UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

PH.D. THESIS

Machine Learning Assisted QoT Estimation for Optical Networks Optimization

Doctoral thesis by:
Ankush Mahajan

A thesis submitted in fulfilment of the requirements for the International doctoral degree

Supervisor: Dr. Ricardo Martínez
Co-supervisor: Dr. Raul Muñoz
Centre Tecnològic de Telecomunicacions de Catalunya, Barcelona, Spain

Industrial Supervisor: Dr. Konstantinos (Kostas) Christodoulopoulos
Nokia Bell Labs, Stuttgart, Germany

Tutor: Prof. Salvatore Spadaro
Universitat Politècnica de Catalunya, Barcelona, Spain

Universitat Politècnica de Catalunya (UPC)
Department of Signal Theory and Communications (TSC)

Ankush Mahajan
Machine Learning Assisted QoT Estimation for Optical Networks Optimization
Ph.D. Thesis
Supervisor (s): Dr. Ricardo Martínez and Dr. Raul Muñoz (CTTC)
Industrial Supervisor: Dr. Konstantinos (Kostas) Christodoulopoulos (Nokia Bell Labs)
Tutor: Prof. Salvatore Spadaro (UPC)

Universitat Politècnica de Catalunya (UPC)
Department of Signal Theory and Communications
Carrer de Jordi Girona, 1-3
08034 Barcelona, Spain

Abstract

This thesis reports the conducted investigations and novel contributions in the field of Machine Learning (ML) assisted optical network planning, operation and dynamic optimization. The tremendous increase in data traffic has spurred a rapid evolution of the optical networks for a reliable, affordable, cost effective and scalable network infrastructure. To meet some of these requirements, network operators are pushing toward disaggregation. Network disaggregation focuses on decoupling the traditional monolithic optical transport hardware into independent functional blocks that interoperate. This enables a relatively free market where the network operators/owners could choose the best-in-class equipment from different vendors overcoming the vendor lock-in, at better prices. In this multi-vendor disaggregation context, the used equipment would impact the physical layer and the overall network behavior. This results in increasing the uncertainty on the performance when compared to a traditional single vendor aggregated approach.

For effective optical network planning, operation and optimization, it is necessary to estimate the Quality of Transmission (QoT) of the connections. Network designers are interested in accurate and fast QoT estimation for services to be established in a future or existing network. Typically, QoT estimation is performed using a Physical Layer Model (PLM) which is included in the QoT estimation tool or Qtool. A design margin is generally included in a Qtool to account for the modeling and parameter inaccuracies, to re-assure an acceptable performance. The PLM with the addition of a design margin is typically referred to as the QoT estimation tool or Qtool. PLM accuracy is highly important as modeling errors translate into a higher design margin which in turn translate into wasted capacity or unwanted regeneration.

During the past few decades, major achievements were accomplished on defining accurate and fast PLMs of optical networks utilizing traditional methods with either analytical or numerical solutions. Recently monitoring and ML techniques have been proposed to account for the actual network conditions and improving the accuracy of the PLM in single vendor networks. This in turn results in more accurate QoT estimation. Indeed, QoT estimation seems to be a suitable problem for ML. The reason behind this is that optical networks are complex; models include several parameters which highly interplay. As so, QoT estimation tool performance is highly dependent on ubiquitous physical layer uncertainties induced by diverse subsystems such as transponders (TPs), Erbium Doped Fiber Amplifiers (EDFAs), Reconfigurable Optical Add and Drop Multiplexer (ROADM), etc., and the transmission medium, i.e., the optical fiber. The model space becomes even more complicated during dynamic network optimization. This is because, there are transient effects, as well as the fact that dynamic optimization requires higher Qtool accuracy to replace static/overprovisioning solutions.

The first part of the thesis focuses on the ML assisted accurate QoT estimation techniques. In this regard, we developed a model that uses monitoring information from an operating network combined with supervised ML regression techniques to understand the network conditions. In particular, we model the generated penalties due to i) EDFA gain ripple effect, and ii) filter spectral shape uncertainties at ROADM nodes. We developed independent ML models for the two effects which are then used for

estimating the penalties of new connection requests, improving the Qtool estimation accuracy and thus reducing the required design margin.

Different TP vendors use different digital signal processing (DSP) techniques and various third-party components, resulting in performance variations within multi-vendor TPs. Furthermore, with the aim of improving the Qtool estimation accuracy in networks with multi-vendor TPs, we propose PLM extensions. In particular, we introduce four TP vendor dependent performance factors that capture the performance variations of multi- vendor TPs. To verify the potential improvement, we studied the following two use cases with the proposed PLM, to: i) optimize the TPs launch power; and ii) reduce design margin in incremental planning. On top of that, we also devised a ML-based scheme to identify these performance factors in both offline and online network planning scenarios.

An accurate and fast Qtool is an essential requirement for almost every optimization task of an optical network. For optimization tasks that involve few calculations/iterations and actions, e.g., the establishment or reconfiguration of a single connection, the accuracy of the Qtool would be acceptable if it was trained/aligned before such action. However, for more complex dynamic optimization tasks involving multiple iterations and reconfigurations, the Qtool accuracy becomes critical. Algorithms used in such dynamic optimization tasks are typically iterative; they calculate several intermediate solutions and improve them to find the optimum, which is then configured in the network. However, the accuracy of the Qtool deteriorates after several intermediate calculations, and at some extent, it can fail to support the optimization calculations.

In consequence, the last part of this thesis aims at investigating and solving the issue of accuracy limitation of Qtool in dynamic optimization tasks. To keep the models aligned to the real conditions, the digital twin (DT) concept is gaining significant attention in the research community. DT is a digital representation of the real/physical system used to understand and then optimize the targeted system. The DT is more than a model of the system; it includes an evolving set of data, and a means to dynamically adjust the model. Based on the DT fundamentals, we devised and implemented an iterative closed control loop process that, after several intermediate iterations of the optimization algorithm, configures the network, monitors, and retrains the Qtool. For the Qtool retraining, we adopt a ML-based nonlinear regression fitting technique. The key advantage of this novel scheme is that whilst the network operates, the Qtool parameters are retrained according to the monitored information with the adopted ML model. Hence, the Qtool tracks the projected states intermediately calculated by the algorithm. This reduces the optimization time as opposed to directly probing and monitoring the network. To make this scheme and the overall optimization task even faster, we also present an adaptive version of the closed control loop/retraining process.

In conclusion, the results attained in this thesis lay the groundwork for physical layer modeling schemes, accurate QoT models, and their interaction with dynamic optimization tasks for optical networks. The obtained results in this thesis can shed light into future ML-oriented research within optical network planning in single and multi- vendor environment, where the modeling of the physical layer and its performance is still under uncertainty.

Acknowledgements

It is incredible how quickly time flies. When I first arrived in Europe and began my Ph.D. studies, I had no idea who I would become. Everything was new to me. Even now, as I sit in my room typing these words, I still can't believe how much has happened to me in the last three years. Without a doubt, it has been a life-changing experience and an excellent adventure. I consider myself fortunate.

People say this is a difficult road, but thanks to my wonderful supervisors, I was able to navigate it without too much difficulty. I would like to express my gratitude to Dr. Ricardo Martínez, my supervisor, for his faith in me as a student, as well as for his ongoing support, guidance, and encouragement. Thank you for your detailed reviews and recommendations on scientific writing. Thank you so much for all of your invaluable advices, technical and non-technical assistance, and support.

I would also like to thank Dr. Raul Muñoz, my co-supervisor, for his sustained guidance, leadership, and support throughout my doctoral studies. This work would not have been possible without his trust on me. His encouragement, leadership and enthusiasm has served as a source of inspiration and motivation to me over the last three years and will continue to drive my endeavors beyond this point. I could not think of a better advisor, motivator, project leader and coordinator to have.

Undoubtedly, this Ph.D. research work would not have been possible without Dr. Konstantinos (Kostas) Christodoulopoulos, my industrial supervisor. I would especially like to thank him for his mentorship, and guidance, in the world of research, for his unmatched confidence, enthusiasm and leadership when diving into new and unexplored topics. He is a very nice leader who is an expert in motivating, encouraging students and an excellent advisor in scientific writing. He is so patient and kind enough to listen to my nonsense questions and comments, to advise me on scientific writing, and to support me in my search for a good job. He is an excellent role model for me in terms of seeing life in a new light. Thank you very much, Kosta!

I would also like to thank Prof. Salvatore Spadaro, my university tutor for his support and guidance over the past three years. Also, thank you for all of your consistent assistance and help from the time I enrolled in the UPC Ph.D. program until my scheduled completion. You made everything go very smoothly and quickly.

I'd also like to express my gratitude to Dr. Michela Svaluto Moreolo, Dr. Josep Maria Fabrega, Dr. Laia Nadal, and Javier Vílchez for all of their assistance and support during experiments at CTTC laboratory and providing timely review for joint deliverables.

I would also like to thank Fabiano Locatelli, my project partner, for all his help, friendship, and the good times we had together. Thank you for always cheering me up and motivating me during my difficult times over the last three years. I would also like to thank Max, my flatmate, for all his help and support during these three years and the fun times we spent together.

I would also like to thank Lars Dembeck and Dr. Wolfram Lautenschlaeger for their support and assistance during my stay in Nokia Bell Labs, Stuttgart, Germany. I also appreciate the help from Karsten Schuh and Dr. Camille Delezoide from Nokia for their insightful feedback and suggestions on my work. I would also give my best appreciations to CTTC's administrative staff for all of their assistance and support during my travels and stays for conferences and industrial secondment at Nokia Bell Labs, Stuttgart, Germany.

I would also like to thank Dr. Paola Parolari from Politecnico di Milano, Italy for providing timely reviews for deliverables, Prof. Andreas Kirstaedter, University of Stuttgart, Germany and Prof. Massimo Tornatore from Politecnico di Milano, Italy for serving as external reviewers and members of Ph.D. evaluation committee. Thank you for your time and help in these milestones!

I am deeply indebted to my family and friends, whose love and support have been unwavering throughout the years. Thank you for your remote support during my study abroad and all your help, education, love, and encouragement in my life. Thanks to my sister Mansi for having faith in me and always keeping me grounded. A very special thank you to my parents, to whom I dedicate this thesis: I am grateful for your endless guidance and love, and for the sacrifices that you have made so that I can be where I am today. I owe all of my successes to the both of you.

Finally, I owe a huge debt of gratitude to the European Commission for supporting me with a Marie Skłodowska-Curie Fellowship.

Contents

List of Figures

List of Tables

Acronyms

AGC	Automatic Gain Control
API	Application Programming Interfaces
ASE	Amplified Spontaneous Emission
AWGN	Additive White Gaussian Noise
B&S	Broadcast & Select
BER	Bit Error Ratio
BW	Bandwidth
CAGR	Compound Annual Growth Rate
CAPEX	Capital Expenditure
CCL	Closed Control Loop
CD	Chromatic Dispersion
CDC	Colorless Directionless Contentionless
CMA	Constant Modulus Algorithm
CS	Constellation Shaping
CTTC	Centre Tecnològic de Telecomunicacions de Catalunya
DB	Database
dB	Decibel
DGE	Dynamic Gain Equalizer
DP	Dual Polarization
DSP	Digital Signal Processing
DT	Digital Twin
EDFA	Erbium Doped Fiber Amplifier
ESR	Early Stage Researcher
EU	European Union
FA	Filtering Uncertainties Aware
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FU	Filtering Uncertainty Unaware
FWM	Four Wave Mixing
Gb/s	Giga Bits Per Second
GFF	Gain Flattening Filter
GN	Gaussian Noise
H2020	Horizon 2020

HD	Hard Decision
ICT	Information and Communications Technology
IM-DD	Intensity Modulated-Direct Detection
IP	Internet Protocol
ISI	Inter Symbol Interference
ITU-T	International Telecommunication Union-Telecommunication
KPI	Key Performance Indicator
LM	Levenberg-Marquardt
LOGO	Local Optimization lead to Global Optimization
M-DB	Physical Layer Monitoring Database
MDP	Markov-Decision Processes
ML	Machine Learning
MMA	Multi-Modulus Algorithm
MMH	Maximum Marginal Hyperplane
MSA	Multi-Source Agreement
MSCA	Marie Sklodowska-Curie Actions
MSE	Mean Square Error
MV	Multi-vendor
NBI	North Bound Interfaces
N-DB	Network Configuration Database
NF	Noise Figure
NLI	Non-Linear Interference
nm	Nano Meter
NMS	Network Management System
NN	Neural Network
NSR	Noise to Signal Ratio
OADM	Optical Add Drop Multiplexers
OAM	Operation and Administration Maintenance
Obj.	Objective
OCM	Optical Channel Monitor
OLS	Open or Disaggregated Line System
ONFIRE	Future Optical Networks for Innovation, Research and Experimentation
OPM	Optical Performance Monitors
OSNR	Optical Signal to Noise Ratio

