10 Mbps, 100 Mbps, and 1 Gbps (as shown in Table 6.2). Other researches on Ethernet-based ALR, including the prototypes introduced in (Tang et al., 2012) and (B. Zhang et al., 2008), focused on providing multiple-rate links.

Table 6.2 Link power consumption in different states

State (s)	**Rate** (c_s)	**Power Consumption** [mW] (e_s)
1	100Mbps	351
2	1Gbps	697
3	10Gbps	2600

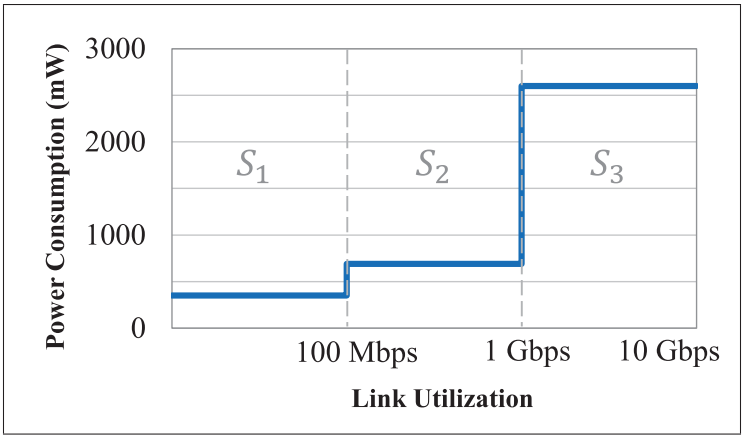


Figure 6.1 Discrete step function of link power consumption $E(u_l)$

Besides those research prototypes, many commercial Ethernet network adaptors exist that allow implementing the ALR in real-life networks. For example, Intel has recently introduced its Multi-Rate Ethernet PHY (*Multi-rate Ethernet PHY Intel FPGA IP Release Notes - 2018*, n.d.), which can dynamically support multiple data rates without any design regeneration or device re-installation. This technology enables the creation of a 1G to 10G link rates that allow dynamic reconfiguration across all Ethernet rates from 10M, 100M, 1G, 2.5G, 5G, and 10G (*Support Documents for Multi-rate Ethernet PHY Intel FPGA IP Core*, n.d.). Many Ethernet network adapters have been introduced based on Intel Multi-Rate Ethernet PHY technology, including Intel Ethernet Network Adapter X710-T2L and X710-T4L (*Intel Ethernet Network Adapter X710-T2L*, n.d.), that support five different link capacities (Table 6.3). Furthermore, many categories of network switches, such as Cisco Nexus 9500 Series or Cisco Nexus 7000 Series Switches, support multi-rate Ethernet links and allow each interface to auto-negotiate its speed mode with the other interface (*Cisco Nexus 9500 Series Switches Data Sheet*, n.d.). Those switches are excellent choices for the core, aggregation, or gateway layers in data centers networks.

Two main characteristics of the above Ethernet technologies are necessary to implement ALR: (i) they allow switching the link speed dynamically without interrupting the traffic, and (ii) their energy consumption decreases as the link rate decreases. For example, the power consumption of a prototype interface in Table 6.2 decreases by reducing the link speed. Also, the power consumption of Intel Ethernet Network Adapter X710-T4L is shown in Table 6.3 follows the same pattern. According to Table 6.3, the power consumption of the interface in 100 Mbps (which is 6.1 W) is nearly %42 of the maximum consumption (in 10 GbE), highlighting the potential ALR gain.

In the ALR schema, the link operates in S states and consumes a lower amount of energy when operating with lower capacities. For example, the three-state configuration illustrated in Table 6.2 has been derived from technical documents of Ethernet devices (Vitturi & Tramarin, 2015). The discrete step increasing function $E(u_l)$ in Fig. 6.1 gives the power consumption

of ALR-enabled link l with load u_l based on Table 6.2. Power consumption decreases with transmission rate (i.e. $e_{s-1} < e_s \Leftrightarrow c_{s-1} < c_s$).

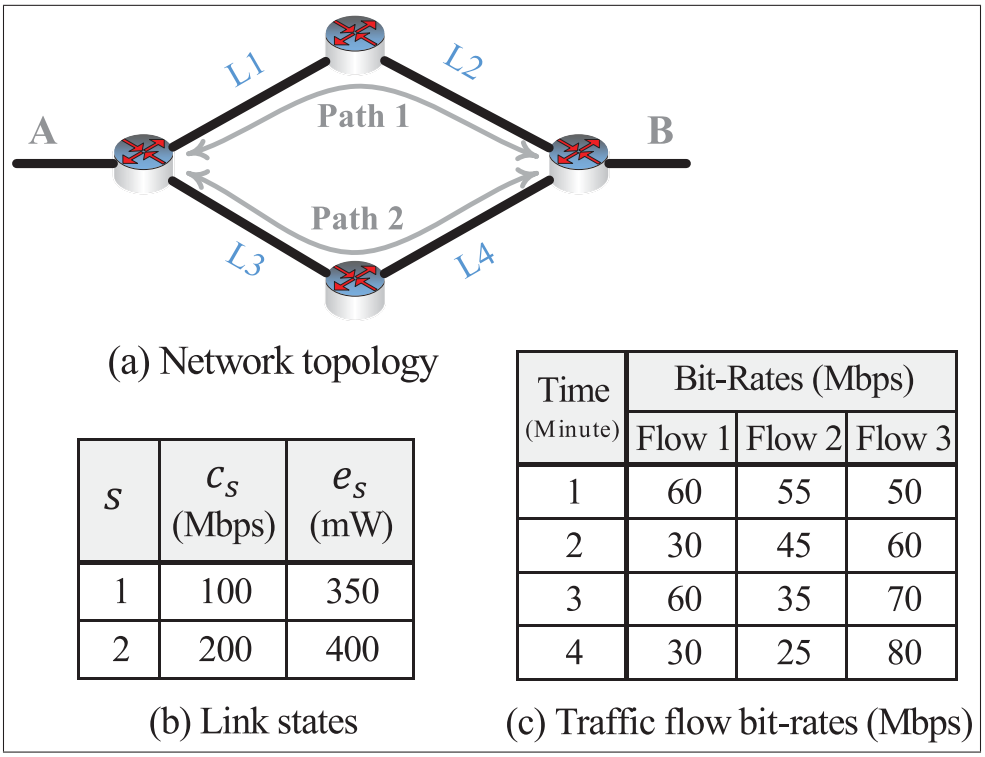
Table 6.3 Energy Consumption of Intel Ethernet Interface X710-T4L (Quad Port) at different rates

State (s)	Link Speed	Typical Power	Max Power
1	100 Mbps	5.7 W	6.1 W
2	1 GbE	7.1 W	7.5 W
3	2 GbE	9.5 W	9.9 W
4	5 GbE	10.5 W	11.1 W
5	10 GbE	13.6 W	14.2 W

Another ALR scheme is established in Energy Efficient Ethernet (EEE) introduced in IEEE 802.3az (Vitturi & Tramarin, 2015). It enables network ports to switch between a higher power state (data mode) and a lower power state (LPI mode) in response to whether data is flowing through them or not. The EEE can be depicted as an ALR mechanism with two states: Idle state (with zero traffic and minimal consumption), and Active state (with maximum consumption). By the way, there is a major difference between EEE and above multi-rate technologies. In the multi-rate Ethernet, the network controller manages the rate of the link. In EEE, however, the link state is automatically switched to LPI when there is no traffic on the link. Therefore, the network controller cannot force the interface to reach the LPI as long as there is traffic on the EEE-enabled link. However, the network controller can re-route all the traffic on the EEE-enabled link to other paths (if possible), so that the link can enter LPI mode automatically. That means the network controller can indirectly activate the LPI mode.

In this work, our modelling and simulation are, for the most part, centred around Ethernet-based networks to be able to examine ALR in real conditions. Well-known examples of these networks are the datacenter networks, which are essential components in cloud computing and edge computing. In edge computing (also known as the cloud close to the ground), the computational facilities are provided at the edge of the network. It relies on the machine-to-machine and device-to-device interactions supported by the nano-data centers (nDCs) and micro-DCs (mDCs)

(Kaur et al., 2018). There are thousands of servers and hundreds of thousands of links in a datacenter, and a central controller regulates the network (i.e., SDN). The datacenter network contributes a notable portion of the datacenter energy cost. Our algorithm offers an ALR-based approach for reducing datacenter network power usage and avoiding network instability.



(a) Network topology

s	c_s (Mbps)	e_s (mW)
1	100	350
2	200	400

(b) Link states

Time (Minute)	Bit-Rates (Mbps)		
	Flow 1	Flow 2	Flow 3
1	60	55	50
2	30	45	60
3	60	35	70
4	30	25	80

(c) Traffic flow bit-rates (Mbps)

Figure 6.2 An intuitive example to show the periodic ALR-based EAR and its limitations. (a) a network topology with four links and two paths between A and B. (b) the link capacities and power consumption in different link states. (c) the bandwidth requirements for three flows at time-slots $t = 1, 2, 3, 4$

6.2.2 Energy Aware Routing (EAR)

ALR-based EAR is a means to allocate flows to available paths in a network with ALR-enabled interfaces. It aims to minimize the power consumption while preserves adequate link capacity to handle the traffic flows. This optimization consists of two tasks: allocating the traffic flows to appropriate paths, and determining the links rates. In the networks with steady traffic flows, the

ALR-based EAR can be solved once the flows are established. For example, if traffic demands are established based on the service level agreements (SLA), their bandwidth requirements are static within the duration specified in the contracts. Under this assumption, the network is optimized once the flows are established, and there is no need to re-optimize the network because the traffic flows are not fluctuating.

The bandwidth requirements fluctuate in successive time-slots when the traffic demands are time-varying. For example, the inter-data-center (J. M. Wang et al., 2015) and intra-data-center (Cao et al., 2016), traffic flows have dynamic time-varying components. Since the traffic demands are changing, the instance of optimization problem varies over time. Thus, the network configurations must be adjusted to the best-found solution in every time-slot to maintain the lowest power consumption. This periodic re-optimization provokes a vast number of network reconfigurations which significantly reduce the QoS (Moulierac & Phan, 2015). We could optimize this overhead by selecting the current configuration considering the forthcoming traffic loads in future time-slots. In other words, the selected network configuration must satisfy two criteria: (i) minimizing the energy consumption in the current time-slot, and (ii) optimizing the modifications that are required to be adapted to the demands in next time-slots. The following example gives an intuitive illustration of this approach.

6.2.3 Example

There are four links and two available routes between A and B in the network shown in Fig. 6.2. The links are ALR-enabled, and each link can operate with two different capacities (i.e., $100Mbps$, and $200Mbps$). The bandwidth requirements at time-slots $t = 1, 2, 3, 4$ are given for three flows while the time-scale is 1 minute. The bit-rate of Flow 1 has a periodic pattern and is oscillating between two values ($30Mbps$ and $60Mbps$). The bit-rates of flows 2 and 3 exhibit linear patterns while Flow 2 decreases and Flow 3 increases. In each time-slot, the paths and the link states must be determined using ALR-based EAR. The network optimization is solved repeatedly at each time-slot to select the optimal configuration.

Consider the flow bit-rates at $t = 1$. According to the demands (at $t = 1$) and the link capacities, there are three possible configurations as shown in Fig. 6.3a (G1, G2, G3). Two flows are on one path, and the other flow is routed on another path. The energy consumption of G1, G2, and G3 is the same and equals $1500mW$ (two links are in state 1 and two links are in state 2, so that the calculation is as follows $(2 \times 350) + (2 \times 400) = 1500$). In the ALR-based EAR, the configurations G1, G2, and G3 have the same chance to be selected as the solution at $t = 1$ because they all minimize the energy cost.

There are three possible network configurations at $t = 2$ shown in Fig. 6.3b (G4, G5, and G6). According to the demands at $t = 2$, the optimal solution is to route Flow 1 and Flow 2 on one path and assign Flow 3 to another path. Thus, G4 is the optimal configuration, and it would be applied to the network at $t = 2$. The number of changes to the network configuration at $t = 2$ (for applying G4) depends on the configuration selected at $t = 1$. Fig. 6.3c illustrates the required number changes for the reconfiguration at $t = 2$. As shown, reconfiguration from G1 to G4 requires the minimum number of modifications. In other words, G1 requires the minimum changes to be adjusted to the optimal solution at $t = 2$. Therefore, if we select G1 at $t = 1$, we reduce the number of changes to the network in the next time-slot. ALR-based EAR cannot achieve this goal because it considers only the traffic demands at the current time-slot. To achieve this goal, we need to concern the predicted values of bit-rates. Therefore, we introduce our approach which is called Rate Adaptation using Prediction (RAP) to address the shortcomings of ALR-based EAR. RAP incorporates the multiple-step-ahead traffic prediction into periodic ALR-based optimization to reduce energy consumption and to maintain network performance.

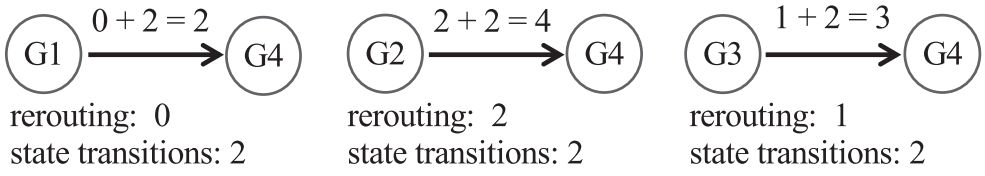
The above example used a simple scenario to present the ALR-based EAR and to point out its limitations. It relaxed many constraints associated with the real networks to clearly demonstrate the concept. Unlike this example, we consider the real network environment and constraints in our approach. We use the real-life traffic time-series (from Abilene) which exhibit complicated patterns. Also, we validate our approach over well-known network topologies (i.e., GEANT network, and a fat-tree topology) with large numbers of nodes and links.

Possible Configs.	Selected Paths			Link States				Energy Consum. (mW)
	Flow 1	Flow 2	Flow 3	L1	L2	L3	L4	
G1	Path 1	Path 1	Path 2	2	2	1	1	1500
G2	Path 1	Path 2	Path 1	2	2	1	1	1500
G3	Path 1	Path 2	Path 2	1	1	2	2	1500

(a) Possible configurations at $t = 1$

Possible Configs.	Selected Paths			Link States				Energy Consum. (mW)
	Flow 1	Flow 2	Flow 3	L1	L2	L3	L4	
G4	Path 1	Path 1	Path 2	1	1	1	1	**1400**
G5	Path 1	Path 2	Path 1	2	2	1	1	1500
G6	Path 1	Path 2	Path 2	1	1	2	2	1500

(b) Possible configurations at $t = 2$



(c) Number of required changes for three reconfigurations

Figure 6.3 Possible configurations for the network in Fig. 6.2 at time-slots (a) $t = 1$, and (b) $t = 2$. Each configuration determines the selected paths and state of the links. (c) number of required changes for reconfiguration from G1, G2, and G3 to G4

6.2.4 Rate Adaptation using Prediction (RAP)

In this section, we formulate the RAP as an optimization problem. There are K traffic flows in the network, and N precalculated paths for each flow. A path is defined as a set of the links. At t_0, the traffic bit-rates during time-slots $\mathcal{T} = \{t_0 + i \mid i = 0, ..., T\}$ are given. The current volumes of demands (at $t = t_0$ which is called $I(t_0)$) are known (by monitoring). Predictor f

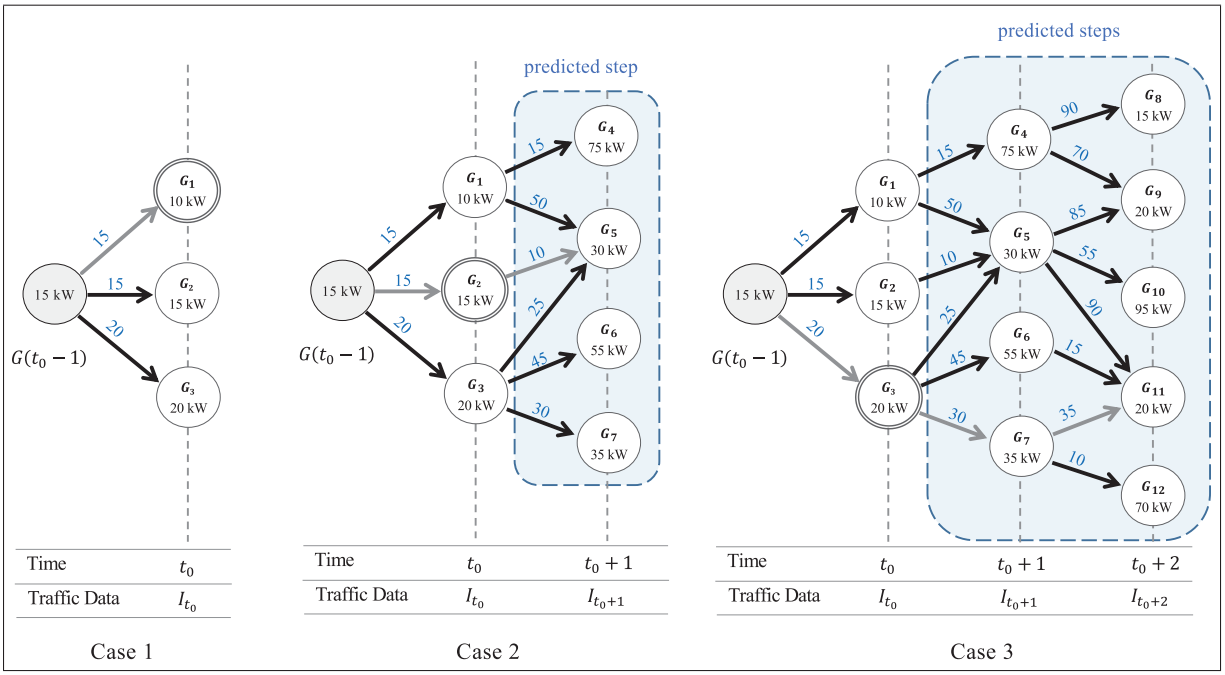


Figure 6.4 Network power optimization problem in three scenarios: (1) the optimization is done only based on the observations, and the traffic prediction is not used, (2) the optimization is done based on the traffic observations (at t_0) as well as one-step-ahead traffic prediction (at $t_0 + 1$), and (3) the optimization is done based on the traffic observations (at t_0) as well as the predictions at $t_0 + 1$ and $t_0 + 2$

gives the volumes of demands during the next T steps ($\{I(t_0 + i), \mid i = 1, 2, ..., T\}$). $I(t)$ includes the bit-rates of all flows at t:

$$I(t) = \{d_k(t) \mid \forall k \in \mathcal{K}\}\,. \tag{6.1}$$

The input of the optimization problem includes the set of precalculated paths (P_k), traffic bit-rates in current time-slot and (predicted) bit-rates in next T steps ($I(t)$, $\forall t \in \mathcal{T}$), and the current network configuration ($G(t_0 - 1)$). The output configuration $G(t_0)$ minimizes energy cost in the current time-slot and needs minimum changes to adapt to the future demands. Decision variables are the selected paths for the flows (r_k) and the states of the links (a_l). Now, we present the objective function and the constraints of our problem formulation as follows.

6.2.4.1 Objective function

We introduce an objective function (Q) which makes a trade-off between the energy cost and the number of future changes. Fig. 6.4 explains the trade-off between energy cost and number of changes. It presents the optimization problem for a given network. The current configuration of the network is $G(t_0 - 1)$ which has been selected and applied at $t_0 - 1$. Since the demands are changed from $t_0 - 1$ to t_0, the network must be re-optimized at t_0. The goal is to select the optimal network configuration at t_0. We neglect the details of the problem, such as the number of links, the number and volume of traffic demands. Instead, from a high perspective, we study the example to depict the advantages of employing traffic prediction in network power optimization. In Fig. 6.4, each node represents a network configuration and its corresponding power consumption (in kW). Each arrow is a transition between two configurations with the required number of changes (including flow reroutings and state transitions).

In this example, three configurations are possible at t_0 (i.e., G_1, G_2, and G_3), and the goal is to select the optimal configuration between them. The result of optimization depends on the number of future traffic steps that are considered in the optimization. Let us look at three different scenarios of this example. The only difference between those scenarios is the input data of the optimization process. Those three scenarios are illustrated in Fig. 6.4. In the first scenario, the traffic prediction is not involved, and only the traffic observation is provided as input. It means $I(t_0)$ is the only available data, and there is no traffic prediction. This scenario is similar to the existing ALR approaches (because they do not use traffic forecasts). The input data in the second scenario consists of traffic data at t_0 and also one-step-ahead predicted values at $t_0 + 1$ (so the available data are $I(t_0)$ and $I(t_0 + 1)$). Based on this input, we can extract the possible network configuration at t_0 and $t_0 + 1$. The third scenario includes the predicted traffic values at $t_0 + 1$ and $t_0 + 2$ as well as the traffic data at t_0 (so $I(t_0)$, $I(t_0 + 1)$, and $I(t_0 + 2)$ are given). In this case, we can derive possible network configurations in the next two steps ahead. We analyze these three cases:

- Case 1 ($T = 0$): the only available information is the current traffic load of each traffic flow ($I(t_0)$). The presumption, in this case, is that the predicted values are not available. In this case, G_1 can be an optimal solution (among possible configurations at t_0) because it reduces the power consumption (from $15kW$ to $10kW$), and needs 15 modifications to $G(t_0 - 1)$ which is not expensive compared to other options.

- Case 2 ($T = 1$): in this case, $I(t_0)$ and $I(t_0 + 1)$ are given. It means the current traffic values and the one-step-ahead predicted values are employed. Based on this information, the network configurations at t_0 and $t_0 + 1$ are extracted. As shown in Fig. 6.4, G_1 is not the optimal solution in Case 2 because (according to $I(t_0 + 1)$) it leads to either a high power consumption state or a big number of reconfigurations in the next time-slot. In other words, G_1 leads to G_4 and G_5, which result in high consumption and big number of changes respectively. Instead, G_2 is a proper choice as it can be switched to G_5 by 10 changes. We can compare the following chains:

  - Chain 1: $G(t_0 - 1) \rightarrow G_1 \rightarrow G_5$:
    - total energy cost: $15 + 10 + 30 = 55\ kW$,
    - total reconfiguration cost: $15 + 50 = 65$.
  - Chain 2: $G(t_0 - 1) \rightarrow G_2 \rightarrow G_5$:
    - total energy cost: $15 + 15 + 30 = 60\ kW$,
    - total reconfiguration cost: $15 + 10 = 25$.

  While the total energy consumption in chain 2 is slightly more than chain 1, its number of required reconfigurations is significantly less than chain 1. So, it compromises between power use (at t_0) and the number of changes (at $t_0 + 1$).

- Case ($T = 2$): $I(t_0)$, $I(t_0 + 1)$, and $I(t_0 + 2)$ are given (i.e., two steps ahead prediction). According to the volume of demands in next 2 time-slots, both G_1 and G_2 are not appropriate choices because they result in costly moves (in term of consumption or number of changes) at $t_0 + 2$. Instead, configuration G_3 is the optimal solution, and the path shown using grey arrows is the optimal trade-off between energy consumption and the number of modifications.

The example in Fig 6.4 demonstrates a possible situation in network optimization. The power consumption values (for each configuration) in Fig 6.4 correspond to a network with thousands of links (e.g., a datacenter network). However, the number of configurations shown in this example does not match a real scenario. Because, in a real situation, there are thousands of possible configurations in each time-slot, which is not suitable to present the concept. Alternatively, we reduced the specifics in our simple illustration to concentrate on the main idea. We use this example to explain the objective function.

RAP makes a balance between the energy consumption and number of changes. The *energy cost function* is defined as the normalized energy consumption:

$$E\left(G(t)\right) = \frac{1}{E_{max}} \sum_{l \in \mathcal{L}} \sum_{s \in \mathcal{S}} v_{s,l}(t) \cdot e_s, \tag{6.2}$$

where $G(t)$ determines the link states and the routes allocated to the flows:

$$G(t) = \{a_l(t), r_k(t)\}, \tag{6.3}$$

and $v_{l,s}(t)$ equals 1 if link l is in state s at t:

$$v_{l,s}(t) = \begin{cases} 1, & \text{if } a_l(t) = s \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

The maximum consumption (E_{max}) happens when all the interfaces are working with maximum capacity (C_{max}). The reconfiguration from $G(t-1)$ to $G(t)$ includes the flow reroutings and the link state transitions. The number of traffic flow reroutings is computed as:

$$R^{'}\left(G(t-1), G(t)\right) = \frac{1}{2} \sum_{k \in \mathcal{K}} \sum_{p \in P_k} \left| w_{p,k}(t) - w_{p,k}(t-1) \right|, \tag{6.5}$$

The number of link state transitions is:

$$R^{''}\left(G(t-1),G(t)\right)=\frac{1}{2}\sum_{l\in\mathcal{L}}\sum_{s\in\mathcal{S}}\left|v_{l,s}(t)-v_{l,s}(t-1)\right|, \tag{6.6}$$

where $w_{p,k}(t)$ equals 1 if flow k is routed on $p \in P_k$ at t:

$$w_{p,k}(t)=\begin{cases}1, & \text{if } r_k(t)=p\\ 0, & \text{otherwise.}\end{cases} \tag{6.7}$$

A single traffic rerouting creates two changes to w. Therefore, in Equation (6.5), the sum of absolute differences is divided by two. The same reason applies to Equation (6.6).

The maximum value for $R^{'}$ is K and the maximum value for $R^{''}$ is L. It means there are K flows that can be rerouted, and L links that their rate can be changed in each reconfiguration. We define the *reconfiguration cost function* as:

$$\begin{aligned}R\left(G(t-1),G(t)\right) \;=\; & \frac{R^{'}\left(G(t-1),G(t)\right)}{2\,K}\;+\\ & \frac{R^{''}\left(G(t-1),G(t)\right)}{2\,L},\end{aligned} \tag{6.8}$$

in which $R^{'}$ and $R^{''}$ are divided by $2K$ and $2L$ to keep the cost function in range of $[0, 1]$ (same as energy cost E).