OTN	Optical Transport Network
p2p	peak-to-peak
PCE	Path Computation Element
PDL	Polarization Dependent Loss
PDM	Polarization Division Multiplexing
Ph.D.	Doctor of Philosophy
PLI	Physical Layer Impairment
PLM	Physical Layer Model
PM	Polarization Multiplexed
PMD	Polarization Mode Dispersion
PSD	Power Spectral Density
QAM	Quadrature Amplitude Modulation
Q-Factor	Quality- Factor
QoT	Quality of Transmission
QPSK	Quadrature Phase Shift Keying
RA	Ripple Aware
RBF	Radial Basis Function
REA	Ripple Dynamic Gain Equalizer Aware
RFA	Ripple and Filtering Aware
RFU	Ripple Plus Filtering Uncertainties Unaware
RL	Reinforcement Learning
ROADM	Reconfigurable Optical Add and Drop Multiplexer
RRC	Root Raised Cosine
RSA	Routing and Spectrum Assignment
RU	Ripple Unaware
Rx.	Receiver
S&S	Switch & Select
SBI	South Bound Interfaces
SBS	Stimulated Brillouin Scattering
SCI	Self Channel Interference
SD	Soft Decision
SDN	Software Defined Network
SNR	Signal to Noise Ratio
SPM	Self Phase Modulation
SRS	Stimulated Raman Scattering

SSF	Split Step Fourier
SSMF	Standard Single Mode Fiber
Std	Standard Deviation
SV	Single Vendor
SVM	Support Vector Machine
SVMR	Support Vector Machine Regression
Tb/s	Tera Bits Per Second
TED	Traffic Engineering Database
THz	Tera Hertz
TP	Transponder
Tx.	Transmitter
UPC	Universitat Politècnica de Catalunya
VOA	Variable Optical Attenuator
VPI	VPIphotonics
WDM	Wavelength Division Multiplexing
WSS	Wavelength Selective Switches
XCI	Cross Channel Interference
XPM	Cross Phase Modulation

Chapter 1

Introduction

This chapter starts with the basic overview about the context and the motivation behind the research work presented in this Doctor of Philosophy (Ph.D.) thesis. Following that, we provide a list of the research objectives addressed in this Ph.D. thesis. We also provide a brief overview about the followed research methodology. We then briefly describe the sponsored project for this Ph.D. thesis. Finally, we include the list of research papers published as an outcome of this Ph.D. thesis.

1.1 Context and Motivations

Internet traffic and, in general, the necessities of high data transport capacity have increased at a nearly exponential rate, posing significant challenges to the telecommunications industry in meeting these demands. Optical fiber-based transmission systems and networks have already proven to be a dominant technology for high-speed, high-capacity data transmission all the way to the last mile. In the last 50 years, the improvements made in terms of capacity, reliability, flexibility, scalability, optimization and management of optical communication systems and networks are extraordinary [1]. The performance of optical networks (specifically, transmission distance and data rate) keeps increasing in the last decades with the development of new technologies such as fixed and flexi-grid wavelength division multiplexing (WDM), erbium-doped fiber amplifiers (EDFAs), wavelength selective switches (WSSs), reconfigurable optical add-drop multiplexer (ROADM), forward error correction (FEC), coherent transmission and digital signal processing (DSP) based optical impairment compensation [2].

Year after year, the rapid development of new emerging services and applications such as cloud computing, high-definition video streaming, and new networking paradigms (e.g., Internet of Things) necessitate even more capacity. All of these requirements, however, must also meet the Quality of Transmission (QoT) requirements for guaranteed end-to-end performance [3], [4]. Emerging optical networks have already introduced flexibility in the optical transport, supporting heterogeneous data rates, optical spectrum channels, modulation formats, etc. [5], [6]. This leads to higher spectral efficiency and capacity, while keeping the network costs as low as possible [3], [7]. The *first* line of research is to efficiently plan and upgrade these optical networks in order to fulfil these demands. For effective optical network planning and upgrade, it is necessary to estimate the QoT of the connections prior to their establishment. This requires accurate models or tools to estimate the QoT of new or reconfigured connections. Typically, QoT

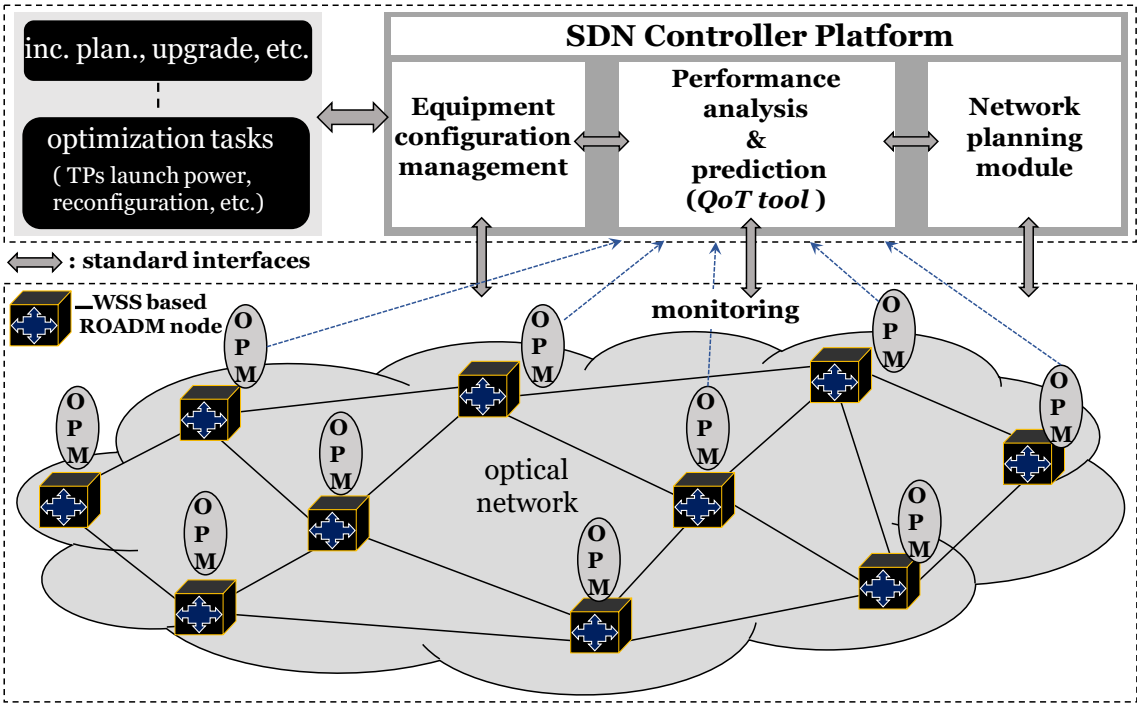


Figure. 1.1. Example of QoT estimation in network control and management.

estimation is performed using an analytical Physical Layer Model (PLM) which is included in the QoT estimation tool or also referred to as a Qtool. However, although the current generation of the optical networks provide vast optimization dimensions, optical networks are traditionally planned to be operated statically. In such static network operating mode, high margins are included in the planning phase to ensure acceptable QoT performance up to the end of life [8], [9]. *Lowering the margins and increasing efficiency reduces the network cost, motivating various research directions.*

Moreover, this substantial traffic growth will also push network operators for a continuous investment in their optical transport infrastructure. To keep up with the increase in the carried traffic, optical transmission systems must not only increase the number of bits transported per fiber, but also reduce the cost of transmission [10]. One of the key requirements to cope-up with the cost, is to develop highly interchangeable products to enable end-to-end vendor-diverse inter-operable coherent optical systems. The design and operation of such networks would require a PLM that accurately estimates the QoT of connections, also referred to as a Qtool, in such a diverse optical network environment. We can also rely on a typical single vendor PLM to estimate the performance of a multi-vendor disaggregated network. However, it may result in huge deviations in the estimated and actual QoT values. A solution to mitigate this is to add/increase the used margins on top of the PLM estimations. However, adding margins leads to both lower the efficiency and underutilization of the network resources (i.e., transport capacity). *Hence either proper extensions or a completely new physical layer modeling scheme are required to estimate QoT in such diverse multi-vendor optical networks.*

Another aligned research direction is to improve the network optimization tasks, which could increase bandwidth (BW) utilization by using techniques like transponder (TP) launch power optimization, physical layer reconfigurability, and so on. As mentioned, in the current deployed optical systems, the optical connections are fixed (statically planned) and remain unchanged once provisioned, with limited network management capabilities. Enabling fine-tuning in the optical physical layer, where individual network components and subsystems can be monitored and controlled, represents an appealing method for increasing the efficiency and thus capacity in optical networks. To perform dynamic control and management operations, one key metric is the QoT of optical connections. *Based on the QoT metric the network operator can make decisions to enhance the management and operation of the optical network.*

Several network designers are also considering the installation of optical performance monitors (OPMs) at nodes to support advanced network operations [11]. It is possible to extract information from the OPMs which is then used to train supervised ML models to gain a better understanding of the actual physical condition. Monitoring and ML schemes are adopted as an enabler to reduce the design margin for both (incremental) planning and dynamic optimization operations of the optical network [12], [13]. However, frequent interactions between both data plane and control planes (for network elements such as EDFAs, TPs, and OPMs) are needed to implement these schemes. A control layer manages all the steps for both planning and optimization operations, as shown in **Figure. 1.1**. A software defined networking (SDN) controller entity can collect telemetry and monitoring data that is then leveraged to train the standard Qtool with ML algorithms. This allows facilitating the physical layer controller actions. Therefore, the development of such control plane capabilities becomes essential for operating the optical systems aiming at improving the overall efficiency, which indeed is the major requirement for the conducted research addressed in this thesis.

1.2 Objectives of the Thesis

This Ph.D. thesis follows two distinct research directions. However, these research directions eventually complement each other to come up with a more efficient design, operation and management of future optical networks.

On the one hand, we explore data analytics and ML schemes to: *i).* model specific effects such as EDFA gain ripple and filtering penalties etc., and *ii).* refine QoT input parameters, to improve its estimation accuracy and reducing the design margin. It also includes novel improved PLM extension for more accurate QoT estimation in multivendor ecosystem. On the other hand, it also reports the schemes and mechanisms for network optimization operations such as transponders (TPs) launch power adjustment in static and dynamic network scenarios. Three specific objectives are stated to achieve these targets:

O.1 – Accuracy improvement and design margin reduction of QoT estimation tool

This objective focuses on the use of ML techniques on the monitoring information to improve the accuracy and reduce the design margin of the standard QoT estimation tool. To this end, this objective is divided into the following two sub-objectives:

O.1.1 - Modeling EDFA gain ripple penalties

O.1.2 - Modeling penalties due to filtering uncertainties at the ROADM nodes

O.2 – Qtool extensions for multivendor partial disaggregated optical networks

This objective addresses the improvement of the Qtool estimation accuracy considering novel performance factors in the existing PLM derived from multi-vendor TPs. The benefits of the proposed PLM are demonstrated by implementing two use cases/sub-objectives:

O.2.1 - Performance improvements achieved by optimizing TPs launch power during the planning phase (static)

O.2.2 - Design margin reduction in the incremental planning

O.3 – QoT estimator retraining for dynamic TPs launch power optimization

This objective focuses on using a novel adaptive and iterative closed control loop process that leverages monitoring and ML methods to improve the performance of dynamic multivariable optimization operations.

A summary of the goals of the thesis is presented in **Table. 1.1**.

Table. 1.1 - Thesis objectives

<table>
<tr><th>Objectives</th><th>Sub-objectives</th></tr>
<tr><td rowspan="2">O1
Accuracy improvement and design margin reduction of QoT estimation tool</td><td>O1.1 Modeling EDFA gain ripple penalties</td></tr>
<tr><td>O1.2 Modeling penalties due to filtering uncertainties at ROADM nodes</td></tr>
<tr><td rowspan="2">O2
PLM extensions for multivendor partial disaggregated optical networks</td><td>O2.1 TPs launch power optimization in planning phase (static)</td></tr>
<tr><td>O2.2 Design margin reduction in incremental planning as use case</td></tr>
<tr><td colspan="2">O3
QoT estimator retraining for dynamic TPs launch power optimization</td></tr>
</table>

1.3 Research Methodology

In this section, we present the methodology that we followed in this Ph.D. thesis. This started with an exhaustive review of the available literature and previously published works to clearly identify areas and potential gaps that had yet to be explored and investigated. This approach was conducted concurrently with the deployment of a new idea/approach/contribution to be exploited. Once the idea was determined, the next step was to formulate it as a formal problem to be solved. This favored identifying how the problem can be tackled and more importantly solved. To this end, for every single problem, a particular approach was designed which eventually was generalized to become a potential solution. The solution was then typically broken up into smaller ones before proceeding with the required implementations. The above methodology workflow is depicted in **Figure. 1.2**.

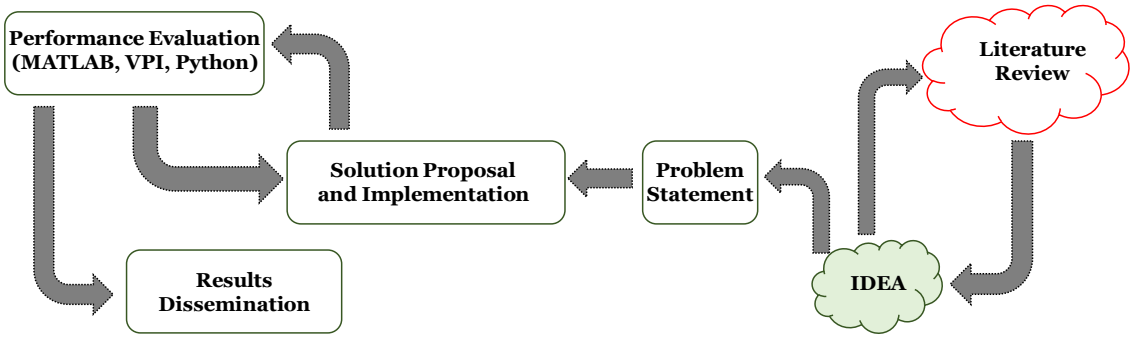


Figure. 1.2. Block diagram detailing the followed methodology during this Ph.D. thesis.

The objectives studied in this thesis required i). the generation of a large amount of data, ii). followed by the development of algorithms, and iii). their implementation and evaluation in appropriate platforms. Depending on the nature of the problem, different programming languages and licensed software tools were used. MATLAB and Python were mostly used to implement the algorithms developed in this thesis. To make the simulations more realistic, small testbeds were also implemented in VPI Transmission Maker [14]. Some experiments were also carried out in the laboratory to capture the performance of specific network elements such as EDFAs and filters. The performance of the proposed solutions was evaluated using these software tools and programming languages, and the obtained results were reported in selected publications (international conferences and journals) for the sake of dissemination.

1.4 Outline of the Thesis

This section overviews the thesis organization identifying the topics explored in the next chapters. **Chapter 2** provides the related background, and it serves as the baseline for the study of different physical layer impairments and the available models along with their effectiveness in the network performance estimation. It also gives the introduction on different ML and nonlinear fitting techniques adopted in the proposed solutions, presented in **Chapter 4** and onwards. In **Chapter 3**, we present the state-of-the-art

reviewing the produced scientific publications related to the topics addressed in this thesis. The specific contributions of this thesis start from **Chapter 4**. For each contribution we exhaustively describe the identified problem, mathematically formulate it, and present the proposed solution, followed with performance evaluation results. **Chapter 4**, **Chapter 5**, and **Chapter 6** also include the published works bound to those chapters. The final **Chapter 7** summarizes the results accomplished in this thesis and outlines the future work with some closing discussions.

Chapter 2

We describe the physical layer impairments that impact the performance of the optical transport networks. Moreover, we introduce the concept of physical layer modeling schemes as well as overviewed the available PLMs in the literature. It also provides a description of the mathematical model considered in this thesis. Specifically, we focus on the Gaussian Noise (GN) model for the non-linear interference noise (NLI) and the generalized signal to noise ratio (SNR) calculations. We also briefly address the concept of ML and nonlinear curve fitting techniques.

Chapter 3

This Chapter covers the state-of-the-art on QoT estimation and transponders (TPs) launch power adjustment techniques and algorithms. This Chapter also includes the approaches available in the literature for QoT estimation. It reports the scientific works used as the background for the main contributions of this thesis as explained in the subsequent chapters.

Chapter 4

We introduce the QoT estimation tool and the concept of design margin to embrace the PLM modeling inaccuracies and input parameter uncertainties. We present an extended version of the GN noise model to account for the EDFA gain ripple effect. Then, it is discussed supervised ML methods to learn and estimate the penalties caused by the EDFA gain ripple effect [15], [16]. To this end, we present ML models (schemes) to estimate the EDFA gain ripple penalties, which allow stating a trade-off between quantity of monitoring information and accuracy of these ML models [17].Additionally, we provide an independent ML model to estimate the filtering penalties due to uncertainties inside the ROADM nodes [18].

Chapter 5

In this Chapter, we propose PLM extensions needed for partial disaggregated optical networks with TPs from multiple vendors. Based on the proposed multi-vendor PLM, we show the performance improvements on two use cases *i)*. TPs launch power optimization [19], and *ii)*. incremental planning. On top of that, in this Chapter we propose a ML-assisted scheme aiming to train the multi-vendor PLM according to either the real network in planning or online operation phases.

Chapter 6

This Chapter focuses on a novel scheme to retrain the QoT estimation tool based on iterative closed control loop process. The proposed scheme allows solving a complex/multi-variable dynamic optimization problem. Particularly, we extended the

TPs launch power optimization problem of Chapter 5 for a dynamic operating network [20]. We also present a detailed discussion about the requirements in terms of monitoring calls and computational time (with and without monitoring errors). We do verify the additional savings in terms of monitoring calls and computation time with an extended version of this scheme. To this end, we extended the scheme to an adaptive number of optimization steps in addition to the iterative closed control loop formulation [21].

Chapter 7

This chapter concludes the thesis providing a discussion on the accomplished achievements through the conducted activities along with outlining the main conclusions. Additionally, we sketch the new research perspectives to be tackled in future works.

1.5 Sponsor

The **ONFIRE** (FUTURE **O**PTICAL **N**ETWORKS **F**OR **I**NNOVATION, **R**ESEARCH, AND **E**XPERIMENTATION) project funded the research described in this Ph.D. thesis [22]. ONFIRE is a Horizon 2020 (H2020) research and innovation program funded by the European Union (EU) under the Marie Sklodowska-Curie Actions (MSCA) grant agreement No. 765275. ONFIRE is a three-years research programme with two Early Stage Researchers (ESRs). The ESRs are benefited from an intensive training process combining the strengths of both: *i).* Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Castelldefels, Spain as research institution to acquire research tools and methodology, with Universitat Politècnica de Catalunya (UPC), Barcelona, Spain as associated partner offering its Ph.D. programme; *ii).* Nokia Bell Labs (Nokia), Stuttgart, Germany as a vendor delivering a highly valuable view of the research activities and its impact on industrial ecosystem. Targeted Ph.D. training programme for the ESRs is devised to maximize the synergy between the collaborators and promote career opportunities of ONFIRE researchers in the European Information and Communications Technology (ICT) research area. This Ph.D. thesis is focused on the research project carried out by ESR 2 of the ONFIRE project.

In general, ONFIRE tackles physical layer monitoring schemes, machine learning (ML) and data mining techniques to leverage massive physical layer monitoring information to adjust and re-optimize the network settings for aggregated (traditional) and next generation disaggregated optical transport networks. ESR 2 Ph.D. thesis focuses on novel methods and techniques for network optimization. One of the targeted optimization operation in ONFIRE project is the optimization of launch power of transponders to improve the overall network capacity. These methods and schemes are indeed proposed leveraging different aspects such as physical layer monitoring information, ML techniques, and other optimization algorithms and methods to design, update, and more efficiently manage optical networks. Specifically, the ML-based schemes are devised to achieve accurate modeling of the physical layer as well as their interaction with dynamic optimization algorithms in the context of optical transport networks.

1.6 Dissemination

This section contains the list of scientific publications produced during this Ph.D.

Peer Reviewed International Journals Publications

- **A. Mahajan**, K. Christodoulopoulos, R. Martínez, S. Spadaro, R. Muñoz, "*Modeling EDFA Gain Ripple and Filter Penalties with Machine Learning for Accurate QoT Estimation*", in IEEE/OSA Journal of Lightwave Technology (JLT), Vol. 39, No. 9, pp. 2616-2629, February 2020.

- **A. Mahajan**, K. Christodoulopoulos, R. Martínez, R. Muñoz, S. Spadaro, "*Quality of Transmission Estimator Retraining for Dynamic Optimization in Optical Networks*", in IEEE/OSA Journal of Optical Communications and Networking (JOCN), Vol. 13, No. 4, pp. B45-B59, February 2021.

- **A. Mahajan**, K. Christodoulopoulos, R. Martínez, R. Muñoz, S. Spadaro, "*Impact of Multi-vendor Transponders Performance on Design Margin in Optical Networks*", in IEEE Access [manuscript under review].

Contributions to International Conferences

- **A. Mahajan**, K. Christodoulopoulos, R. Martínez, S. Spadaro, and R. Muñoz, "*Machine Learning Assisted EDFA Gain Ripple Modelling for Accurate QoT Estimation*", in Proceedings of 45th European Conference on Optical Communication (ECOC), Dublin, Ireland, September 2019.

- **A. Mahajan**, K. Christodoulopoulos, R. Martínez, S. Spadaro, and R. Muñoz, "*Modeling Filtering Penalties in ROADM-based Networks with Machine Learning for QoT Estimation*", in Proceedings of Optical Fiber Communication Conference (OFC), San Diego, California, United States, March 2020.

- **A. Mahajan**, K. Christodoulopoulos, R. Martínez, S. Spadaro, and R. Muñoz, "*Improving QoT Estimation Accuracy with DGE Monitoring using Machine Learning*", in Proceedings of 24th International Conference on Optical Network Design and Modelling (ONDM), Castelldefels, Spain (Virtual Event), May 2020.

- **A. Mahajan**, K. (Kostas) Christodoulopoulos, R. Martínez, S. Spadaro, and R. Muñoz, "*Modelling Multi-Vendor Transponders Performance and Optimizing Launch Power*", in Proceedings of 46th European Conference on Optical Communication (ECOC), Bruxelles, Belgium (Virtual Event), December 2020.

- **A. Mahajan**, K. Christodoulopoulos, R. Martínez, R. Muñoz, and S. Spadaro, "*Adaptive and Iterative QoT Estimator Retraining for Launch Power Optimization*", in Proceedings of Optical Fiber Communication Conference (OFC), San Francisco, California, United States (Virtual Event), June 2021.

Chapter 2

Background

In this chapter, we give a quick overview of optical networks and recent advances in the field, such as disaggregation, models for different noise sources, and ML related applications. In addition, we also discuss the main sources of noise and optical impairments present on the physical layer that affects the optical signal quality. Then, we provide a brief introduction about the mathematical and analytical models to account for these noises.

This chapter is organized as follows. In Section 2.1, we introduce optical networks and the concept of disaggregation along with its possible levels. In Section 2.2, we briefly overview the optical impairments and noise sources in the physical layer. Starting from Section 2.3 to Section 2.5, we focus on the available models that can be used to mathematically estimate and quantify the noises as described in Section 2.2. In the same sections, we also address the related concept of design margin. Section 2.6 provides a brief overview about the candidate SDN architecture to integrate with the schemes proposed in this Ph.D. thesis. In both Section 2.7 and Section 2.8, we overview the concept of ML providing the mathematical details for some supervised ML algorithms adopted in this thesis. Since Chapter 5 and Chapter 6 rely on nonlinear curve fitting techniques, Section 2.9 gives the mathematical details of some nonlinear curve fitting algorithms.

2.1 Introduction to Optical Networks

The basic aim of any network is simple: enabling the remote communication between the specific endpoints. That said, the underlying characteristics of the used network influence on the overall obtained performance. The network capacity, cost, flexibility, scalability, operation simplicity and reliability are some of the key indicators to assess a network [23]. Network designers often face up tradeoffs among these factors and are continuously seeking for technological advances to make the networks more efficient.

One such development came in the 1980s as the telecommunications industry moved much of the physical layer of their inter-city networks to fiber-optic cable [24]. Optical fiber is a lightweight cable that provides low-loss transmission with remarkable networking capacity. Fiber optics not only opened the possibilities of huge data transmission capabilities, but also gave rise to the generic concept of optical networking. In a nutshell, Optical Networks can be defined as the communication networks used for

the exchange of information through an optical fiber cable which nowadays constitute the enabling backbone technology of today's Internet.