The cornerstone of the objective function in RAP is a concept called the *chain of configurations*. The set of consecutive configurations that are chosen for the time-slots $\mathcal{T}$ form a chain of configurations (denoted as $\bar{G}$). For example, $\{\bar{G}(t_0)=G_2,\ \bar{G}(t_1)=G_5,\ \bar{G}(t_2)=G_9\}$ is a possible chain in Fig. 6.4. We define the cost of $\bar{G}$ as:

$$q(\bar{G})=\sum_{t=t_0}^{t_0+T}\left(\beta\cdot E(\bar{G}(t))\;+\;(1-\beta)\cdot R(\bar{G}(t-1),\bar{G}(t))\right). \tag{6.9}$$

Cost function $q(\bar{G})$ is the weighted sum of two *normalized* terms: the energy and the reconfiguration cost functions. Coefficient β allows managing the weighted sum based on the importance of each term. The cost function considers only the energy consumption when $\beta = 1$, and it reflects only the number of changes when $\beta = 0$. A trade-off between these two terms is achieved when $0 < \beta < 1$. Consider the set of *all possible chains of configurations* that G is their first configuration (at $t = t_0$):

$$\mathcal{G}_G = \left\{ \bar{G} \mid \bar{G}(t_0) = G \right\}. \tag{6.10}$$

For example, there are 6 possible chains of configurations started from G_3 in Fig. 6.4:

$$\begin{aligned} \mathcal{G}_{G_3} \quad = \quad & \{ \{G_3,\ G_5,\ G_9\ \},\ \{G_3, G_5, G_{10}\}, \\ & \{G_3,\ G_5,\ G_{11}\},\ \{G_3,\ G_6,\ G_{11}\}, \\ & \{G_3,\ G_7,\ G_{11}\},\ \{G_3,\ G_7,\ G_{12}\}\ \}. \end{aligned}$$

The *cost of choosing configuration* $G(t_0)$ *at* t_0 is defined as the cost of the best chain of configurations in $\mathcal{G}_{G(t_0)}$:

$$Q(G(t_0)) = min\left(\{ q(\bar{G}) \mid \bar{G} \in \mathcal{G}_{G(t_0)} \} \right). \tag{6.11}$$

For example, in Fig. 6.4, the cost of choosing G_3 at t_0 is equal to the cost of chain $\{G_3, G_7, G_{11}\}$ because it is the chain with minimum cost in $\mathcal{G}_{G_3}$. Function $Q(G(.))$ is the *objective function of RAP* which makes a trade-off between the energy and reconfiguration costs. The goal is to find the configuration $G(t_0)$ which *minimizes* $Q(G(t_0))$.

6.2.4.2 Constraints

The search space consists of the valid chains of configurations. We define $\bar{G}$ as a valid chain of configurations if all the configurations in $\bar{G}$ are valid configurations. $G(t)$ is a valid configuration

if it satisfies three conditions: (i) each link operates only in one state during a time-slot:

$$\sum_{s \in \mathcal{S}} v_{l,s}(t) = 1, \ \forall l \in \mathcal{L}, \tag{6.12}$$

(ii) there is no unallocated flow:

$$\sum_{p \in P_k} w_{p,k}(t) = 1, \ \forall k \in \mathcal{K}, \tag{6.13}$$

and (iii) capacity of each link is more than the total bandwidth demand of flows that pass through the link:

$$u_l(t) \leq \left(C_{thr} \cdot \textstyle\sum_{s \in \mathcal{S}} (v_{l,s}(t) \cdot c_s) \right), \ \forall l \in \mathcal{L}, \tag{6.14}$$

where $\sum_{s \in \mathcal{S}} (v_{l,s}(t) \cdot c_s)$ determines the capacity of link (based on the link state), and C_{thr} is a value in range $[0, 1]$. It is used to make sure the link is not overloaded. For example, when $C_{thr} = 0.6$, only 60% of link capacity is considered in the flow allocation to prevent the congestion and QoS degradation during traffic bursts. The traffic load $u_l(t)$ on link l at t is the sum of the bandwidth of the flows that are routed on l:

$$u_l(t) = \sum_{k \in \mathcal{K}} \sum_{\forall p : l \in p} w_{p,k}(t) \cdot d_k(t). \tag{6.15}$$

The optimization model of RAP (the objective function and constraints) is summarized as follows.

$$\text{minimize} \quad Q(G(t_0)) \tag{6.16}$$

where :

$$Q(G(t_0)) = min\left(\{q(\bar{G}) \mid \bar{G} \in \mathcal{G}_{G(t_0)}\}\right) \tag{6.17}$$

$$q(\bar{G}) = \sum_{t\in\mathcal{T}} \beta \cdot E(\bar{G}(t)) + \sum_{t\in\mathcal{T}} (1-\beta) \cdot R(\bar{G}(t-1), \bar{G}(t)) \tag{6.18}$$

subject to :

$$\sum_{p\in P_k} w_{p,k}(t) = 1, \ \forall t \in \mathcal{T}, \ \forall k \in \mathcal{K} \tag{6.19}$$

$$\sum_{s\in\mathcal{S}} v_{l,s}(t) = 1, \ \forall t \in \mathcal{T}, \ \forall l \in \mathcal{L} \tag{6.20}$$

$$u_l(t) \leq \left(C_{thr} \cdot \textstyle\sum_{s\in\mathcal{S}}(v_{l,s}(t) \cdot c_s)\right), \ \forall t \in \mathcal{T}, \forall l \in \mathcal{L}. \tag{6.21}$$

6.2.5 Challenges

There are two challenges in this approach. The first challenge is the traffic prediction which has a crucial role in RAP. The flows time-series in the intuitive example of Fig. 6.3 are predictable because they have linear pattern. However, in a real network environment, traffic prediction is a difficult task since traffic time-series consist of complex patterns. To address this issue, we exploited our comprehensive studies on traffic modeling and prediction which resulted in single-step-ahead (Bayati et al., 2016) and multiple-step-ahead (Bayati et al., 2018) traffic predictors. These approaches can handle different traffic characteristics including periodicity, long/short-range dependency, and multiscale behavior.

The second challenge is the time complexity of the optimization model. The optimization problem for the traditional ALR-based EAR NP-hard (Andrews et al., 2012). It has been solved using different techniques including a greedy algorithm in (Tang et al., 2012), and a Genetic Algorithm in (Zemmouri et al., 2016). The complexity of RAP is more than traditional ALR-based EAR because it considers the current and future time-slots. Its computational requirement increases exponentially by the rise in the number of precalculated paths, the number of traffic flows, the number of links and link states, and length of prediction horizon T. In the example of Fig. 6.4, there are at most three possible configurations for the next move of any G_i. However, in a real problem, the number of possible moves (for any configuration) is in order of $N^K \cdot S^L$. Therefore, the total number possible chains of configurations is in order of $(N^K \cdot S^L)^T$. A precise method is required to find the best solution in such a large search space. In this work, we designed a heuristic algorithm based on simulated annealing (SA) that solves RAP in a reasonable time. The proposed heuristic algorithm avoids the temporary reconfigurations to decrease the search space and to maintain the QoS.

6.3 System Overview

This section explains our proposed framework RAP shown in Fig. 6.5. Traffic is monitored using the SDN controller. After performing the flow analysis on the observed traffic data, feature vectors are extracted and stored in a database. The historical data in the database is used for initializing and training the predictor which estimates the future steps of traffic given a new sample. The outcome of the predictor is used in the optimization to find the optimal paths for flows.

6.3.1 Traffic Sampling and Flow Analysis

The traffic sampling and the flow analysis modules monitor and collect the traffic samples for each flow. The definition of traffic flow depends on the type of network. For example, in an IP network, all the packets that have the same protocol type, source/destination IP, and source/destination port belong to the same traffic flow. In the networks with a massive number

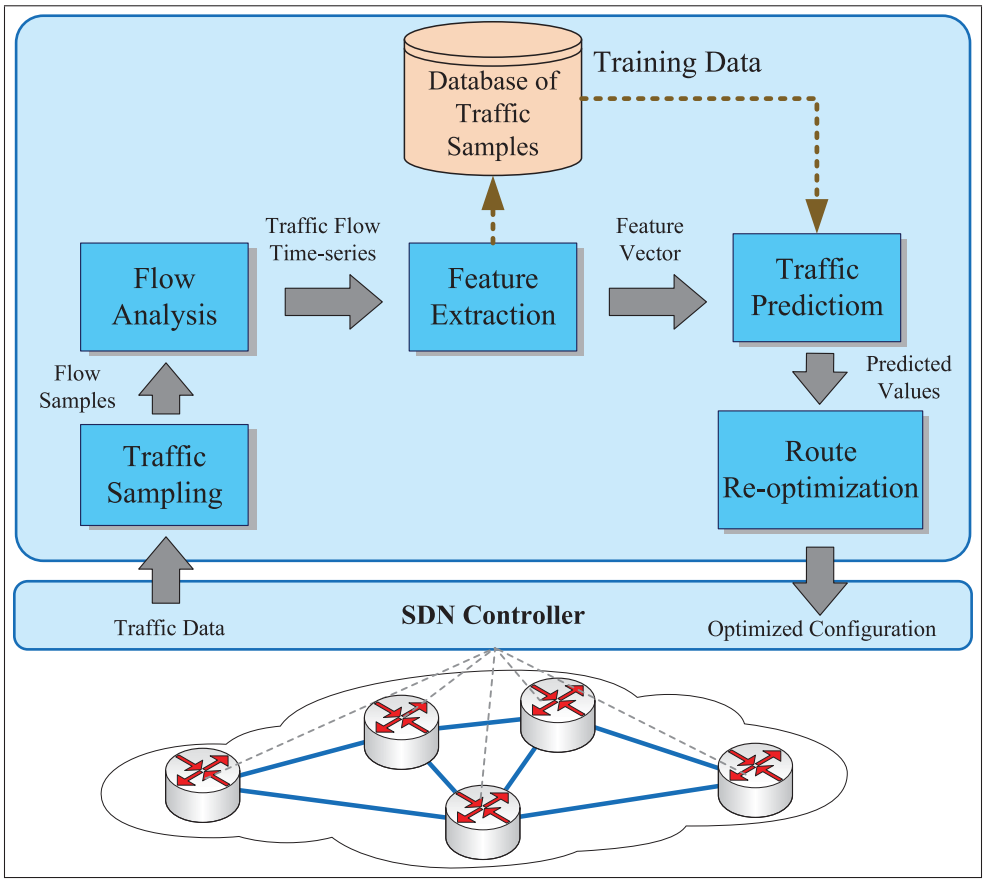


Figure 6.5 The proposed Framework for ALR-based EAR

of traffic flows, it is not applicable to track and monitor all the flows. Instead, to reduce the monitoring and prediction overhead, it is possible to observe only the big flows. It is well-known that a large portion of traffic (e.g., 90% of traffic load) is carried by a small number of flows (e.g., 2% of flows) called big flows (or elephant flows). In (Y. Li et al., 2016), prediction of elephant flows has been employed to manage inter-data-center traffic. The energy consumption can be optimized by managing the big flows inside the network while the small flows (or mice flows) can be routed using a general-purpose routing algorithm (such as a shortest-path algorithm). Thus, there is no need to monitor and predict the small flows. The flow analysis has to provide the information of the traffic flows on both directions (upstream and downstream).

6.3.2 Feature Extraction and Traffic Prediction

The goal of this part is to train a prediction algorithm with horizon T. Prediction horizon T is defined as the number of future time-slots that are predicted by the algorithm. First, we need to extract the features of traffic samples and create a training set. Consider $\bar{d}(t) = (d(t), d^{'}(t))$ as the bandwidth utilization of flow at t where $d(t)$ and $d^{'}(t)$ are upstream and downstream values respectively. A traffic sample (x_t, y_t) is composed of feature vector x_t, and multivariate label y_t. The length of the feature vector is m while $m/2$ samples are taken from upstream and $m/2$ samples are from downstream:

$$\begin{aligned} x_t \;=\; & [d(t-1),\ d(t-2),\ \ldots,\ d(t-m/2), \\ & d^{'}(t-1),\ d^{'}(t-2),\ \ldots,\ d^{'}(t-m/2)]. \end{aligned} \tag{6.22}$$

Since there is a relation between upstream and downstream demand, both are used in the feature vector to increase the prediction accuracy. The length of multivariate label y_t is equal to the prediction horizon T. It includes future values of flows:

$$y_t = [d(t),\ d(t+1),\ \ldots,\ d(t+T-1)]. \tag{6.23}$$

Fig. 6.6 illustrates the feature extraction for upstream traffic at $t = 53$. The bit-rates of $d(t)$ and $d^{'}(t)$ are shown in Fig. 6.6a. Fig. 6.6b presents traffic sampling at the time-scale of 1 minute. Length of feature vector x_{53} is 10 while 5 samples are from $d(t)$, and 5 samples are from $d^{'}(t)$. Length of multivariate label $y_{53} = [d(53),\ d(54),\ d(55)]$ is equal to $T = 3$. The samples that are generated in this examples are utilized for 3-step-ahead prediction of upstream demand. The samples for downstream prediction have the same feature vector. Unless their labels include the future steps of $d^{'}(t)$.