Over the past 50 years, optical networks have advanced rapidly both in transmission and networking [1], [24], [25]. One of the major leaps in optical communications has been represented by the evolution from point-to-point communication, where data is transmitted between only two nodes, towards full meshed networking, where optical channels (also referred to as lightpaths) are added, dropped, and routed over multiple network nodes. Current optical networks architectures provide high-capacity backhauling to upper network layers, such as the widely used IP data traffic. These optical networks are also managed/controlled by well-devised intelligent functionalities to realize complex network operations such as efficient routing, grooming, protection, fast restoration etc. However, all these features and capabilities are backed up by the advancements in photonic technologies that provide capabilities to handle such challenging features in an efficient manner. The performance of optical networks are tightly related to the evolution and advances on the designs in optical fiber, transmission and photonic technologies etc. which are the foundation of these networks. Generally speaking, optical networks are made up of two planes: *i).* data plane and *ii).* control plane. The data supports the transportation of the data information between different end-nodes; while the control plane takes over of the configuration and management control functions over the transport infrastructure.

A key technological advance in the optical networks was the ability to carry multiple channels of on a single optical fiber. Each channel is carried at a different wavelength or frequency and multiplexed onto a single optical fiber, giving rise to WDM [26], [27]. The State-of-the-art in backbone core and metro optical networks segments rely notably on high-capacity optical transmission links utilizing WDM technology. Besides the demand for high capacity, optical transport networks desire long un-regenerated optical transmission distance to effectively support metropolitan, regional, and national network applications. A key enabler of these cost-effective WDM systems was the development of the EDFA [28]. The EDFA optically amplifies all the wavelengths on a fiber at once, removing the barriers on costly opto-electrical conversions. In the 1990s, commercial WDM systems experienced an aggregate per-fiber capacity growth at a 100% compound annual growth rate (CAGR), i.e. doubling capacity year by year. After the year 2000, such growth has rapidly decreased to a 20% CAGR indicating that the number of WDM channels in commercial systems have been constant for almost two decades (close to 100 channels) [27], [29]. However, the introduction of C+L band systems, which exploit close to 200 WDM channels, has sparked interest, and commercial systems with these characteristics are nowadays available.

In terms of transmission technology, more specifically transceivers, data rates per channel advanced rapidly over the past 50 years. The typical single-channel data rate of fiber-optic transmission increased from 2.5 Gb/s in 1989 to nearly 1.0 Tb/s nowadays, i.e., over 100 times. The main technologies include high-speed electro-optical modulation, high-speed optical detection, hard-decision forward error correction (HD-FEC), coherent optical detection assisted by digital signal processing (oDSP), soft-decision FEC (SD-FEC), polarization-division multiplexing (PDM), high-order quadrature amplitude modulation (QAM), constellation shaping (CS) such as

probabilistic CS (PCS), advanced oDSP such as faster-than-Nyquist detection, and 100 Gbaud-class high-speed opto-electronic devices. With the introduction of superchannel technology, the aggregated channel rate can even exceed 1 Tb/s [25]. These modern transceivers/ transmission capabilities allowed a relevant simplification of optical line systems leading to the removal of in-line dispersion compensating units, allowing a relevant complexity reduction in link design and optical infrastructure management.

The other breakthrough advancement in optical networking architectures is represented by the introduction of add, drop and routing capabilities of optical channels at individual network nodes [30], [31]. The optical networks do not stay indefinitely in the configuration defined during the network planning phase. Network configurations may change during the planning or upgrade phase due to either traffic change (usually an increase with time) or link/node failures. The network elements that enabled such dynamicity in optical networks are optical add drop multiplexers (OADMs). OADMs were first constructed from optical filters, allowing for independent (but fixed) wavelengths to be terminated at network nodes, or to bypass them. OADMs are based on prearranged and fixed optical configurations, thus one of their main disadvantages is their limited adaptability with respect to the variation in traffic and channels configuration. To solve this issue, ROADM with WSS technologies were proposed. The use of ROADMs is seen as essential offering a pursued degree of freedom to attain more agile, flexible and dynamically reconfigurable optical networks.

2.1.1 Disaggregation Flavors

The traditional optical transport networks have historically been closed and proprietary systems. The hardware forming an optical network infrastructure is diverse including optical transponders, optical amplifiers, optical switches, wavelength multiplexers/demultiplexers, WSSs, and gain equalizers. Such network elements are coupled together and the control and management functions and operations are tightly tailored to such hardware as well. Due to that, numerous efforts have been made to open up the transport networks and thus move away from closed proprietary systems. Despite these efforts, all of the optical subsystems still need to be co-designed and optimized to get the best system performance, resulting in little progress toward open and disaggregated optical systems in the past [32]. **Figure. 2.1** depicts a traditional single-vendor closed line system, where a vendor-provided controller is responsible for the management of both the line system and the terminal equipment. This scenario illustrates the so-called vendor lock-in, where operators do not have much options/opportunities for the deployment of their custom controllers since, they need to rely completely on the vendor provided control software. Consequently, this ends up on an overall rigidity in the design and management of optical networks.

Recently, open and disaggregated optical networks have regained the interest and attention of the industry. The disaggregation of the optical transport constitutes a viable way to lower the cost by many network operators. The development and actual practices of SDN control in the past decade and its successful deployment in real networks have granted network operators with the experience, and confidence to invest on open network technologies. SDN which decouples the data and the control planes is seen as a

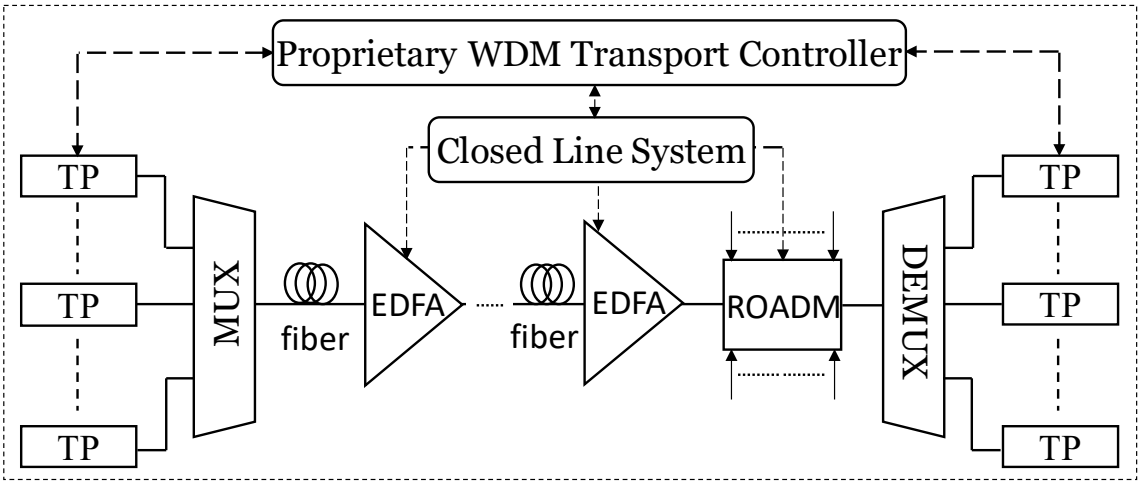


Figure. 2.1. Traditional closed line systems in optical networks with single vendor integration.

key driver to foster the network disaggregation. The term *disaggregation* in the context of WDM transport network is often used to collectively designate all the operational models in which telecom operators are actively involved in the design, assembly, testing and lifecycle management of the WDM transport systems deployed in their networks. Disaggregation in optical networks has attracted significant telecom operator's interest since it overcomes the vendor lock-in, leading to potential capital expenditure (CAPEX) saving.

The concept of disaggregation in optical networking is brought from the datacenter's architecture. Datacenters are built upon interchangeable, highly flexible computing and network nodes [33]. This flexible datacenter architecture approach, fuels optical networking to explore multi-vendor disaggregating hardware and software with a strong focus on interoperability. In this context, substantial effort has been placed in developing vendor-neutral software controllers, third party network orchestrators, and northbound/ southbound open control interfaces. Moreover, some works demonstrated line systems that are open to multiple vendor network elements [10], [34]. Such developments have an ultimate objective to enable an open or disaggregated line system (OLS), where the optical hardware from multiple vendors can be interconnected and controlled centrally through a common vendor-neutral control plane. Thus disaggregated optical transport system is considered to accomplish higher flexibility and cost reduction. Therefore, telecom operators agree that disaggregation approach will bring notable savings that eventually could make the difference in the upcoming years. Basically, the disaggregation allows network operators composing individual network nodes, by selecting the most appropriate vendor solutions for each function, for e.g., transponder, ROADM, line system, control, monitoring etc. [35]. At least two flavors of disaggregation can be considered at the optical layer: *partial* and *full* disaggregation. The former being much more attractive than the latter, which however is actively supporting standardization initiatives.

- Partial Disaggregation

The partial disaggregation flavor in optical networks, considers the optical transport infrastructure (e.g., OLS including ROADMs and amplified links) managed by a single vendor, while transponders (TPs), provided in pairs, rely on vendor neutral control such as NETCONF/YANG as shown in **Figure. 2.2** In short, in this disaggregation approach, the line system is open (OLS) to TPs from *M* multiple vendors as shown in **Figure. 2.2**.

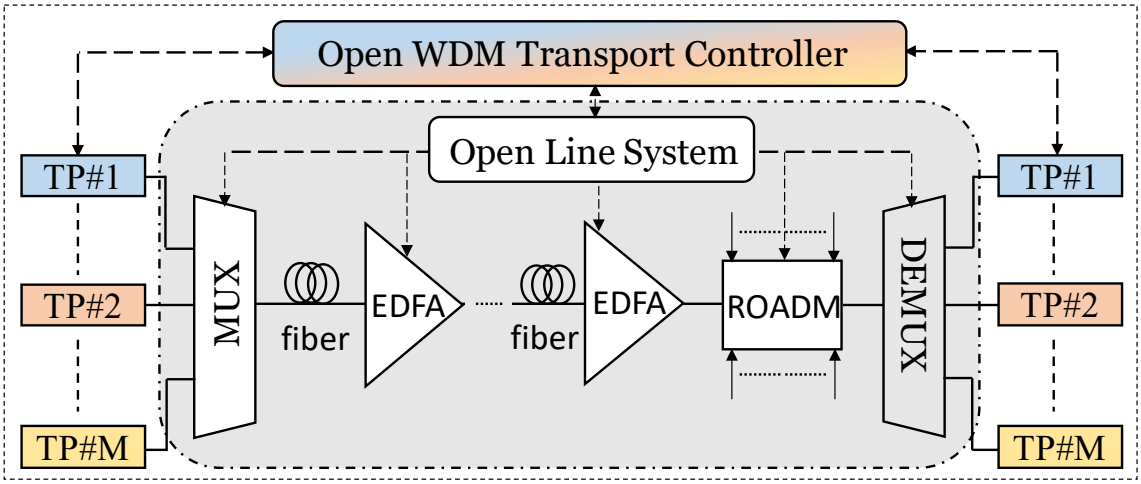


Figure. 2.2. Open line system (OLS) approach with TPs from multiple vendors.

The rationale behind this approach is that the operational life of the line system is much longer than that of TPs, whose useful life is generally governed by the continuous increase in capacity needed, requiring a very strong pace of innovation and therefore obsolescence. Furthermore, the multi-vendor TP environment provides the telecom operators the freedom to choose the best supplier for each specific application favoring from time-to-time performance, cost or other metrics.

- Full Disaggregation

The second flavor, called full disaggregation, further disaggregates the OLS, considering ROADMs as white boxes to be controlled in a vendor-neutral way. In broader terms, in full disaggregation approach, every single optical component (or subsystem) could be selected from multiple vendor which exposes its programmability through open application programming interfaces (APIs) as shown in **Figure. 2.3**. In case of fully disaggregated systems, telecom operator has the freedom to choose even subsystem blades (each with different functionality), possibly from different hardware suppliers, within in a ROADM architecture.