Database $\mathcal{D} = \{(x_{t_i}, y_{t_i}) \mid i = 1, 2, \ldots, N\}$ includes N historical traffic samples from different flows. $\mathcal{D}$ is used to train a supervised machine learning algorithm f. At time t, predictor f uses

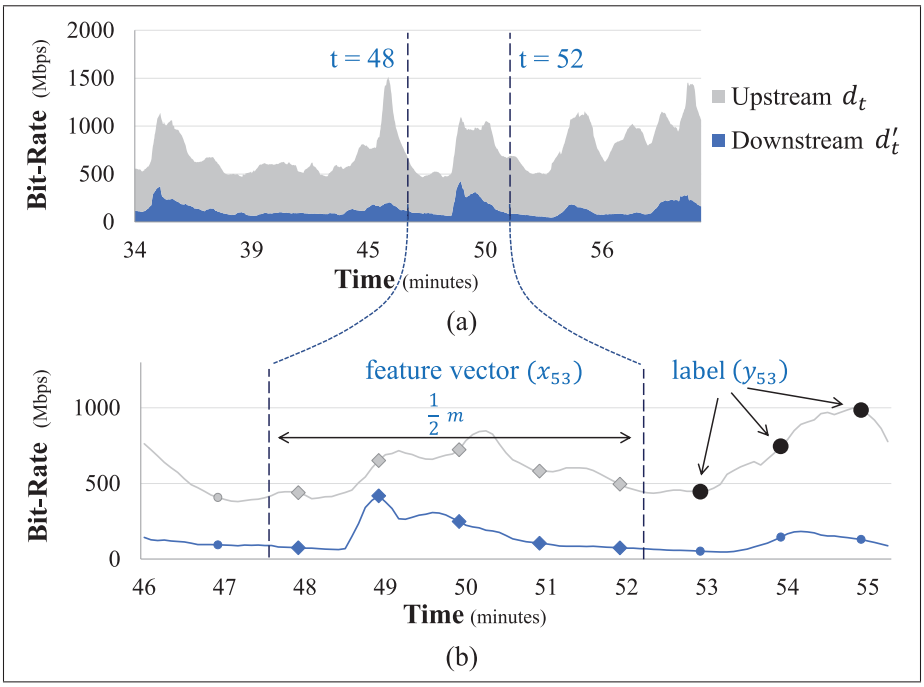


Figure 6.6 Traffic Sampling and Feature Extraction

x_t as input to predict $y_t^{'}$:

$$f(x_t) = y_t^{'}\,, \tag{6.24}$$

$$y_t^{'} = \left[d(t),\ d(t+1),\ \ldots,\ d(t+T-1)\right] + \varepsilon_t\,, \tag{6.25}$$

where $y_t^{'}$ is the estimation for y_t, and ε_t is the prediction error (a vector with length T). The proposed framework for the ALR-based EAR (in Fig. 6.5) is not restricted to a particular prediction algorithm, and it is possible to employ any predictor including ARIMA, FARIMA, LSTM, etc.

We employed our multiple-step-ahead traffic predictor proposed in (Bayati et al., 2018). Unlike other time-series predictors, it has been designed based on traffic characteristics. The key idea behind this algorithm is the multiscale behavior of traffic. It employed traffic information from different time-scales to reduce error propagation. The workflow model of our algorithm is presented in Fig. 6.7. It consists of two steps. There are H GPR experts in Step 1 which predict traffic at different time-scales. In Step 2, the outcomes of experts f^{h+1} and f^{h-1} are merged

and used to predict h-th step ahead value of traffic ($d(t_0 + h)$). In (Bayati et al., 2018), we proved this structure exploits the traffic characteristics (multi-scale behavior, and long/short range dependency) to control the error propagation.

6.3.3 Route Optimization

The optimization module selects the optimal routes for the traffic flows to reduce network energy consumption and to minimize future reconfigurations in the network. It solves the RAP problem iteratively using the output of the traffic prediction module. Since the RAP is an NP-hard problem, the route optimization module exploits a simulated annealing (SA) heuristic algorithm to efficiently solve it. The SA heuristic algorithm has been designed to avoid the solutions which consist of temporary reconfigurations.

Simulated annealing (P. Zhang et al., 2021) has been motivated by the physical process of heating a material and then slowly lowering the temperature to minimize the system energy. It is a popular algorithm for finding the global optimum in a large search space with many local optimums. SA accepts solutions that are worse than the current solution with some probability to escape local optimums during the search process. The probability of accepting a worse solution lowers with a control parameter called *temperature*. SA explores the search space when the temperature is high, and it converges as the temperature is decreasing. Also, the probability of accepting a worse solution has a direct relationship with the difference between the objectives of the current and new solution. SA needs four parameters: starting temperature (A_{max}), ending temperature (A_{min}), temperature decrement (A_d), and the number of iterations at each temperature (I_{max}). Generally, A_{max} is set to 1, and A_{min} is set to a value close to zero (e.g., 0.000001).

Algorithm 6.1 presents the proposed SA-based method. It finds the optimal chain of configuration $\bar{G}_{opt}$ according to Equation (6.11). The output of algorithm is the first configuration of the optimal chain $\bar{G}_{opt}$. Temperature is initially set to A_{max} and decreased to reach A_{min}. SA starts from an initial chain of configurations ($\bar{G}_{start}$) and generates a random neighbor ($\bar{G}_{new}$) of the

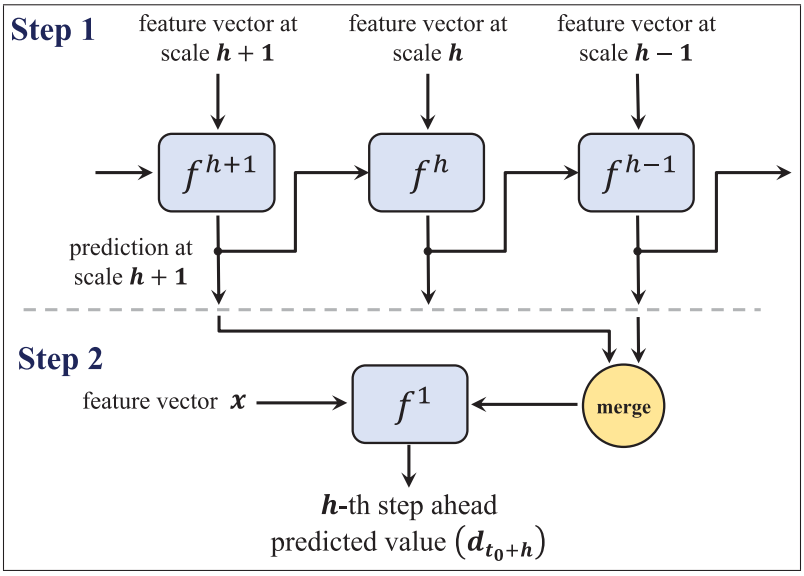


Figure 6.7 Multiple-step-ahead traffic prediction algorithm

current chain ($\bar{G}$) in each iteration. The generated solution is accepted (and becomes the current solution in the algorithm) with probability $Pr(O, O', A)$:

$$Pr(O, O', A) = exp\left(-\frac{O' - O}{A}\right) , \tag{6.26}$$

where A is the current temperature and O and O' are respectively the costs of current and generated solutions. The cost function $q(\bar{G})$ is calculated using Equation (6.9). Note that the new solution is accepted if it has a lower cost compared to the current solution (because $Pr(O, O', A)$ takes a value more than 1). The initial point ($\bar{G}_{start}$) is a solution in which the current configuration at $t_0 - 1$ remains unchanged during $\mathcal{T}$:

$$\bar{G}_{start}(t) = G(t_0 - 1), \ \forall t \in \mathcal{T} . \tag{6.27}$$

In Algorithm 6.1, the boolean function $Valid(\bar{G})$ at line 6 returns $True$ if all the configurations in $\bar{G}$ satisfy the conditions in Equations (6.12), (6.13), and (6.14). In other words, it tests whether $\bar{G}$ is a valid chain. Function $GenerateRandomNumber$ (Algorithm 6.1, line 11) returns a random value between 0 and 1.

Algorithm 6.1 Simulated Annealing Heuristic for RAP

Input: A_{min}, A_{max}, A_d, I_{max}
Output: $\bar{G}_{opt}(t_0)$
$A = A_{max}$, $\bar{G} = \bar{G}_{start}$, $O_{opt} = \infty$, $i = 1$
$O = q(\bar{G})$
while $A \geq A_{min}$ **do**
  if $Valid(\bar{G})$ *and* $O < O_{opt}$ **then**
    $\bar{G}_{opt} = \bar{G}$, $O_{opt} = O$
  end
  $\bar{G}_{new} = GenerateNeighborCOC(\bar{G})$
  $O^{'} = q(\bar{G}_{new})$
  $r = GenerateRandomNumber()$
  $i = i + 1$
  if $Pr(O, O^{'}, A) \geq r$ **then**
    $\bar{G} = \bar{G}_{new}$, $O = O^{'}$
  end
  if $i \geq I_{max}$ **then**
    $A = A_d \cdot A$, $i = 1$
  end
end
return $\bar{G}_{opt}(t_0)$

Function $GenerateNeighborCOC$ generates and returns a random chain of configurations which is a neighbor to $\bar{G}$. It avoids the neighbor solutions in which a path for a flow or the state of a link changed more than 1 time in $\mathcal{T}$. It aims to reduce the probability of selecting a neighbor solution which includes temporary reconfigurations. This strategy (for exploring the search space) improves the time-complexity of the SA algorithm and reduces the temporary changes in the final solution. Algorithm 6.2 shows the steps in $GenerateNeighborCOC$. In each run, it takes a chain $\bar{G}_{in}$ and performs a modification in it which can be either a random change in $a_l(t)$ (for link states) or in $r_k(t)$ (for paths). The type of change depends on z which is initialized by $RandomChoice$ (Function $RandomChoice$ returns a randomly selected element of a given set). A random change in the paths is performed when $z = 1$. A flow $k^{'}$ is selected from the set $\mathcal{K}^{'}$, and $k^{'}$ is removed from $\mathcal{K}^{'}$ (function Pop removes an element from a set). The chance of selecting flow $k^{'}$ depends on the number of changes in its path within the chain $\bar{G}_{in}$. The number

Algorithm 6.2 *GenerateNeighborCOC*

```
1  Input: $\bar{G}_{in}$
2  Output: $\bar{G}_{out}$
3  $t = RandomChoice(\mathcal{T})$
4  $z = RandomChoice(\{1, 2\})$
5  if $z = 1$ then
6      $\mathcal{K}' = \mathcal{K},\ k = Null$
7      while $k = Null$ do
8          $r = GenerateRandomNumber()$
9          $k' = RandomChoice(\mathcal{K}')$
10         $\mathcal{K}' = Pop(\mathcal{K}',\ k')$
11         $n = CountReroutings(\bar{G}_{in},\ k')$
12         if $Pr_s(n) \geq r$ then
13             $k = k'$
14         end
15     end
16     $\bar{G}_{out} = ChangePath(\bar{G}_{in},\ k,\ t)$
17 else
18     $\mathcal{L}' = \mathcal{L},\ l = Null$
19     while $l = Null$ do
20         $r = GenerateRandomNumber()$
21         $l' = RandomChoice(\mathcal{L}')$
22         $\mathcal{L}' = Pop(\mathcal{L}',\ l')$
23         $n = CountRetransitions(\bar{G}_{in},\ l')$
24         if $Pr_s(n) \geq r$ then
25             $l = l'$
26         end
27     end
28     $\bar{G}_{out} = ChangeLinkState(\bar{G}_{in},\ l,\ t)$
29 end
30 return $\bar{G}_{out}$
```

of changes in the path of flow k' is calculated using function *CountReroutings* (line 11):

$$CountReroutings(k') = \frac{1}{2} \sum_{t \in \mathcal{T}} \sum_{p \in P_k} \mid w_{p,k'}(t) - w_{p,k'}(t-1) \mid \ . \tag{6.28}$$

The probability of choosing flow $k^{'}$ decreases as the value of $CountReroutings(k^{'})$ increases and is calculated as:

$$Pr_s(n) \;=\; exp(-n). \tag{6.29}$$

If there is no change in path of $k^{'}$ in $\bar{G}$ (i.e., $n = 0$), then $k^{'}$ is selected with probability 1 (because $exp(0) = 1$).

A random change in the link states is performed when $z = 2$. The process is the same as the random change in the paths. Function $CountRetransitions(l^{'})$ (Algorithm 6.2, line 23) calculates the number of changes in the state of link $l^{'}$:

$$CountRetransitions(l^{'}) = \frac{1}{2} \sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}} \mid v_{l',s}(t) - v_{l',s}(t-1) \mid \;. \tag{6.30}$$

The proposed SA heuristic can solve large instances of RAP in a reasonable time. In our experiments, we apply the proposed algorithm to solve large network topologies with hundreds of traffic flows.

Table 6.4 Description of the topologies used in the experiments

Topology	**GEANT**	**Fat-Tree**
number of hosts	23	128
number of (bidirectional) links	38	384
number of traffic flows	250	1500

6.4 Summary

We provided an optimization framework for networks with fluctuating traffic demands which is called Rate Adaptation using Prediction (RAP). Traffic prediction has been involved in the proposed approach to proactively control the number of changes and preserve QoS during network re-optimizations. The optimization problem has been formulated and solved using

a simulated annealing heuristic algorithm. Our results confirm the proposed approach can significantly reduce the number of changes and maintain the network performance while it provides energy conservation.