So far, partial disaggregation has attracted significant interest from operators and vendors, since it neglects most of the optical data plane complexity without significantly compromising on transmission performance. The business model that drives partial disaggregation considers that the optical transport (i.e., OLS) is a mature technology that, once deployed, can last for more than ten years without relevant upgrades. On the other hand, transmission technology (i.e. TPs) is evolving at extremely high pace, with impressive and continuous increase of the symbol rate. In some network scenarios, to cope with the continuous increase of data traffic, TPs can be replaced with higher rate versions even every three years [32]. Thus, partial disaggregation enables telecom

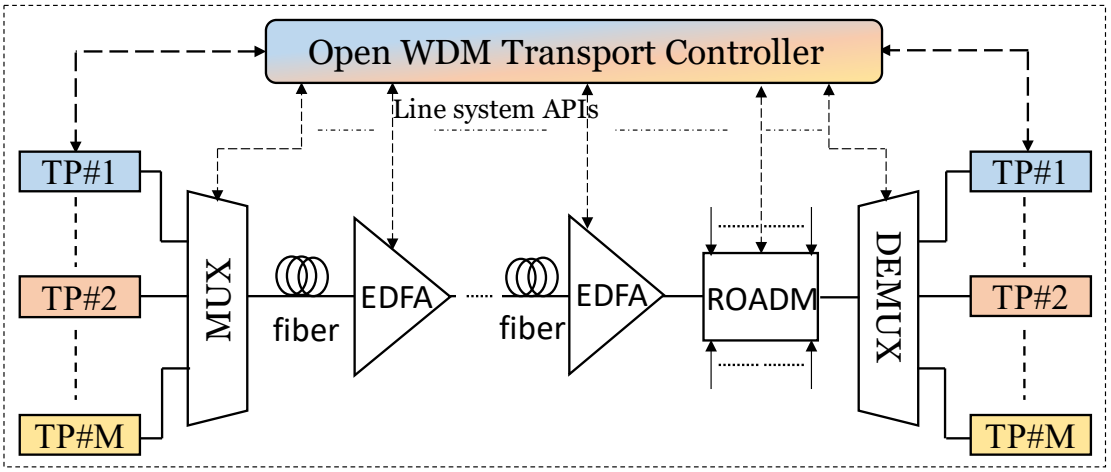


Figure. 2.3. Full disaggregation approach with open APIs.

operators not to be bounded to a single vendor for new TP deployments, still enabling a single vendor to have full responsibility of the OLS. On the other hand, the full disaggregation approach is more complex and appears to be less evident at the moment, and potentially beneficial only as a long term approach. However, some initiatives such as Open ROADM [36], OpenConfig [37] etc. are currently working on defining and implementing multi-source agreements (MSA) for such full disaggregation based optical whiteboxes.

2.2 Optical Physical Layer Impairments

The optical fiber is often seen as a perfect transmission medium with almost limitless bandwidth, but in practice the propagation through optical fiber is limited as distance is increased to multi-span amplified systems. Optical signals that propagate along optically amplified multi-span fiber links undergo several physical layer impairments (PLIs) that limits their capability of carrying information in terms of reach and signal quality. Some of these noise sources are generated by the TPs itself such as thermal, quantization, shot, laser phase noise etc. and are stochastic in nature. However, keeping these noises aside, the root cause of transmitted signal quality degradation is the Amplified Spontaneous Noise (ASE) noise generated by the optical amplifiers and noise generated due to fiber nonlinearity, more specifically Nonlinear Interference noise (NLI) causing signal distortion [38]. However, the signal quality also degrades after traversing ROADM nodes due to filtering and bandwidth narrowing effect. **Figure. 2.4**. highlights the sources and location of these noises or propagation impairments along the signal path.

All propagation effects that depends linearly with the electric field are classified as *linear* ones. Linear transmission impairments mainly include ASE noise generated due to optical amplifiers, attenuation, chromatic dispersion (CD), polarization mode dispersion (PMD) and polarization dependent loss (PDL) [38], [39]. CD is the dependency of the group velocity of the signal and its optical frequency, PMD is a dependency of the group velocity of the signal and its polarization state whereas the PDL is a random dependency between signal attenuation and the polarization state. Additionally in WDM systems linear noise arising from filtering and crosstalk effects at optical filters/de-multiplexers and WSS inside the ROADM nodes also affects the quality of the propagating signal

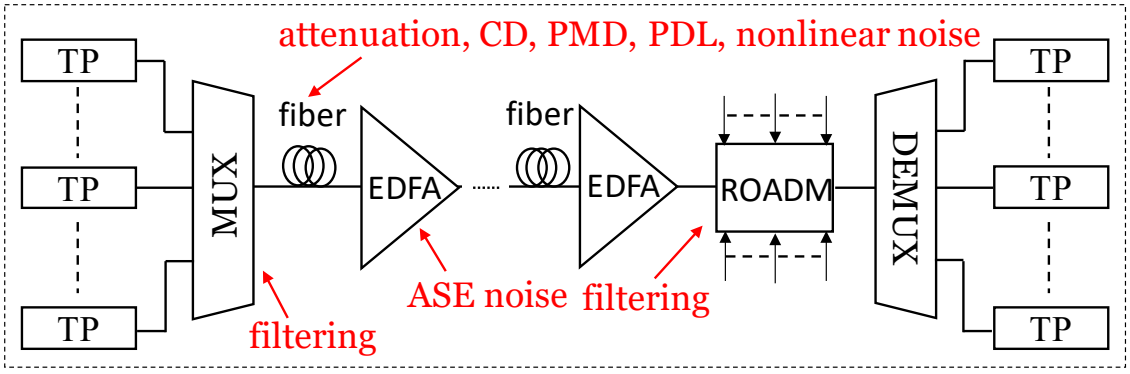


Figure. 2.4. Main sources and location of noise and impairments generated in an optical link.

[40], [41]. In general, all these linear effects such as dispersion and filtering cause signal pulses to spread in time, yielding inter-symbol interference (ISI) that degrades the signal quality. However, with the advancement of optical coherent technology and digital signal processing (DSP) techniques, most of these effects can be compensated at the receiver end [42]. The losses due to the fiber inherent attenuation can be easily compensated with the amplifier but with the accumulation of additional ASE noise.

On the other hand, *nonlinear* effects in transmission system occur either due to electric field or intensity dependency of refractive index of the medium or due to inelastic-scattering phenomenon. For easy understanding these effects are generated/produced when signal with high optical power is injected in the transmission system. Kerr effect induced as Self-Phase Modulation (SPM), Cross-Phase Modulation (XPM) and Four-Wave Mixing (FWM), Stimulated Raman Scattering (SRS), and the Stimulated Brillouin Scattering (SBS) are some of the main nonlinear effects that degrades the signal quality during propagation [38]. More specifically, the Kerr effect results in signal phase distortion due to the dependency of the refractive index of the fiber with respect to the power of the optical signal propagating through it. Due to fiber dispersion, such phase distortion is converted into amplitude distortion causing substantial penalties in the signal quality. Furthermore, the Kerr effect imposes a limitation on the launch power of the signal that needs to be transmitted into an optical fiber link. To this purpose, proper launch power optimization strategies are needed to maximize the signal Quality of Transmission (QoT). As the Kerr effect is one of the most prominent nonlinear effect, the estimation of its impact on the signal quality and the derivation of the optimal launch power strategies is already an active field of research over the last couple of decades. Several models have already been presented in order to accurately estimate the nonlinear noise due to Kerr effect and we will briefly review them in the next section. We will also highlight the most relevant model used throughout this thesis.

The nonlinear scattering effects such as SRS and SBS in transmission systems, occur due to thermal molecular vibrations within the transmission media i.e. optical fiber. Both SRS and SBS are related to vibrational excitation modes of optical fiber (silica). The fundamental difference between these two effects is that optical phonons participate in SRS while SBS is through acoustic phonons. As a result of this difference, SBS occurs only in one direction i.e., backward while SRS can occur in both directions-forward and

backward. SRS is a wideband effect, as the scattering process is significant over spectral occupations of the order of few THz. In optical systems, SRS causes a power transfer between lower wavelength channels to higher wavelength ones. This results in a tilt over the power spectrum of the WDM channels propagating in the system. As SRS efficiency in fiber (silica core) is very high for signals with a wavelength separation of approximately tens of nm [38], it is a relevant effect for WDM systems operating over the full C-band and also for next generation C+L band WDM systems. On the other hand, SBS is a narrow-band effect that only affects a single channel causing power depletion through the generation of a counter-propagating wave inside the fiber. Nevertheless, SBS is a weak effect for modern communications optical systems where optical signals have in general a spectral occupation much larger than the one for which SBS has peak efficiency. Furthermore, modern multilevel modulation formats are carrier suppressed, which further helps in reducing the SBS.

This thesis, however, excludes SRS and SBS because we only deal with C-band transmission systems with launch power values well below the system's nonlinear noise threshold.

2.3 Available Physical Layer Models

We will give a brief mathematical description of PLMs available in the literature for estimating various optical impairments such as ASE noise, NLI noise, filtering penalties, and so on in this section. These models will also serve as the foundation for our simulations in the following Chapters.

2.3.1 Model for ASE Noise

ASE noise is one of the most common and main source of impairment in optical communication systems. ASE noise is produced by the spontaneous emission, that has been optically amplified by the process of stimulated emission in a gain medium. In particular, in optical systems, ASE noise arises from spontaneous emission of the gain medium in an optical amplifier [43]-[45]. In case of EDFAs, it is possible to model the ASE noise as an additive white Gaussian noise (AWGN) with bilateral power spectral density (PSD) including both polarization given by

$$G_{ase} = 2\,h\;v\,n_{sp}\,(G-1) \tag{2.1}$$

where $h = 6.626 \times 10^{-34}$ J.s is the Planck's constant, v is the center frequency of the signal being amplified, G is the average gain of the amplifier (linear units) and n_{sp} is the spontaneous emission factor ($n_{sp} > 1$). n_{sp} is directly related to the population level in ground and excited state (or population inversion) of the Erbium atoms. It specifies the noise performance of the amplifier. From a system perspective, noise performance of an amplifier is often expressed using the noise figure (NF) parameter F, that is given by

$$F = 2\,n_{sp} \tag{2.2}$$

Eq. (2.1) can be easily expressed in terms of Eq. (2.2) as

$$G_{ase} = h\;v\,F\,(G-1) \tag{2.3}$$

Besides EDFAs, in wideband long-haul transmission systems, Raman amplifiers (or hybrid i.e., EDFAs and Raman) are also widely used. In comparison to (C+L) band EDFAs, the main benefit of Raman amps is the long band amplification with improved noise performance. However, in this thesis we only assumed the use of EDFAs (for C-band transmission) that generate ASE noise according to Eq. (2.3).

2.3.2 Model for NLI Noise - Gaussian Noise (GN) Model

In this thesis we consider traditional WDM optical networks with no dispersion compensation and optically amplified links in the C-band. This transmission scenario is well-aligned with the current optical networks and it has been extensively shown that the impairments of fiber propagation due to fiber loss (attenuation), dispersion (CD and PMD), and Kerr nonlinearity (as presented in Section 2.2) can be approximated as a small perturbative additive Gaussian disturbance on any single frequency [46], [47]. This effect is well known in the literature as nonlinear interference (NLI) effect. In general, NLI contains a phase-noise component [48]. However it is well proven that, under normal operating conditions (phase noise has a long correlation [49]-[50]) this component can be completely (almost) compensated at the phase recovery circuitry of the standard coherent receivers. This backs up for the assumption of modeling NLI as an additive white Gaussian noise (AWGN). These fundamental results triggered the development of the series of Gaussian noise models in the optical networks, that are basically analytical perturbation models for the NLI noise estimation. These models obtain approximate solutions for the nonlinear Manakov Schroedinger equation [38] based on the simplified assumption that each WDM channel can be treated as a Gaussian noise spectrally shaped as the signal [50]. This approach to model nonlinear effects was firstly proposed back in 1993 [51] and adapted up to early 2000s [52] with subsequent modifications to model nonlinear Kerr effect in IM-DD systems. Based on the same approach, series of GN models were proposed for coherent optical systems after 2010 [46], [49], [50]. In this thesis, the GN model is used to account for the nonlinear noise to perform more realistic simulations [50].

Let us now assume that link m with equally spaced N_{ch} WDM channels as shown in **Figure. 2.4**. For channel n centered at wavelength λ_{ch}, we assume transmitted power P_n and symbol rate, R_n. In general, with incoherent noise accumulation assumption, the PSD of NLI noise at link end can be calculated as the sum of the NLI noise produced in each single span. With the assumption of noise to be additive Gaussian [50], the GN model calculates PSD of NLI noise at link end, as

$$G_{nli,m}(\lambda_{ch}) = \frac{16}{27}\sum_{n_s=1}^{N_s} \gamma_{n_s}^2 L_{eff,n_s}^2 \cdot \sum_{n=1}^{N_{ch}} G_{n,n_s} G_{n,n_s} G_{c,n_s} (2-\delta_{nc})\, \Psi_{\lambda_n,\lambda_{n,n_s}} \cdot \prod_{n_s'=1}^{n_s-1} g_{n_s'}^3\, e^{-6\alpha_{n_s'} L_{n_s'}} \cdot \prod_{n_s'=n_s}^{N_s} g_{n_s'}\, e^{-2\alpha_{n_s'} L_{n_s'}} \quad (2.4)$$

where Ψ is the phased array factor which under the assumption of incoherent accumulation is given by Eq. (128) and Eq. (129) of [50]; L_{n_s} is the n_s span length; γ_{n_s} is its non-linear coefficient (1/W/km); L_{eff,n_s} is its effective length; α_{n_s} is its attenuation coefficient (1/km), g_{n_s} is the gain of the span EDFA; $G_{n,ns}$ is the power spectral density

(PSD) in W/Hz of then n-th WDM channel (n=1, 2… , N_{ch}) at the start of the n_s-th span; δ_{nc} is the factor that distinguishes the self-channel interference (SCI) and cross-channel interference (XCI) terms as given by Eq. (122) of [50]. The detailed derivation of Eq. (2.4) along with parameters description are available in [50].

2.3.3 Contribution of Filtering Penalties

Whenever the transmitted signal traverses a ROADM node, it encounters filtering penalty due to the WSSs that reside within that ROADM node (**Figure. 2.4**). Filtering penalties is the additional limiting factor to the capacity of the ROADM based networks as the ones considered in this thesis. Filtering penalties become more relevant as the optical channels traverse multiple ROADMs along their path. This results in bandwidth narrowing of the propagating signals. Finally, bandwidth narrowing causes inter symbol interference (ISI) at the receiver (destination node). Modeling such bandwidth narrowing/filtering induced penalties in general is nontrivial. This is because, these penalties depend upon many deterministic parameters such as filter and signal spectral shape, modulation format, buadrate, interchannel spacing and on the DSP receiver structure [40], [53].

In the literature, these filtering penalties are often measured experimentally depending upon the above parameters. Since a general model for filtering penalties is nontrivial to derive, in optical networking filtering penalties have often been neglected or taken into account in a simplified way, such as imposing a maximum tolerable number of cascaded ROADMs that a signal can traverse[40], [41] or by considering a fixed SNR penalty term for each traversed ROADM. Many studies also considered these penalties in terms of exponential functions with the number of traversed ROADM nodes or WSSs as the independent variable.

2.4 The QoT Metric - SNR and OSNR

In optical networks, many different metrics are available to estimate or measure the QoT of the optical signal. Some of these matrices are in the optical domain such as the optical signal to noise ratio (OSNR), CD, PMD etc. and other are in the electrical domain such as the electrical Signal to Noise Ratio (SNR), Bit Error Ratio (BER), Quality (Q)- factor etc. In the legacy optical networks with IM-DD systems with dispersion compensation modules, the most widely used QoT metric was the Q-factor. The Q-factor is directly related to the BER of the signal as given by

$$BER = \frac{1}{2}\ erfc\left(\frac{Q}{\sqrt{2}}\right) \tag{2.5}$$

BER and the Q-factor are the most widely used QoT performance metric parameters in the last few decade [54].

However, with the advancement in the coherent technology, the reference metric for assessing the QoT of a signal moved to the SNR. In coherent digital transmission systems, for channels impaired by AWGN, SNR is defined as the ratio between the average power of the output demodulated (electrical) signal at the receiver P_{Rx} to the average noise power P_{noise_Rx} impairing it as given by

$$SNR = \frac{P_{Rx}}{P_{noise\_Rx}} \tag{2.6}$$

In the above Eq. (2.6), both the signal and noise powers are generally evaluated in the digital domain at the receiver (after DSP such as equalization, down/up sampling, matched filtering etc.). However, the current coherent receiver's performance is almost near to ideal, hence there is a direct one-to-one mapping between the optical (analog) domain to electrical (digital) domain. Due to this reason, the SNR is often related with the *optical* SNR (OSNR), which is the ratio between the optical signal PSD to the total noise PSD. One of the main points to be noted here is that OSNR is expressed with respect to a specific noise bandwidth (BW) denoted by R_{noise}, whereas the SNR by default refers to the BW (equal to the symbol rate, R_S) of the matched filter used for its demodulation. A well accepted reference noise bandwidth chosen for OSNR calculations is 0.1 nm or 12.5 GHz. This offers a very straightforward and quick relationship between these two parameters (of different domain) as given by

$$OSNR = \left(\frac{R_S}{R_{noise}}\right) SNR \tag{2.7}$$

In this thesis, SNR is assumed as the reference QoT metric for all the upcoming Chapters. Note that SNR is directly related to the BER, which is the QoT metric that represents the quantity of information that can be transmitted over the network with respect to the number of bits in error. It is also assumed that all noise-related propagation impairments can be modeled as AWGN as described in Section 2.3. In particular, such assumptions holds pretty well for both ASE noise and nonlinear disturbances generated by the Kerr effect (NLI noise) calculated with the previously discussed so called GN model.

2.4.1 Generalized SNR

The most important propagation impairments for dispersion uncompensated coherent optical communication systems are the ASE noise and the NLI noise, as presented in the previous sections. Both noises can be represented as additive Gaussian sources of disturbances to the transmitted signals. This property led to the definition of the generalized (or effective) SNR, that is a QoT metric which includes both ASE and NLI noise. One of the benefit of this QoT metric is the direct evolution of the signal quality, equivalent to electrical SNR, with the assumption of perfect DSPs for impairment compensations [50]. The generalized SNR is given as

$$SNR = \frac{G_O}{G_{ase} + G_{nli}} \tag{2.8}$$

where G_O is the optical signal average PSD, G_{ase} is the ASE noise PSD, and G_{nli} is the NLI noise PSD contribution.

Eq. (2.8) can be further break down to Eq. (2.9) as

$$SNR = \left(\frac{G_{ase}}{G_O} + \frac{G_{nli}}{G_O}\right)^{-1} = \left(\frac{1}{SNR_{linear}} + \frac{1}{SNR_{NLI}}\right)^{-1} \tag{2.9}$$

where SNR_{linear} and SNR_{NLI} provides a quantitative measure of how much the SNR is degraded due to ASE and NLI generated in the considered fiber span.

If we consider an optical signal transmitted through a multispan link with N_S number of spans, assuming incoherent accumulation of NLI noise, the total SNR can be calculated as

$$SNR = \left(\sum_{n_s=1}^{N_s} \frac{1}{SNR_{linear,n_s}} + \sum_{n_s=1}^{N_s} \frac{1}{SNR_{NLI,n_s}} \right)^{-1} \quad (2.10)$$

Note that Eq. (2.10) provides a fast and efficient way (without compromising much of accuracy [50]) to estimate the SNR. In the upcoming Chapter of this thesis, we will use Eq. (2.10) as QoT metric for SNR calculations.

2.4.2 LOGO Strategy

From Eq. (2.10), for a multispan link, it is possible to figure out the optimal power at the input of every single span independently from the characteristics of all the other spans of the link. This is because from Eq. (2.8) – Eq. (2.10), the optimal power of the span (mostly) depends only on the ASE and the NLI noise generated in that span. Eq. (2.10) became the foundation to optimize (or maximize) the SNR (or QoT metric) of a multispan link by locally optimizing each span of the link. The approach is well-known, and referred to as Local Optimization lead to Global Optimization (LOGO) [55].

As an incoherent noise accumulation assumption is made, it is possible to characterize every span of an optical network by a noise-related quantity that expresses the amount of QoT degradation introduced by the span itself. Based on this, Eq. (2.9) and Eq. (2.10) can be adopted as inverse generalized SNR (or noise to signal ratio), that can be defined as

$$NSR = \frac{1}{SNR} = \frac{1}{SNR_{linear}} + \frac{1}{SNR_{NLI}} \quad (2.11)$$

Extending LOGO approach in a transparent optical network, let's consider a channel is propagating from node **a** to node **c** of a 4 node topology as shown in **Figure. 2.5**. In this case, the generalized SNR of the channel is simply given by the inverse of the summation of all SNR degradation terms of all the spans making up the path, given by

$$SNR_{ac} = \left(\sum_{n_s=1}^{N_s} NSR_{n_s} \right)^{-1} \quad (2.12)$$

where N_s is the number of spans making up the path from node **a** to node **c**, each of which is characterized by an SNR degradation term NSR_{n_s}.

Now assume to compute the QoT of a channel from node **a** to **c**, along the path **a-d-b-c**, one can simply compute it as:

$$SNR_{ac} = (NSR_{ad} + NSR_{db} + NSR_{bc})^{-1} \quad (2.13)$$

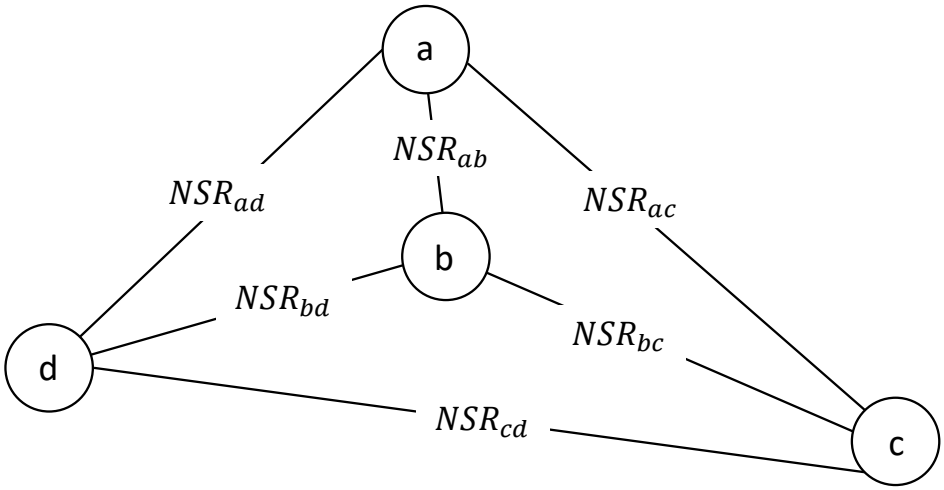


Figure. 2.5. Sample network topology for representation of LOGO strategy for QoT calculations.

LOGO power optimization strategy is feasible on the above Eq. (2.13) to maximize the SNR at the final node, **c** [55]. This strategy states that the global optimization of the SNR (at node **c**) over a generic network can be achieved by locally maximizing the SNR or minimizing the noises (ASE and NLI) at either each span or node to node level across the traversed path. Mathematically, this statement implies to minimize the weight of each link, which is the NSR in the equivalent graph representation of the network topology (**Figure. 2.5**). The LOGO model targets on maximizing the SNR at the destination node by maximizing each span's SNR (or minimizing each span's NLI noise). The main benefit of this model is its flexibility to optimize the power of each span irrespectively of the span lengths (homo or heterogeneous). However, the LOGO assumes full load and the same power levels for all channels (at each span) and thus does not consider the connection distance.

2.5 PLM Inaccuracies and Design Margin

As presented in Chapter 1 that QoT estimation (from the previously discussed metrics) is performed using an analytical PLM [12], [56]. The PLM is generally an analytical parametric model that takes as input several parameters such as fiber and EDFA specifications, information about the channel state in the network such as path, modulation format, assigned central wavelength, buadrate etc. Eq. (2.1) – Eq. (2.13) represents one such PLM with parameter vector, θ such as signal power, noise power, NF of EDFA and so on. The parametric PLM will then estimate the QoT (SNR) for the required signal but only to a certain degree of accuracy. In general, the main noise sources accounted in these PLM calculations (more specifically, GN model based PLMs [50]) are the ASE noise generated at both span and node amplifiers and the NLI noise, which considers fiber non linearities, mostly self- and cross-channel interferences (i.e., SCI and XCI). **Figure. 2.6** represents the schematic of such parametric PLM scheme for QoT estimation.