CHAPTER 7

EVALUATION

7.1 Experimental Results

7.1.1 Setup

To validate our proposed solution, we used two well-known topologies in our experiments. The first topology is the GEANT network (pan-European research and education backbone) with 23 nodes and 38 bidirectional links. The second topology is a 3-level fat-tree network (a typical interconnection used in data-centers) with 8 pods containing 128 servers that are connected using 80 switches and 384 bidirectional links. Each link can operate with three different rates: $100Mbps$, $1Gbps$, and $10Gbps$. The energy consumption of the links is calculated using the values in Table 6.2. Table 6.4 highlights the parameters of the topologies. We used these two different topologies to demonstrate our algorithm's value for both small and large networks. GEANT has a small number of nodes and links, and it was the subject of many studies. Fat-tree, on the other hand, is popular in the datacenter network where there are thousands of servers and hundreds of thousands of links.

In our simulation, there are 250 traffic flows in the GEANT topology and 1500 traffic flows in the fat-tree topology. The source and destination of each flow are selected randomly. There are at most 4 precalculated paths (4 first shortest paths) between each pair of nodes in the networks. The network configuration is re-optimized every 5 minutes. We used real-life traffic traces collected from Abilene nodes during 2007-01-01 and 2007-10-14 (*Internet2 Abilene Network*, 2007) to mimic the real situation. The time-scale of the traffic time-series is 5 minutes (each sample is the average of bandwidth utilization in a 5-minutes interval). Selecting a suitable length of time-slot for traffic prediction and network optimization is a trade-off between configuration cost and power consumption. A short time-window can lead to a dynamic network with reduced power consumption. It can also trigger a significant number of changes to the network configuration

in shorter periods. On the other hand, a longer time-window reduces reconfiguration costs (so the network is stable), but the network power consumption may not be minimized by all the moments.

We performed 150 independent trials for each topology (150 rounds of simulations on GEANT, and 150 rounds of simulations on the fat-tree topology). In each trial, we emulated the traffic flows between the source-destination pairs for 12 hours. The range of traffic demands of a flow is between $50Mbps$ to $2Gbps$. We performed 6 different categories of experiments for each topology. In each category of experiment, we imposed a particular traffic load on the network. The average network utilization (i.e., average of links' utilization) in categories 1 to 6 are respectively as follows: 5%, 10%, 15%, 20%, 25%, and 30%. Each category includes 25 trials (i.e., $6 \times 25 = 150$).

We measured the following terms:

- *Energy cost*: it has been measured using Equation (6.2). The maximum energy cost is 1 (when all the interfaces are working with maximum rate $10Gbps$).

- *Average number of link state transitions*: in each time-slot (i.e., every 5 minutes), the number of link state transitions is counted. The maximum number of link state transitions is equal to the number of interfaces. The smaller number of transitions means the network is more stable.

- *Average number of flow reroutings*: in each time-slot, the number of flows that are rerouted is measured. The maximum number of flow reroutings is equal to the number of flows. The number of reroutings must be decreased to improve the QoS.

- *Reconfiguration cost*: we used Equation (6.8) which is a combination of the link state transitions and flow reroutings. The maximum cost equals 1 and corresponds to a time-slot when all the flows are rerouted and all the link states are changed.

- *Average time between link state transitions*: the state of an interface may change several times during the experiments. The time (minutes) between state transitions for each interface is measured. The longer period between rate transitions is preferred.

- *Average time between traffic flow reroutings*: the time between reroutings of each flow is measured. The longer period between reroutings is preferred.

7.1.2 Implementation of Traditional ALR

Our proposed model (RAP) has been compared to the traditional ALR. We use the following model as the traditional ALR which is similar to the models in (Zemmouri et al., 2016) and (Tang et al., 2012):

$$\text{minimize} \quad \sum_{l \in \mathcal{L}} \sum_{s \in \mathcal{S}} e_s v_{l,s}(t) \tag{7.1}$$

subject to :

$$\sum_{p \in P_k} w_{p,k}(t) = 1, \ \forall k \in \mathcal{K} \tag{7.2}$$

$$\sum_{s \in \mathcal{S}} v_{l,s}(t) = 1, \ \forall l \in \mathcal{L} \tag{7.3}$$

$$u_l(t) \leq \left(C_{thr} \cdot \textstyle\sum_{s \in \mathcal{S}} (v_{l,s}(t) \cdot c_s) \right), \ \forall l \in \mathcal{L}. \tag{7.4}$$

Traditional ALR has the same constraints as RAP. The main difference between traditional ALR and RAP is in their objective function. Unlike RAP, the network changes are not considered in the traditional ALR. The objective function of the traditional ALR considers only the energy cost. The traditional ALR has been solved using a greedy algorithm in (Tang et al., 2012), and a heuristic based on genetic algorithm in (Zemmouri et al., 2016). In our approach, we used a simpler version of Algorithm 6.1 to solve the traditional ALR. Our Simulated Annealing (SA) heuristic for traditional ALR is presented in Algorithm 7.1. There are two main differences between SA for RAP and SA for traditional ALR. First, the output of Algorithm 7.1 is a

Algorithm 7.1 Simulated Annealing (SA) Heuristic for traditional ALR

Input: A_{min}, A_{max}, A_d, I_{max}
Output: G_{opt}

$A = A_{max}$, $G = G_{start}$, $O_{opt} = \infty$, $i = 1$
$O = E(G)$
while $A \geq A_{min}$ **do**
    if $Valid(G)$ *and* $O < O_{opt}$ **then**
        $G_{opt} = G$, $O_{opt} = O$
    end
    $k = RandomChoice(\mathcal{K})$
    $G_{new} = ChangePath(G, k)$
    $O^{'} = E(G_{new})$
    $r = GenerateRandomNumber()$
    $i = i + 1$
    if $Pr(O, O^{'}, A) \geq r$ **then**
        $G = G_{new}$, $O = O^{'}$
    end
    if $i \geq I_{max}$ **then**
        $A = A_d \cdot A$, $i = 1$
    end
end
return G_{opt}

configuration while the output of SA for RAP is a chain of configurations. Second, in Algorithm 7.1, function E in Equation (6.2) is used instead of q to calculate the cost function.

7.1.3 Results

Both models (traditional ALR, and our proposed RAP) have been tested on two network topologies (GEANT and fat-tree). First, we present the results of multiple-step-ahead prediction algorithm. Then, we show the results of experiments on the GEANT topology. Finally, we discuss the results achieved on the fat-tree topology.

7.1.3.1 Prediction error

The prediction error of our multiple-step-ahead prediction algorithm has been shown in Figure 7.1. The prediction error is measured using Normalized Mean Squared Error (NMSE) (Bayati et al., 2018). The model has been compared to 3 well-known time-series predictors (ARIMA, FARIMA, and LSTM). Fig. 7.1 includes 8 steps ahead prediction results ($T = 8$) while the time-scale is 5 minutes. The prediction error of different predictors increases with prediction horizon. In the first steps, the algorithms have almost the same performance. The propagation of error in our algorithm (i.e., the Multiscale Predictor) is less than other predictors.

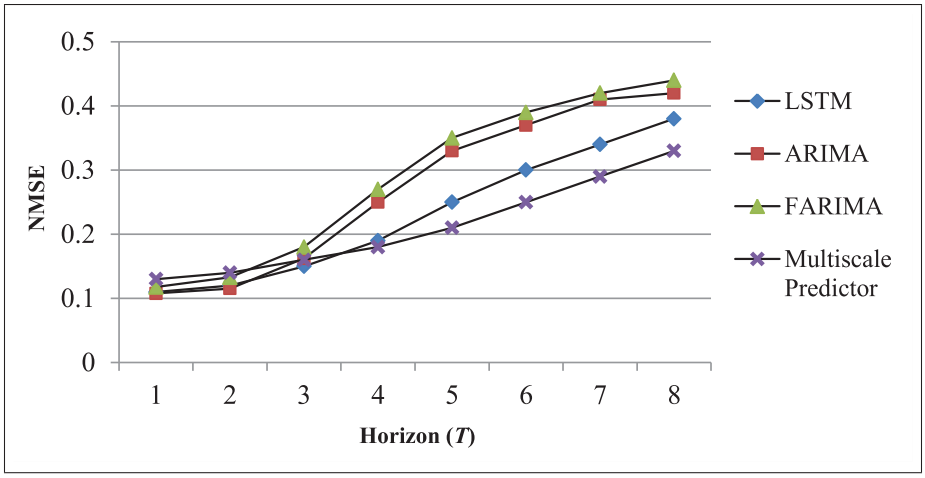


Figure 7.1 Multiple-steps-ahead prediction on the Abilene network traffic data at time-scale of 5 minutes

7.1.3.2 GEANT topology

The results of our simulation on GEANT topology are presented in Fig. 7.2, 7.3, and 7.4. They compare the traditional ALR with three versions of RAP (using three different values of T). The energy cost and reconfiguration cost have been measured for different amounts of network utilization and shown in Fig 7.2. The network energy consumption increases as the load on the network grows. The results in Fig. 7.2a show that energy consumption is lower when the traditional ALR is used to configure the network. However, the difference between energy conservation in traditional ALR and RAP is not significant. On the other hand, the

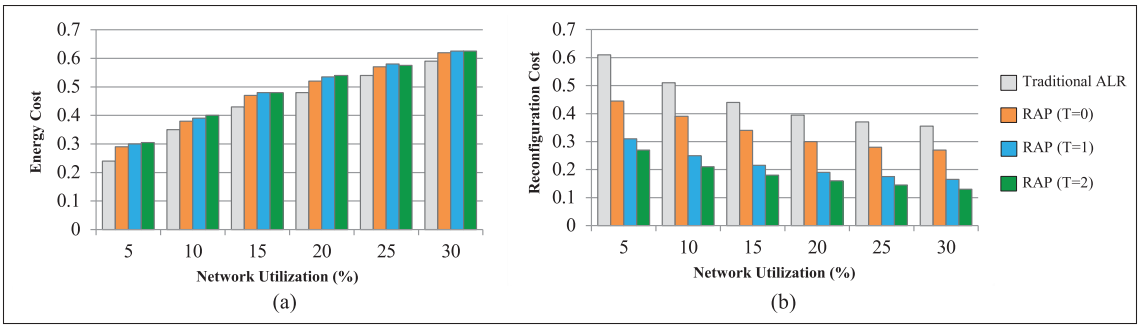


Figure 7.2 Energy and reconfiguration costs (GEANT topology)

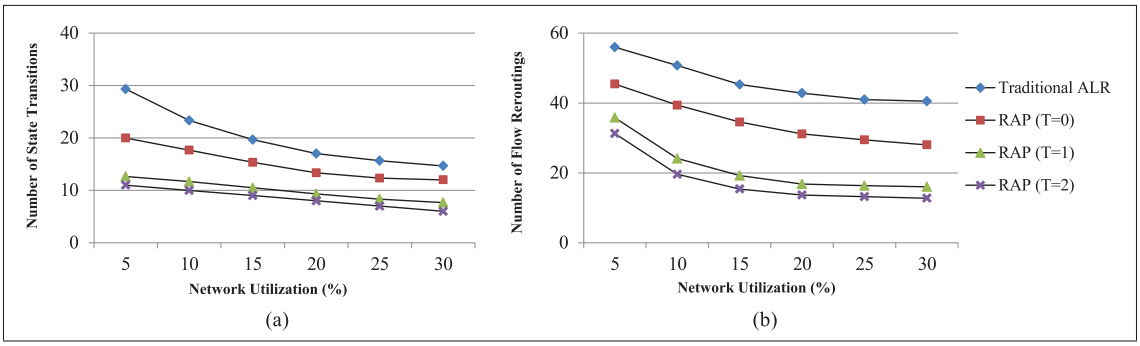
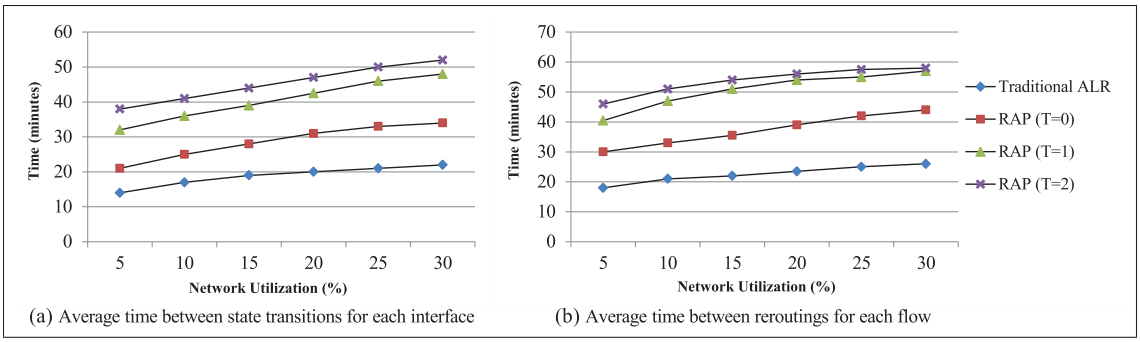


Figure 7.3 Average number of state transitions and average number of flow reroutings in each time-slot (GEANT topology)

network reconfiguration cost is significantly smaller when RAP is used to configure the network compared to traditional ALR. Fig. 7.2b shows that the reconfiguration cost and the network utilization have direct relationship. Also, it shows that RAP reduces the network reconfiguration cost. As the prediction horizon (T) in RAP increases, the reconfiguration cost decreases. Thus, the minimum reconfiguration cost belongs to RAP with $T = 2$.

In our simulation, there are 38 bidirectional (76 unidirectional) links and 250 traffic flows in GEANT topology. Average number of changes (rate transitions and flow reroutings) in each time-slot is shown in Fig. 7.3. The number of changes (in different methods) has an downtrend regarding the network utilization. As shown, RAP reduces the number of rate transitions in GEANT topology (specially when $T = 1$ and $T = 2$). For example, according to Fig. 7.3a, traditional ALR results in 20 state transitions (on average) in each time-slot when network utilization is 15%. In the same situation, RAP ($T = 2$) leads to 9 state transitions. This is also



(a) Average time between state transitions for each interface

(b) Average time between reroutings for each flow

Figure 7.4 Average time between changes in the network configurations (GEANT topology)

true for number of flow reroutings. According to Fig. 7.3b, when the network utilization is 15%, traditional ALR reroutes 45 traffic flows (out of 250) while RAP ($T = 2$) changes the paths of 15 flows in each time-slot.

The average time between state transitions for each link is shown in Fig. 7.4a. Also, the average time between reroutings for each flow is presented in Fig. 7.4b. The time between changes increases as the load on the network increases which means network is more stable when the load increases.

7.1.3.3 Fat-tree topology

The results of our simulation on the fat-tree topology are presented in Fig. 7.5, 7.6, and 7.7. Unlike the GEANT topology, the reconfiguration cost of the fat-tree network (using different algorithms) grows as the traffic load on the network increases (Fig. 7.5b). It means the number of changes increases as the load on the network increases (Fig. 7.6). This observation is also correct for traditional ALR. The explanation is that in the fat-tree topology, the number of links and paths between hosts is much higher compared to GEANT topology. Therefore, in order to reduce power consumption, there are many possible solutions for the optimization model to configure the paths and link states. As the load increases, we need to reconfigure many routes and link-states to reach the optimal configuration.

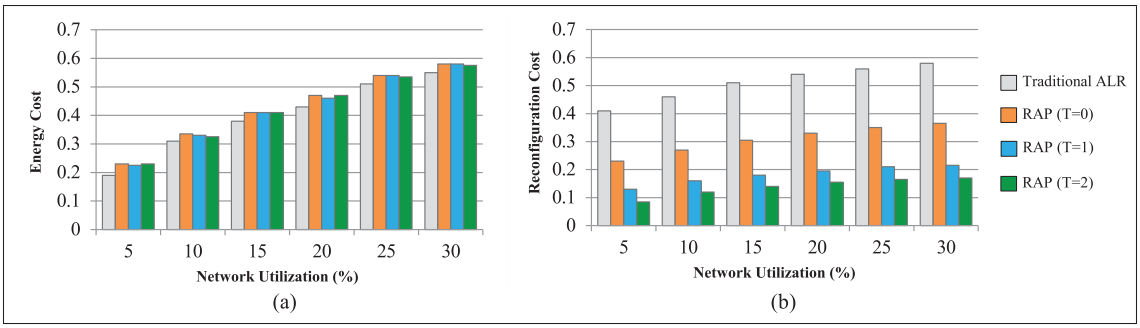


Figure 7.5 Energy and reconfiguration costs (fat-tree topology)

In this simulation, RAP provides almost the same energy cost compared to traditional ALR. However, the difference between the reconfiguration cost of RAP and traditional ALR is notable. For example, consider the number of flow reroutings for network utilization %30 (in Fig. 7.6b). The number of rerouting in RAP ($T = 1$) is %60 less than traditional ALR which is notable. The reconfiguration cost of the traditional ALR is high because it is designed to determine the optimal configuration for each instance of the problem. RAP does consider both costs, though, and find a balance between them. This RAP feature makes this framework a perfect approach for actual scenarios.

7.1.3.4 Time-complexity

The model has been implemented using Python programming language. The simulation has been done a computer with a Core-i7 CPU (3.4GHz with 8 cores). The required time for SA algorithm can be tuned using parameter I_{max}. For the GEANT topology, the I_{max} has been set to 100, and for the fat-tree topology, it has been set to 300. The SA algorithm solves the problems in reasonable time. On average, SA needs 14 seconds to solve an instance of GEANT, and 38 seconds to solve an instance of the fat-tree topology.

7.1.3.5 Prediction horizon

In the results of our experiments on the GEANT and fat-tree network, the performance of RAP (for finding stable configurations) improves as the prediction horizon increases. In both

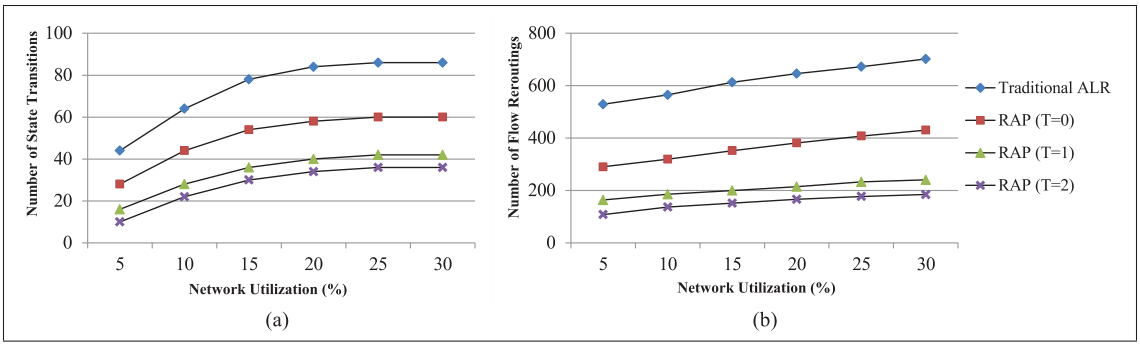


Figure 7.6 Average number of state transitions and average number of flow reroutings in each time-slot (fat-tree topology)

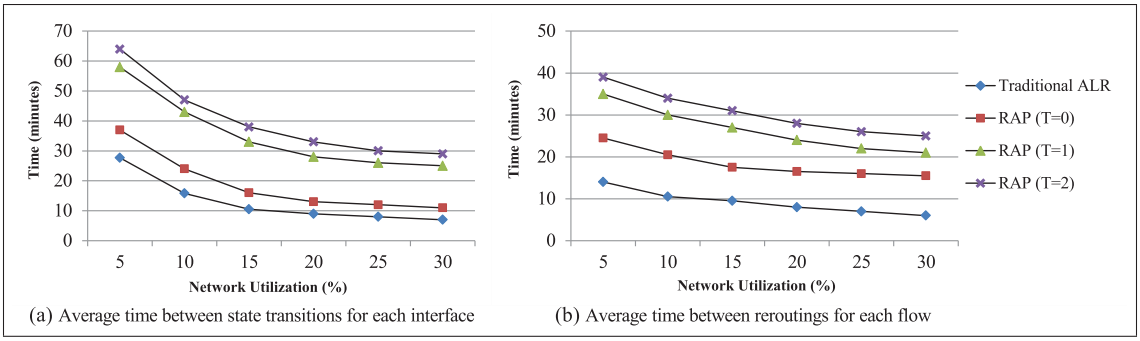


Figure 7.7 Average time between changes in the network configurations (fat-tree topology)

topologies, the RAP with $T = 2$ outperforms RAP with $T = 1$. We analyzed the performance of RAP with different prediction horizons to find the optimal value for T. In this part of experiment, we used the fat-tree topology when there are 1500 traffic flows in the network and the network utilization is 15%. We compared the results of RAP with different values of T.

The results are illustrated in Fig. 7.8. The reconfiguration cost is decreasing from $T = 0$ to $T = 4$. From $T = 5$, the reconfiguration cost increases. The main reason for this instability for higher values of T is the accumulation of prediction errors in the input of RAP (d). The results in Fig. 7.8 proves that the prediction horizon in RAP should be selected according to the prediction error.

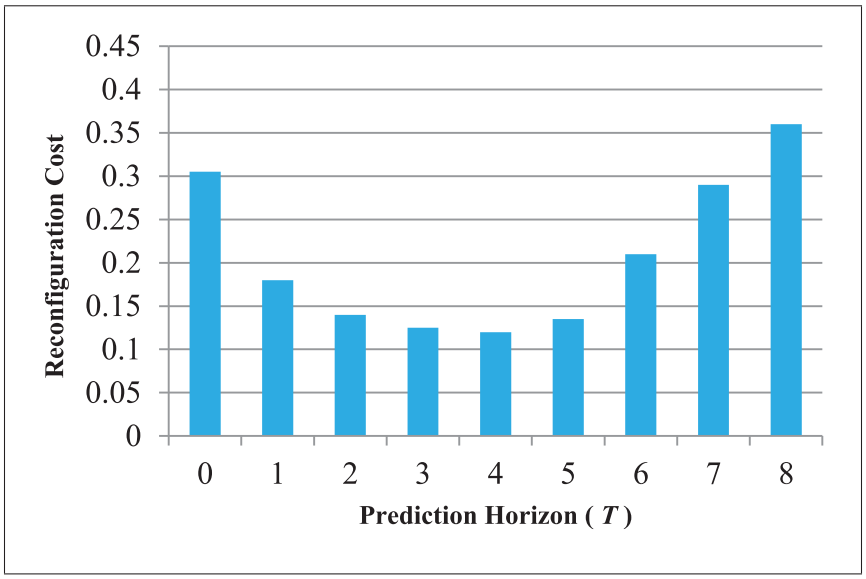


Figure 7.8 Network reconfiguration cost using RAP with different values of T

7.2 Summary

The RAP optimization model has been evaluated in this chapter using real traffic data and well-known network topologies. Using new technologies (e.g., ALR-based adapters) and EAR, we reduced energy costs in a dynamic traffic network. The outcome shows that RAP provides energy-saving, and it avoids network instability during repeated optimization. RAP considers two different objectives: energy cost and changes in the network. Combining these two objectives allows RAP to compromise between energy consumption and network reconfiguration. The RAP's key benefit among existing energy efficiency models is the ability to operate in networks with dynamic traffic and maintaining network stability during repeated optimizations. RAP compromises the power saving to guarantee network stability. Our experiments show that RAP is comparable to other approaches in terms of energy saving. Moreover, its constraints on network stability distinguish RAP from existing approaches. This outcome is essential for real-life networks where traffic demands vary over time. To be exact, RAP achieves almost the same power saving compared to existing ALR solutions while it eliminates more than 65% of the network instability (issued by traditional solutions), which is a notable gain. This significant

accomplishment is achieved by integrating our accurate prediction algorithm (explicitly designed for traffic data) into energy efficiency models, and it paves the way for our proposed framework to be utilized in production networks, especially in data center networks.

CONCLUSION AND FUTURE WORK

Energy consumption is a crucial problem in modern networks (e.g. DCNs) that have usually high-capacity and are tuned for maximum performance (peak network load or busy-hour), making them highly power-hungry. Increased energy use would undoubtedly result in a high carbon footprint and more greenhouse gas emissions, which are the major contributors to global warming. In response to this issue, numerous proposals for energy efficiency at various levels of the DCN have been studied and proposed. In addition to developments in energy-efficient server architecture, other studies have looked at ways to dynamically track DC network traffic loads to align them and maximize DCN power consumption.

In this research, we proposed a new optimization model that is based on the predicted values from a traffic model. It is designed to optimize a network with dynamic traffic, such as data center networks. Our approach is divided into two main parts. The first part is traffic modelling and prediction, and the second part is the energy optimization model. In traffic modelling, we used Gaussian Process Regression (GPR) as the basis of our prediction algorithm. We studied traffic characteristics and integrated them into different layers of our model. The energy optimization model is based on ALR technology. It is an EAR model that takes the predicted traffic values and optimizes the network according to the current and predicted traffic load. Its goal is to find a network configuration that minimizes traffic for the current traffic load and avoids massive network changes for optimizations in future time-slots.

This strategy has two difficulties. The first challenge is traffic prediction, which has a central role to play in RAP. In a real network environment, traffic prediction is difficult because traffic time-series consist of complicated patterns. We used our extensive traffic modelling and forecasting studies to resolve this problem, which resulted in single-step and multi-step traffic predictors.

The second problem is the time complexity of the optimization model. The problem of optimization for the traditional ALR-based EAR is NP-hard. It has been overcome using a number of methods, including greedy algorithms and genetic algorithms. RAP's sophistication

is more than the traditional ALR-based EAR because it views the present and future time slots. Its computational requirement is exponentially increased by the increase in the number of pre-calculated routes, the number of traffic flows, the number of link and link states, and the length of the predicted horizon. A precise method is required to find the best solution in such a large search space. In this work, we designed a heuristic algorithm based on simulated annealing (SA) that solves RAP in a reasonable time. The proposed heuristic algorithm avoids the temporary reconfigurations to decrease the search space and to maintain the QoS.

This method can be used in a network with a centralized controller (e.g., SDN). The input of this model is the traffic loads on the network. The final output is the commands for changing the routes and switching the link states. The network controller is able to monitor and provide the required inputs, and it can force the reconfiguration commands to the network. The algorithm is designed for networks with ALR-enabled links. It can also be applied to the network with regular links. However, we must consider that there are only two states for regular links (i.e., on/off). The case of standard interfaces has not been studied in this thesis. To apply this model to the network with regular links, we need to study the switch delay between on and off states.