In general, a lot of simplifications are done in these PLMs to make them faster at the expense of degrading the accuracy. In particular, the speed bottleneck is the modeling of nonlinear effects, for which approaches range from the split step Fourier (SSF) method

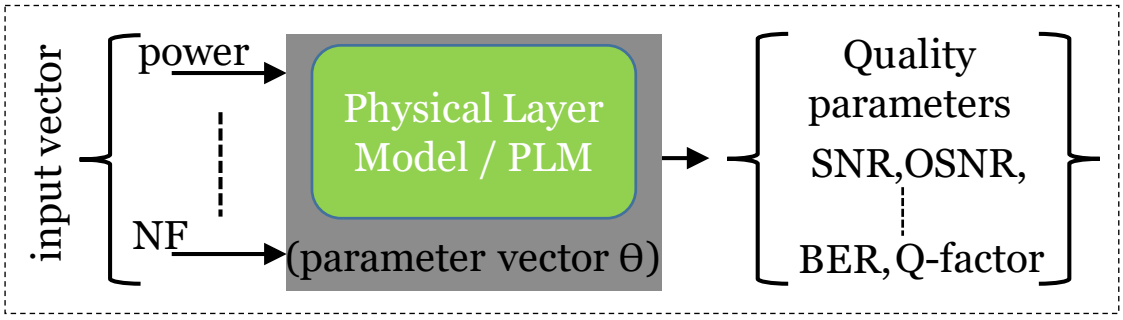


Figure. 2.6. Schematic for a parametric PLM based QoT estimation tool.

to analytic models; the SSF method is very accurate and versatile as it can address complex scenarios including the mix of non-coherent and coherent signals in networks with dispersion management; however, the trade-off is with speed, as SSF is very slow. As already discussed in the last Section 2.3 and Section 2.4, the (coherent) Gaussian noise model [50] is much faster and is accurate only within its application domain, which does not include, for instance, lines with dispersion management; an approximation of the coherent Gaussian noise model is the incoherent Gaussian noise model [50], which is faster but less accurate (for shorter link lengths) than the coherent version. The other major source of inaccuracy for these parametric PLMs comes from the inaccurate input parameters. Several reasons exist for this inaccurate input parameters such as outdated information about the fiber specification, tolerance values in EDFA parameters such as NF, gain etc. To account for the modeling simplification assumptions and other uncertainty parameters, a common practice is to add a design margin on top of the PLM calculations [57]. For a better understanding, consider Eq. (2.13) as PLM with a design margin of ~2 dB as a Qtool [8]. However, adding such a design margin results in less accurate QoT estimation. PLM and the design margin are commonly used to estimate the signal's QoT, also referred to as a QoT estimation tool or Qtool.

Note that removing such uncertainties would allow increasing the estimation accuracy of the Qtool and an equivalent reduction in the margins. This lead to attain higher efficiency and/or lower cost during network planning, upgrading and optimization operations. Majority of this thesis tackles this particular issue of QoT estimation accuracy and design margin reduction in the upcoming Chapters.

2.6 Introduction to Software Defined Networks Architecture

SDN is defined as a logically centralized control framework aiming at supporting the programmability of the underlying network elements and devices (e.g., optical switches, transceivers, etc.) by decoupling the data plane from the control plane. To this end, SDN exploits the advantages of supporting a generic programmability (i.e., configuration) of the underlying transport infrastructure relying on the utilization of open and well-defined control interfaces at both southbound interfaces (SBI) and northbound interfaces (NBI). For the sake of completeness, SBI supports the control communications between the SDN controller and the transport elements. The NBI allows the communication between specific applications (e.g., on-demand optical service app) and the SDN controller. The basic SDN control functions cover the maintenance of a transport network infrastructure information view, the computation and selection of

resources for accommodating new transport connectivity services, the connection lifecycle management involving the establishment, re-optimization/restoration and deletion [58]. Thus, the SDN controller takes over of the centralized management and control hosting the required network intelligence to come out with accurate decisions (e.g., resource selection) at the time of realizing network configurations and optimization.

The use of SDN in optical networks entails supporting additional capabilities when compared to its application on the traditional networks. In this regard, the control plane for an optical network does not only configure the switching element but also enables other automatic control functionalities such as strategies required when heterogeneous technological networks are considered as well as the programmability of optical transmit devices. The SDN control framework fosters the implementation of user-defined and application specific complex data collection, monitoring techniques and the employment of emerging data analytics and ML techniques. In this context, an SDN controller can dynamically not only programming the data plane but also monitoring key elements to continuously fulfil the service requirements, ensuring key performance indicators (KPIs) and maintaining the required quality of service and/or adequate signal quality. To support the collection of a huge amount of information, a scalable and integrated database (DB) is used to store both the network state information (e.g., available resources, connectivity, existing connections, etc.) along with monitoring information from the data plane devices. The latter can be attained through optical performance monitoring techniques. Next, ML applications, running on top of the database retrieve monitoring data from the database, are used to predict/estimate the performance in terms of link or end-to-end connections. Thus, the SDN controller takes over of the updating such a database to push information about the network paths and then query the database-stored monitored information to conduct the ML based estimations.

In light of the above, it is well-accepted that ML will play a vital role in the efficient control and management of SDN networks, including optical. Specifically, it will enable the automation of networking functionalities through the softwarization of the network planning and operations that are becoming increasingly complex in an ever changing, heterogeneous, and uncertain environment. Several ML applications have been already developed and explored for optical network planning purposes [59]. QoT estimation is one such use case, that relies on the fundamental of ML and a frequent interaction between the controller, monitoring data, path computation element, etc. to improve its estimation accuracy [59]-[61]. In general, for most of the ML assisted use cases such as QoT estimation, fault and failure localization, etc., the state-of-the-art assumes that the optical network is centrally controlled by an SDN controller [16] equipped with storage, processing, and monitoring capabilities.

As SDN controlled optical networks evolve to support highly flexible and dynamic connectivity services, they also need to offer more reliable connectivity along with increased network resource efficiency. To achieve this, human interventions based on network measurement and management is one of the major limiting factors. This limitation becomes more prominent when scaling in the network size. In currently SDN-controlled optical networks, one of the main benefits is to exploit the closed control loop (CCL) operations such as *observe*, *decide/analyze* and *act* as shown in **Figure. 2.7**. In

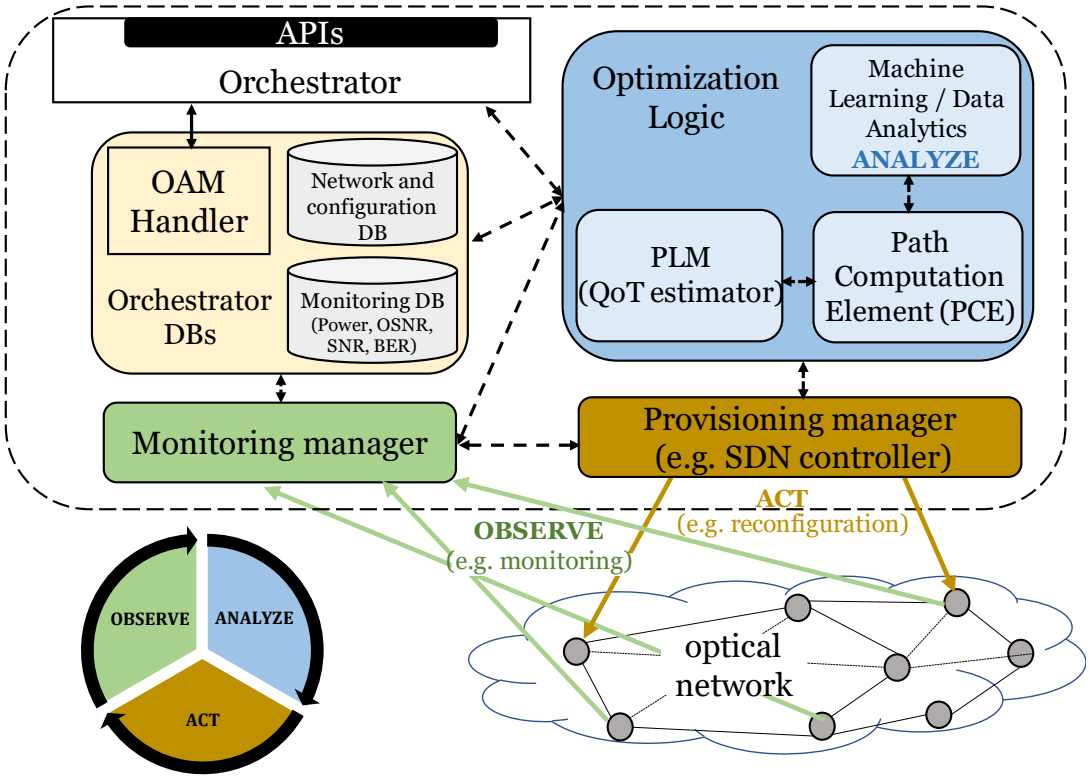


Figure. 2.7. Interaction between different entities of SDN based optical network forming automated closed control loop.

general, CCLs implemented within the SDN framework include several functional blocks related to monitoring, network optimization and control/management functionalities. **Figure. 2.7** shows a candidate architectural design of CCL that monitors (observe) the network state, undergoes data processing (analyze) to infer meaningful information from the network states, and finally recommends the SDN controller to trigger (act) some actions either accordingly or for different purposes such as increasing the network efficiency. In the context to this thesis, the action is to leverage the monitoring information, to train the QoT estimation tool to improve the network efficiency with uses cases such as incremental planning and transponders launch power optimization. Note that the proposed Optimization Logic block can be considered either as a module of the SDN controller, or as an external entity/application that interfaces with the SDN controller through the NBI. Herein, we specifically focus on the design where the Optimization Logic block is a module that resides within the SDN controller framework.

- Databases (DBs)

They include both the Traffic Engineering Database (TED) and the Network Configuration Database (N-DB) where the traffic engineering and network information such as the range of spectrum occupied per link, traversed interfaces, bitrate, occupied frequency slot, etc. is stored [60], [61]. Besides these databases, we further consider the Physical Layer Monitoring Database (M-DB). This DB stores all the accessible information related to the physical layer, which is known from the devices' data sheets or accessible through monitoring. M-DB may include OSNR, BER of active connections, amplifiers' information (e.g., noise figure (NF)), fiber information (e.g., attenuation, CD,

average differential group delay, and effective area), and so on. They also store information about the relationship between network elements and connections: e.g., a given link is composed by a specific set of fibers and amplifiers, a connection traversing a given set of links and nodes is using a specific transponder. Such information is relevant for the algorithms performing the targeted QoT estimation. We will learn about these algorithms and schemes in the upcoming Chapters 4 to Chapter 6.

- Orchestrator

It coordinates the operations of all the functionalities and operations within the controller. Furthermore, it allows an API to receive incoming requests from upper-layer applications. In a nutshell, the orchestrator module takes over of the required workflows and interactions (e.g., to handle the connection lifecycle) among the different components such as the Operation and Administration Maintenance (OAM) Handler, the Path Computation Element (PCE) and the Provisioning Manager [60].

- Path Computation Element (PCE)

This module is devoted to the path computation across a network graph. This module performs the wavelength or spectrum assignment in circuit-switched optical networks. The PCE receives from the orchestrator element requests for path computation. To do so, the PCE operates with a network topology view stored in the TED and considers the requirements arriving to the controller through the NBIs. Such requirements specify the source, destination, and the bandwidth demand. The PCE interacts with the PLM, more specifically the Qtool, to compute the quality of the requested connections at the considered paths. Moreover, with the current ML driven SDN architecture, an additional ML block is also present as shown in **Figure. 2.7**. This ML block accesses the DBs and train the ML models. In our work, this ML module deals with a trained ML model to enhance the estimation accuracy of the standard Qtool.

- Monitoring Manager

This module is devoted to collecting information from the different monitoring points deployed over the network. The information is aggregated and eventually used to detect faults and/or degradations which may trigger the corresponding actions (e.g., restoring, re-allocation, etc.). In this context, the OAM Handler is the responsible for processing such monitored information, correlating it, and triggering via other system components the required actions to either preserve or recover the affected services.

- Provisioning Manager

It is the function that handles the interactions between the SDN controller and the local agents governing each network element (i.e., optical switch, transceiver, etc.). Thus, the provisioning manager relying on a defined SBI, enables the actual programmability of the underlying transport infrastructure to provision new lightpaths and/or dynamic recon-figuring existing ones upon the decisions made by the PCE.

2.7 Introduction to Machine Learning

In this section, we give a broad categorization of the ML algorithms based on the learning procedure. Then we introduce some basic regression algorithms including the support vector machine (SVM).