In our research, we investigated the application of machine learning and prediction algorithms in the optimization models. We designed a novel traffic model and integrated that into the optimization model. It leads to an energy efficiency solution that adapts the network to the traffic load. Compared to the solutions designed for fixed traffic demands, our solution considers realistic scenarios where the network carries dynamic traffic. This is one step forward in addressing the network energy efficiency problem.

8.1 Future Works

Our thesis's proposed framework consists of two main modules: the traffic prediction module and the network optimization module. These two modules work together while the prediction modules feed the optimization module to make proactive decisions. There are other approaches like reinforcement learning (RL), where those two separate parts (prediction and optimization)

are combined into one module. Reinforcement learning techniques is the subfield of machine learning concerned with decision making and action control. It studies how an agent can learn to achieve goals in a complex, uncertain environment. An RL agent observes previous environment states and rewards and then decides an action to maximize the reward. RL has achieved good results in many challenging environments in recent years with advances in deep neural networks (DNN), making few assumptions about their environments and thus can be generalized in other settings.

Inspired by these results, we are motivated to enable deep reinforcement learning (DRL) for energy efficiency problems. The problem of network energy efficiency has the elements of RL's challenges: the network environment is complex and uncertain, and the network controller decisions trigger the actions to change the environment, which may lead to a reward or penalty according to the results of decisions. This approach leads to a proactive optimization model that combines the two separate modules (prediction and optimization) using the powerful RL techniques. As an advantage, the RL-based method is an online solution, which adapts to the network traffic fluctuations over time.

8.2 List of Publications

List of journal publications:

- **A. Bayati**, C. Pham, K. Nguyen, M. Cheriet, "Reducing Network Energy Consumption Using Adaptive Link Rate and Traffic Prediction," IEEE Transactions on Sustainable Computing, 2021 (undergoing review);

- **A. Bayati**, K. Nguyen, M. Cheriet, "Gaussian Process Regression Ensemble Model for Network Traffic Prediction," IEEE Access, 2020 (Bayati et al., 2020);

- **A. Bayati**, K. Nguyen, M. Cheriet, "Multiple-Step-Ahead Traffic Prediction in High-Speed Networks," IEEE Communications Letters, October 2018 (Bayati et al., 2018);

- M. Si Saber, M. Ghorbani, **A. Bayati**, K. Nguyen, and M. Cheriet. "Online Data Center Traffic Classification Based on Inter-Flow Correlations." IEEE Access (2020) (Saber et al., 2020);

- N. Haddaji, **A. Bayati**, K. Nguyen, and M. Cheriet. "BackHauling-as-a-Service (BHaaS) for 5G optical sliced networks: an optimized TCO approach," Journal of Lightwave Technology, 2018 (Haddaji et al., 2018);

List of conference papers:

- **A. Bayati**, K. K. Nguyen and M. Cheriet, "Greening The Network Using Traffic Prediction and Link Rate Adaptation," 2019 IEEE Sustainability through ICT Summit (StICT), Montréal, QC, Canada, 2019 (Bayati et al., 2019b);

- **A. Bayati**, K. Nguyen, and M. Cheriet. "Embedding Multiple-Step-Ahead Traffic Prediction in Network Energy Efficiency Problem," 15th IEEE IWCMC, 2019 (Bayati et al., 2019a);

- **A. Bayati**, V. Asghari, K. Nguyen, and M. Cheriet. "Gaussian process regression-based traffic modeling and prediction in high-speed networks," IEEE GLOBECOM, Washington, D.C. USA, 2016 (Bayati et al., 2016);

- M. Si-Saber, **A. Bayati**, K. Nguyen, and M. Cheriet, "Featuring Real-time Imbalanced Network Traffic Classification," in IEEE SmartData-2018, Halifax, Canada, August 2018 (Amina et al., 2018);

- S. Aghabozorgi, **A. Bayati**, K. Nguyen, C. Despins, M. Cheriet, "Toward predictive handover mechanism in software-defined enterprise Wi-Fi networks," IEEE Sustainability through ICT Summit StICT 2019 (Aghabozorgi et al., 2019);

BIBLIOGRAPHY

Adas, A. (1997, Jul). Traffic models in broadband networks. *Communications Magazine, IEEE*, *35*(7), 82-89.

Addis, B., Capone, A., Carello, G., Gianoli, L. G., & Sanso, B. (2014). Energy management through optimized routing and device powering for greener communication networks. *IEEE/ACM Transactions on Networking (TON)*, *22*(1), 313–325.

Aghabozorgi, S., Bayati, A., Nguyen, K.-k., Despins, C., & Cheriet, M. (2019). Toward predictive handover mechanism in software-defined enterprise wi-fi networks. In *2019 ieee sustainability through ict summit (stict)* (pp. 1–6).

Alanazi, S., Dabbagh, M., Hamdaoui, B., Guizani, M., & Zorba, N. (2017). Reducing data center energy consumption through peak shaving and locked-in energy avoidance. *IEEE Transactions on Green Communications and Networking*, *1*(4), 551–562.

Amina, S. S. M., Abdolkhalegh, B., Khoa, N. K., & Mohamed, C. (2018). Featuring real-time imbalanced network traffic classification. In *2018 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)* (pp. 840–846).

Andrews, M., Anta, A. F., Zhang, L., & Zhao, W. (2012, Feb). Routing for power minimization in the speed scaling model. *IEEE/ACM Transactions on Networking*, *20*(1), 285-294.

Barrow, D. K., & Crone, S. F. (2016). A comparison of adaboost algorithms for time series forecast combination. *International Journal of Forecasting*, *32*(4), 1103–1119.

Bayati, A., Asghari, V., Nguyen, K. K., & Cheriet, M. (2016, December). Gaussian process regression based traffic modeling and prediction in High-Speed networks. In *2016 ieee global communications conf. (globecom2016 cqrm).* Washington, USA.

Bayati, A., Nguyen, K., & Cheriet, M. (2018). Multiple-step-ahead traffic prediction in high-speed networks. *IEEE Communications Letters*, 1-1.

Bayati, A., Nguyen, K. K., & Cheriet, M. (2019a). Embedding multiple-step-ahead traffic prediction in network energy efficiency problem. In *2019 15th international wireless communications & mobile computing conference (iwcmc)* (pp. 1757–1763).

Bayati, A., Nguyen, K. K., & Cheriet, M. (2019b). Greening the network using traffic prediction and link rate adaptation. In *2019 ieee sustainability through ict summit (stict)* (pp. 1–7).

Bayati, A., Nguyen, K.-K., & Cheriet, M. (2020). Gaussian process regression ensemble model for network traffic prediction. *IEEE Access*, *8*, 176540–176554.

Brockwell, P., & Davis, R. (2013). *Introduction to time series and forecasting*. Springer New

York.

The CAIDA UCSD Anonymized Internet Traces. (2008-2015). Center for Applied Internet Data Analysis. Retrieved from http://www.caida.org/data/passive/

Cao, Z., Kodialam, M., & Lakshman, T. (2016). Joint static and dynamic traffic scheduling in data center networks. *IEEE/ACM Transactions on Networking (TON)*, *24*(3), 1908–1918.

Challita, U., Dong, L., & Saad, W. (2018). Proactive resource management for lte in unlicensed spectrum: A deep learning perspective. *IEEE Trans. on Wireless Communic.*, *17*(7), 4674-4689.

Chatzis, S. P., & Demiris, Y. (2012). Nonparametric mixtures of gaussian processes with power-law behavior. *IEEE transactions on neural networks and learning systems*, *23*(12), 1862–1871.

Chen, D. (2017). Research on traffic flow prediction in the big data environment based on the improved rbf neural network. *IEEE Transactions on Industrial Informatics*, *13*(4), 2000–2008.

Cheng, P., Wang, S., Ma, J., Sun, J., & Xiong, H. (2017). Learning to recommend accurate and diverse items. In *Proceedings of the 26th international conference on world wide web* (pp. 183–192).

Choudhury, G., Lynch, D., Thakur, G., & Tse, S. (2018). Two use cases of machine learning for sdn-enabled ip/optical networks: traffic matrix prediction and optical path performance prediction. *IEEE/OSA Journal of Optical Communications and Networking*, *10*(10), D52–D62.

Christensen, K., Reviriego, P., Nordman, B., Bennett, M., Mostowfi, M., & Maestro, J. A. (2010). Ieee 802.3 az: the road to energy efficient ethernet. *IEEE Communications Magazine*, *48*(11), 50–56.

Cisco Nexus 9500 Series Switches Data Sheet. (n.d.). Retrieved from https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/ethernet-x710-t2l-t4l-brief.pdf

Dehury, C. K., & Sahoo, P. K. (2020). Failure aware semi-centralized virtual network embedding in cloud computing fat-tree data center networks. *IEEE Transactions on Cloud Computing*.

Destounis, A., Paris, S., Maggi, L., Paschos, G. S., & Leguay, J. (2018). Minimum cost sdn routing with reconfiguration frequency constraints. *IEEE/ACM Transactions on Networking*, *26*(4), 1577–1590.

Ferreira, A. J., & Figueiredo, M. A. T. (2012). Boosting algorithms: A review of methods, theory, and applications. In C. Zhang & Y. Ma (Eds.), *Ensemble machine learning: Methods and applications* (pp. 35–85). Boston, MA: Springer US.

Fisher, W., Suchara, M., & Rexford, J. (2010). Greening backbone networks: reducing energy consumption by shutting off cables in bundled links. In *Proceedings of the first acm sigcomm workshop on green networking* (pp. 29–34).

Francini, A. (2012). Selection of a rate adaptation scheme for network hardware. In *Infocom, 2012 proceedings ieee* (p. 2831-2835).

Francini, A., Fortune, S., Klein, T., & Ricca, M. (2015, May). A low-cost methodology for profiling the power consumption of network equipment. *IEEE Communications Magazine*, *53*(5), 250-256.

GeSi SMARTer2030. (n.d.). Retrieved from https://smarter2030.gesi.org/downloads/Full_report.pdf

Girard, A., Rasmussen, C. E., Candela, J. Q., & Murray-Smith, R. (2003). Gaussian process priors with uncertain inputs-application to multiple-step ahead time series forecasting. *Advances in neural information processing systems*, 545–552.

Gunaratne, C., Christensen, K., Nordman, B., & Suen, S. (2008, April). Reducing the energy consumption of ethernet with adaptive link rate (alr). *IEEE Transactions on Computers*, *57*(4), 448-461.

Gupta, M., & Singh, S. (2003). Greening of the internet. In *Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications* (pp. 19–26). New York, NY, USA: ACM.

Haddaji, N., Bayati, A., Nguyen, K.-K., & Cheriet, M. (2018). Backhauling-as-a-service (bhaas) for 5g optical sliced networks: an optimized tco approach. *Journal of Lightwave Technology*, *36*(18), 4006–4017.

Haghighat, N., Kalbkhani, H., Shayesteh, M. G., & Nouri, M. (2015). Variable bit rate video traffic prediction based on kernel least mean square method. *IET Image Processing*, *9*(9), 777-794.

Hardegen, C., Pfülb, B., Rieger, S., & Gepperth, A. (2020). Predicting network flow characteristics using deep learning and real-world network traffic. *IEEE Transactions on Network and Service Management*, *17*(4), 2662–2676.

Hernández, S., & Donoso, Y. (2017). Green path: Optimization model for network energy efficiency using adaptive link rate. In *2017 ieee 14th international conference on mobile ad hoc and sensor systems (mass)* (pp. 610–615).

Hong, P., Xue, K., & Li, D. (2018). Resource aware routing for service function chains in sdn and nfv-enabled network. *IEEE Transactions on Services Computing*.

Hu, J., Li, X., & Ou, Y. (2014, Dec). Online gaussian process regression for time-varying manufacturing systems. In *Control automation robotics vision (icarcv), 2014 13th*

international conference on (p. 1118-1123).

Huin, N., Rifai, M., Giroire, F., Pacheco, D. L., Urvoy-Keller, G., & Moulierac, J. (2018). Bringing energy aware routing closer to reality with sdn hybrid networks. *IEEE Transactions on Green Communications and Networking*, *2*(4), 1128–1139.

Intel Ethernet Network Adapter X710-T2L. (n.d.). Retrieved from https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/ethernet-x710-t2l-t4l-brief.pdf

Internet2 Abilene Network. (2007). Internet2. Retrieved from http://noc.net.internet2.edu/i2network/live-network-status/historical-abilene-data.html

Islam, M. A., Ren, S., Mahmud, A. H., & Quan, G. (2015). Online energy budgeting for cost minimization in virtualized data center. *IEEE Transactions on Services Computing*, *9*(3), 421–432.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, *3*(1), 79–87.

Jiang, J., Ma, S., Li, B., & Li, B. (2016). Adia: Achieving high link utilization with coflow-aware scheduling in data center networks. *IEEE Transactions on Cloud Computing*.

Karagiannis, T., Molle, M., & Faloutsos, M. (2004, Sept). Long-range dependence ten years of internet traffic modeling. *Internet Computing, IEEE*, *8*(5), 57-64.

Kaur, K., Garg, S., Aujla, G. S., Kumar, N., Rodrigues, J. J., & Guizani, M. (2018). Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. *IEEE communications magazine*, *56*(2), 44–51.

Khan, S. S., & Ahmad, A. (2018). Relationship between variants of one-class nearest neighbors and creating their accurate ensembles. *IEEE Transactions on Knowledge and Data Engineering*, *30*(9), 1796–1809.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, *37*, 132–156.

Krithikaivasan, B., Zeng, Y., Deka, K., & Medhi, D. (2007, June). Arch-based traffic forecasting and dynamic bandwidth provisioning for periodically measured nonstationary traffic. *IEEE/ACM Trans. Netw.*, *15*(3), 683–696.

Leland, W. E., Willinger, W., Taqqu, M. S., & Wilson, D. V. (1995). On the self-similar nature of ethernet traffic. *ACM SIGCOMM computer communication review*, *25*(1), 202–213.

Leland, W. E., & Wilson, D. V. (1991). High time-resolution measurement and analysis of lan traffic: Implications for lan interconnection. In *Ieee infcom'91. the conference*

on computer communications. tenth annual joint comference of the ieee computer and communications societies proceedings (pp. 1360–1366).

Li, D., Shang, Y., He, W., & Chen, C. (2015). Exr: greening data center network with software defined exclusive routing. *IEEE Transactions on Computers*, *64*(9), 2534–2544.

Li, P., & Chen, S. (2016). A review on gaussian process latent variable models. *{CAAI} Transactions on Intelligence Technology*, *1*(4), 366 - 376.

Li, X., Garraghan, P., Jiang, X., Wu, Z., & Xu, J. (2017). Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy. *IEEE Transactions on Parallel and Distributed Systems*, *29*(6), 1317–1331.

Li, Y., Liu, H., Yang, W., Hu, D., Wang, X., & Xu, W. (2016, Dec). Predicting inter-data-center network traffic using elephant flow and sublink information. *IEEE Transactions on Network and Service Management*, *13*(4), 782-792.

Li, Y., Wen, Y., Tao, D., & Guan, K. (2019). Transforming cooling optimization for green data center via deep reinforcement learning. *IEEE transactions on cybernetics*, *50*(5), 2002–2013.

Liu, D., Khoukhi, L., & Hafid, A. (2018, July). Prediction-based mobile data offloading in mobile cloud computing. *IEEE Transactions on Wireless Communications*, *17*(7), 4660-4673.

Liu, G.-R., Lin, P., & Awad, M. K. (2018). Modeling energy saving mechanism for green routers. *IEEE Transactions on Green Communications and Networking*, *2*(3), 817–829.

Liu, J., Vitelli, V., Zio, E., & Seraoui, R. (2015). A novel dynamic-weighted probabilistic support vector regression-based ensemble for prognostics of time series data. *IEEE Transactions on Reliability*, *64*(4), 1203–1213.

Lu, Z., Wu, X., & Bongard, J. C. (2015). Active learning through adaptive heterogeneous ensembling. *IEEE Transactions on Knowledge and Data Engineering*, *27*(2), 368–381.

Lyu, X., Li, Y., Ren, N., Nan, C., Cao, D., & Jiang, S. (2019). Optimization of high-density and high-efficiency switched-tank converter for data center applications. *IEEE Transactions on Industrial Electronics*.

Maaloul, R., Taktak, R., Chaari, L., & Cousin, B. (2018). Energy-aware routing in carrier-grade ethernet using sdn approach. *IEEE Transactions on Green Communications and Networking*, *2*(3), 844–858.

Manjate, J. A., Hidell, M., & Sjödin, P. (2018). Can energy-aware routing improve the energy savings of energy-efficient ethernet? *IEEE Transactions on Green Communications and Networking*.

Meeds, E., & Osindero, S. (2006). An alternative infinite mixture of gaussian process experts. In *Advances in neural information processing systems* (pp. 883–890).

Minku, L. L., White, A. P., & Yao, X. (2009). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on knowledge and Data Engineering*, *22*(5), 730–742.

Montgomery, D., Jennings, C., & Kulahci, M. (2015). *Introduction to time series analysis and forecasting*. Wiley.

Morales, F., Ruiz, M., Gifre, L., Contreras, L. M., López, V., & Velasco, L. (2017). Virtual network topology adaptability based on data analytics for traffic prediction. *IEEE/OSA Journal of Optical Communications and Networking*, *9*(1), A35–A45.

Moulierac, J., & Phan, T. K. (2015). Optimizing IGP link weights for energy-efficiency in multi-period traffic matrices. *Computer Communications*, *61*, 79 - 89.

Multi-rate Ethernet PHY Intel FPGA IP Release Notes - 2018. (n.d.). Retrieved from https://www.intel.com/content/www/us/en/programmable/products/intellectual-property/ip/interface-protocols/m-alt-10g-multirate-phy.html

Naeem, F., Tariq, M., & Poor, H. V. (2020). Sdn-enabled energy-efficient routing optimization framework for industrial internet of things. *IEEE Transactions on Industrial Informatics*, *17*(8), 5660–5667.

Nguyen, T. D., Khan, J. Y., & Ngo, D. T. (2018). A distributed energy-harvesting-aware routing algorithm for heterogeneous iot networks. *IEEE Transactions on Green Communications and Networking*, *2*(4), 1115–1127.

Palm, R. (2007). Multiple-step-ahead prediction in control systems with gaussian process models and ts-fuzzy models. *Engineering Applications of Artificial Intelligence*, *20*(8), 1023 - 1035.

Peng, Y., Chen, K., Wang, G., Bai, W., Zhao, Y., Wang, H., . . . Gu, L. (2016, Aug). Towards comprehensive traffic forecasting in cloud computing: Design and application. *IEEE/ACM Transactions on Networking*, *24*(4), 2210-2222.

Polikar, R. (2012). Ensemble machine learning: Methods and applications. In C. Zhang & Y. Ma (Eds.), *Ensemble machine learning: Methods and applications* (pp. 1–34). Boston, MA: Springer US.

Rahman, M. M., Islam, M. M., Murase, K., & Yao, X. (2016). Layered ensemble architecture for time series forecasting. *IEEE transactions on cybernetics*, *46*(1), 270–283.

Rahnamay-Naeini, M., Baidya, S. S., Siavashi, E., & Ghani, N. (2016, Feb). A traffic and resource-aware energy-saving mechanism in software defined networks. In *2016*

international conference on computing, networking and communications (icnc) (p. 1-5).

Rasmussen, C. E., & Ghahramani, Z. (2001). Infinite mixtures of gaussian process experts. In *Proceedings of the 14th international conference on neural information processing systems: Natural and synthetic* (pp. 881–888). USA: MIT Press.

Rasmussen, C. E., & Ghahramani, Z. (2002). Infinite mixtures of gaussian process experts. In *Advances in neural information processing systems* (pp. 881–888).

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Saber, M. A. S., Ghorbani, M., Bayati, A., Nguyen, K.-K., & Cheriet, M. (2020). Online data center traffic classification based on inter-flow correlations. *IEEE Access*, *8*, 60401–60416.

Seijo-Pardo, B., Porto-Díaz, I., Bolón-Canedo, V., & Alonso-Betanzos, A. (2017). Ensemble feature selection: homogeneous and heterogeneous approaches. *Knowledge-Based Systems*, *118*, 124–139.

Sun, S., & Xu, X. (2010). Variational inference for infinite mixtures of gaussian processes with applications to traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, *12*(2), 466–475.

Sun, Y., Tang, K., Minku, L. L., Wang, S., & Yao, X. (2016). Online ensemble learning of data streams with gradually evolved classes. *IEEE Transactions on Knowledge and Data Engineering*, *28*(6), 1532–1545.

Support Documents for Multi-rate Ethernet PHY Intel FPGA IP Core. (n.d.). Retrieved from https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/rn/rn-multirate-ethernet.pdf

Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, *39*(8), 7067 - 7083.

Tang, J., Mumey, B., Xing, Y., & Johnson, A. (2012, March). On exploiting flow allocation with rate adaptation for green networking. In *Infocom, 2012 proceedings ieee* (p. 1683-1691).

Tekin, C., Yoon, J., & Van Der Schaar, M. (2016). Adaptive ensemble learning with confidence bounds. *IEEE Transactions on Signal Processing*, *65*(4), 888–903.

Tresp, V. (2001). Mixtures of gaussian processes. In *Advances in neural information processing systems* (pp. 654–660).

Trinh, H., Calyam, P., Chemodanov, D., Yao, S., Lei, Q., Gao, F., & Palaniappan, K. (2018). Energy-aware mobile edge computing and routing for low-latency visual data processing. *IEEE Transactions on Multimedia*, *20*(10), 2562–2577.

Tsybakov, B., & Georganas, N. D. (1998, Sep). Self-similar processes in communications networks. *Information Theory, IEEE Transactions on*, *44*(5), 1713-1725.

Tunca, C., Isik, S., Donmez, M. Y., & Ersoy, C. (2014). Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink. *IEEE Transactions on Mobile Computing*, *14*(9), 1947–1960.

Uhlig, S., Quoitin, B., Lepropre, J., & Balon, S. (2006). Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, *36*(1), 83–86.

Vereecken, W., Van Heddeghem, W., Deruyck, M., Puype, B., Lannoo, B., Joseph, W., ... Demeester, P. (2011). Power consumption in telecommunication networks: overview and reduction strategies. *IEEE Communications Magazine*, *49*(6), 62–69.

Vitturi, S., & Tramarin, F. (2015). Energy efficient ethernet for real-time industrial networks. *IEEE Transactions on Automation Science and Engineering*, *12*(1), 228–237.

Waikato VIII 2011. (2011). University of Waikato. Retrieved from http://wand.net.nz/wits/waikato/8/

Wang, J. M., Wang, Y., Dai, X., & Bensaou, B. (2015). Sdn-based multi-class qos guarantee in inter-data center communications. *IEEE Transactions on Cloud Computing*, *7*(1), 116–128.

Wang, L., Zhang, F., Hou, C., Aroca, J. A., & Liu, Z. (2013, Aug). Incorporating rate adaptation into green networking for future data centers. In *Network computing and applications (nca), 2013 12th ieee international symposium on* (p. 106-109).

Wang, Y., Wang, J., Zhang, W., Yang, J., & Gui, G. (2020). Deep learning-based cooperative automatic modulation classification method for mimo systems. *IEEE Transactions on Vehicular Technology*, *69*(4), 4575–4579.

Willinger, W., Paxson, V., Riedi, R., & Taqqu, M. (2003). *Theory and applications of long-range dependence* (P. Doukhan, G. Oppenheim, & M. Taqqu, Eds.). Birkhauser Basel.

Xie, Y., Hu, J., Xiang, Y., Yu, S., Tang, S., & Wang, Y. (2013, Sept). Modeling oscillation behavior of network traffic by nested hidden markov model with variable state-duration. *IEEE Transactions on Parallel and Distributed Systems*, *24*(9), 1807-1817.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems* (pp. 802–810).

Yan, W. (2012). Toward automatic time-series forecasting using neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(7), 1028–1039.

Yang, J., Zeng, X., Zhong, S., & Wu, S. (2013, June). Effective neural network ensemble approach for improving generalization performance. *IEEE Transactions on Neural Networks and Learning Systems*, *24*(6), 878-887.

Yin, J., Lu, X., Zhao, X., Chen, H., & Liu, X. (2015, March). Burse: A bursty and self-similar workload generator for cloud computing. *Parallel and Distributed Systems, IEEE Transactions on*, *26*(3), 668-680.

Yuan, C., & Neubauer, C. (2009). Variational mixture of gaussian process experts. In *Advances in neural information processing systems* (pp. 1897–1904).

Zemmouri, S., Vakilinia, S., & Cheriet, M. (2016, October). Let's adapt to network change: Towards energy saving with rate adaptation in SDN. In *12th international conference on network and service management (cnsm 2016).* Montreal, Canada.

Zhan, Y., Ghamkhari, S. M., Akhavan-Hejazi, H., Xu, D., & Mohsenian-Rad, H. (2018). Cost-aware traffic management under demand uncertainty from a colocation data center user's perspective. *IEEE Transactions on Services Computing*, 1-1.

Zhang, B., Sabhanatarajan, K., Gordon-Ross, A., & George, A. (2008). Real-time performance analysis of adaptive link rate. In *2008 33rd ieee conference on local computer networks (lcn)* (pp. 282–288).

Zhang, C., Lim, P., Qin, A., & Tan, K. C. (2017). Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems*, *28*(10), 2306–2318.

Zhang, M., Fu, H., Li, Y., & Chen, S. (2017). Understanding urban dynamics from massive mobile traffic data. *IEEE Transactions on Big Data*, *5*(2), 266–278.

Zhang, N., Yao, Z.-Q., Liu, Y., Boyd, S. P., & Luo, Z.-Q. (2017). Dynamic resource allocation for energy efficient transmission in digital subscriber lines. *IEEE Transactions on Signal Processing*, *65*(16), 4353–4366.

Zhang, P., Song, S., Niu, S., & Zhang, R. (2021). A hybrid artificial immune-simulated annealing algorithm for multiroute job shop scheduling problem with continuous limited output buffers. *IEEE Transactions on Cybernetics*.

Zhang, X.-L., & Wang, D. (2016). A deep ensemble learning method for monaural speech separation. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, *24*(5), 967–977.

Zhao, J., & Sun, S. (2016). High-order gaussian process dynamical models for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, *17*(7), 2014–2019.