In general, based on the learning process, the set of ML algorithms can be broadly partitioned in supervised, unsupervised, and reinforcement learning algorithms. The application of each type of algorithm to be used mostly depends on the availability of labeled data and on the actual problem goals. In ML, while using these algorithms, the first issue to be addressed is to understand which type of data can be provided to the algorithm and the amount of information that can be used to teach the algorithm to solve the problem. In optical networks, we observe that different problems can be approached with different types of ML algorithms [59]. The complexity of optical network makes this choice broad: huge information starting from physical to application layer is generated at different monitors such as OPMs, sensors etc. The choice of which information to be used is fundamental and need to consider a multitude of factors, including the easiness to retrieve the data, the amount of this data, the cost etc.

2.7.1 Supervised Learning

In supervised learning, the algorithms learn using the labeled dataset. In this type of learning algorithm, the objective is to create a statistical model for predicting, or estimating, an output based on one or more inputs. Generally, supervised learning involves observing several examples (instances) of a random input *feature* vector $\mathbf{x}$ and an associated value or *target* vector $\mathbf{y}$ to learn the mapping function f from $\mathbf{x}$ to $\mathbf{y}$, usually by estimating $p(\mathbf{y}|\mathbf{x})$. The term *supervised* learning originates from the view of the target $\mathbf{y}$ being provided by an instructor who shows the ML system what to do. In other words, in supervised learning one can consider that the learning is guided by a *supervisor*. We have a dataset which acts as a supervisor and its role is to train the model or the machine. In general, *regression* and *classification* are categorized under the same umbrella of supervised ML algorithms. Both share the same concept of utilizing known datasets (referred to as training datasets) to make predictions. However, the main difference between them is that the target variable in regression is numerical (or continuous) while that for classification is categorical (or discrete). In ML, regression algorithms attempt to estimate the mapping function f from the input variables $\mathbf{x}$ to numerical or continuous output variables $\mathbf{y}$. On the other hand, classification algorithms attempt to estimate the mapping function f from the input variables $\mathbf{x}$ to discrete or categorical output variables $\mathbf{y}$.

For example, based on the collection of monitoring data from established set of connections in an optical network, we want to implement a real-time QoT estimator that can predict the exact (continuous) value of the quality metric such as the SNR. Similarly, for classification the goal is to identify which new connection requests are either feasible or not before its actual deployment (i.e. category). If we succeed in collecting a sufficient number of samples, we can train in a supervised way, a ML algorithm where each session is labeled according to the specific task (i.e. regression or classification).

2.7.2 Unsupervised and Semi-supervised Learning

With unsupervised statistical learning, the goal is to learn structure from the input data. In this ML approach, the algorithm learns through observation and try to find structures in the data. The algorithm receives unlabeled input dataset (only **x**) with the objective to find a pattern and learn relationships from it. In this case, we let the algorithm learn by itself, without providing the target variable (**y**) to the problem we want to solve. Conversely to the supervised learning techniques, the unsupervised algorithms involve observing several examples of a random vector **x**, and attempting to either implicitly or explicitly learn the probability distribution $p(\mathbf{x})$, or some interesting pattern or properties of that distribution. In the unsupervised learning, there is no as such *supervisor* and the algorithm learn to find pattern in the unlabeled data without any supervision, hence called as "unsupervised" learning algorithms. The most widely used algorithms under unsupervised ML are clustering algorithms that group data together based on their similarities or patterns.

In Semi-supervised Learning, the learning process is a mix of supervised and unsupervised. In theory, there is no strict definition for this type of algorithms. However, the learning includes both supervised and unsupervised tasks. For example, within this category, the algorithms make use of partially labeled datasets. Typically, in practical scenarios, only a small number of samples are labeled, and the objective is to assign a label to the unlabeled portion of data, which is normally larger. An alternative approach is to train a ML model on the labeled portion of your data set, then using the same model to generate labels for the unlabeled portion of your data set. Finally, the complete dataset can be then used to train a new model. For example, one way to do semi-supervised learning is to combine clustering and classification algorithms.

2.7.3 Reinforcement Learning

Reinforcement Learning (RL) is defined as a set of algorithms to solve sequential decision processes where one agent directly interacts with the environment, which returns a reward through which the agent learns the optimal policy. The definition of a RL problem involves an environment that can be defined by a set of actions A, a set of states S, and a reward function R. The formulation of the problem is derived from Markov-Decision Processes (MDP), and mathematically the solutions are equivalent for both formulations. In the context of optical networks, we can model a problem with RL, when the network agents take actions to manage the network procedures, protocols or functions. In simple terms, RL determines the ability of an agent to interact with the environment and find out what is the best outcome. It follows the concept of "hit and trial" method. The agent is rewarded or penalized according to a correct or a wrong prediction. By doing so over large training examples and relying on the positive rewards the model trains itself. Once the system is trained, it becomes ready to predict a new incoming input data presented to it. Although RL is not a novel paradigm within ML, its application to optical networks has not yet been fully investigated. Mainly, RL is used in network self-configuration, more specifically for resource allocation strategies and network optimization tasks.

2.8 Supervised Machine Learning Algorithms

In this section, we provide the mathematical description of the algorithms used later in this Ph.D. thesis.

2.8.1 Linear Regression

Linear Regression is a supervised ML algorithm where the predicted output is continuous and has a constant slope. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y (output) as shown in **Figure. 2.8**. It is used to predict values within a continuous range rather than trying to classify them into categories. Simple linear regression uses traditional slope-intercept form, where m and b are the variables that algorithm will try to learn to produce the most accurate predictions given by

$$\mathrm{y} = m\,\mathrm{x} + b \tag{2.14}$$

There is also multivariable linear ML regression model that are used to estimate the relationship between two or more independent variables and one dependent variable. A more complex, multi-variable linear equation with n independent variables $(\mathrm{x}_1, \dots, \mathrm{x}_\mathrm{n})$ and one target variable (y) with $n + 1$ or $(m_1, \dots, m_n, b)$ coefficients can be represented by

$$\mathrm{y} = m_1\,\mathrm{x}_1 + \dots\dots + m_n\,\mathrm{x}_\mathrm{n} + b \tag{2.15}$$

$x_1, \dots, x_n$ represents the attributes, or distinct features for each training sample within each observation. By achieving the best-fit regression line, the model aims to predict target value (y') such that the error difference between predicted value and true value is minimum. So, it is very important to update the $(m_1, \dots, m_n, b)$ values, to reach the best value that minimize the error between predicted and true y values. Several functions can be used to achieve this objective and these functions are known as *cost functions*. However, mean square error (MSE) is the most widely used cost function that for simple (one variable) linear regression it is given by

$$minimize\ MSE = \frac{1}{N}\sum_{i=1}^{N}(\,y_i - \,y_i')^2 \quad = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2 \tag{2.16}$$

where N denotes the total number of observations (data points).

To update $(m_1, \dots, m_n, b)$ values to reduce the cost function (minimizing MSE value) and achieving the best fit line, it is common to use solvers like gradient descent. The basic idea behind the solver is to start with random $(m_1, \dots, m_n, b)$ values and then iteratively update them to reach the minimum cost. Gradient descent consists of looking at the cost function value with the current weights/coefficients, using the derivative of the cost function to find the gradients. Gradients can be considered as the slope of the cost function using current weight and then changing the weight to move in the direction opposite of the gradient. We need to move in the opposite direction of the gradient since the gradient points up (maximize) the slope instead of down (minimize), so we move in the opposite direction to decrease or minimize our cost function.

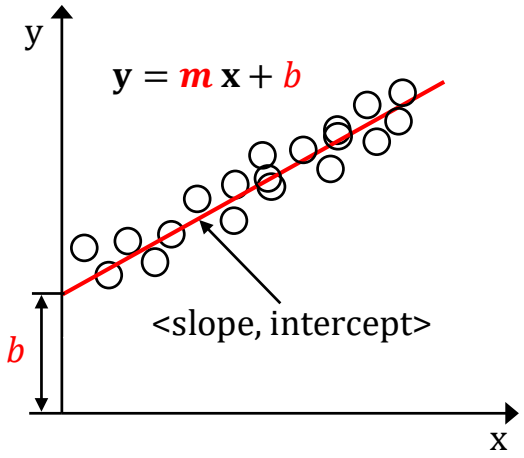


Figure. 2.8. An example of linear regression with one independent one target variable.

In this thesis, we leverage Linear regression of polynomial complexity to estimate the EDFA gain ripple penalties. Such an implementation is detailed within the formulation and proposed as a solution part in Chapter 4.

2.8.2 Support Vector Machine

The Support Vector Machines (SVM) method is a ML algorithm proposed by Vapnik et al. based on statistical learning theory in the early 1990s [62]. Generally, SVM is a classification approach, but it can be also exploited in regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes.

- **SVM as Classifier**

Unlike any other classifiers, the SVM explicitly tries to find the best separating line or decision boundary to separate the data points depending upon their class. In case of dataset space with more than two dimensions, the decision boundary become surface or plane, referred as hyperplane. It is very clear from the **Figure. 2.9(a)**, that there are infinite possibilities of hyperplanes that can be identified while classifying the data points. The SVM algorithm searches for the closest points from both classes, called as the *support vectors*. The name SVM is due to the fact that points are like vectors and that the best line depends on or is supported by the closest points as shown in **Figure. 2.9(b)**. In SVM, the so-called support vectors chosen within the training set plays a crucial role, since the hyperplane choice is only influenced by their position in the problem space.

Moreover, it is a well-known fact that the higher dimensional transformations allow separating data to achieve better classification predictions. This higher-dimensional space is also known as feature space. This transformation is generally carried out to classify data which eventually is nonlinearly separable, which is the cost for most of the real-world problems. In real applications, there might be many features in the data and applying transformations $\varphi(\cdot)$. This involve many polynomial combinations (during higher dimension transformation) of these features that results in extremely high and

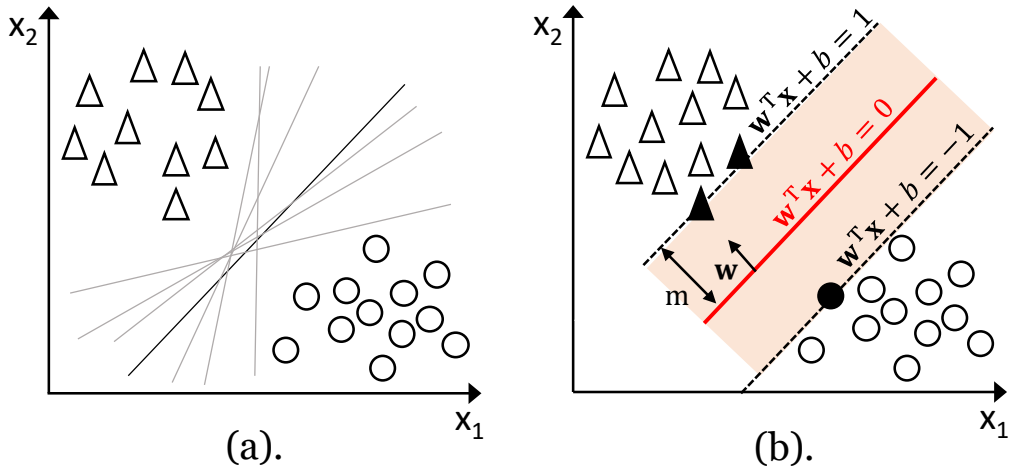


Figure. 2.9. (a) Hyperplane identification: many options are available for dividing the two classes of points, and (b) Optimal hyperplane (red line) separating two data classes (circles and triangles). The solid circles represent the borderline points of each class, also known as support vectors.

impractical computational costs. The kernel trick provides a solution to this problem. The *trick* is that kernel methods represent the data only through a set of pairwise similarity comparisons between the original data observations $\mathbf{x}$ (with the original coordinates in the lower dimensional space), instead of explicitly applying the transformations and representing the data by these transformed coordinates in the higher dimensional feature space. Indeed, leveraging a kernel function, the algorithm can avoid to explicitly transform the data, which would be extremely computationally expensive, but can still exploit the benefits that would derive from the transformation. Polynomial (homogeneous and inhomogeneous), hyperbolic and the Gaussian Radial Basis Function (RBF), hyperbolic tangent are some of the commonly employed kernel functions in the SVM algorithm.

In the following paragraphs we present the methodology to train the SVM model, translating the basic concepts introduced above into mathematical formulas.

SVMs are much harder to interpret in higher dimensions. It is much harder to visualize how the data can be linearly separable, and what the decision boundary would look like. A hyperplane in d -dimensions is a d-1 dimensional flat subspace that lies inside the larger d -dimensional space. In 2 dimensions, the hyperplane is just a line. In 3 dimensions, the hyperplane is a regular 2-d plane. Following the notation of Figure. 2.9., a generic hyperplane of the feature space can be defined as

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0 \tag{2.17}$$

where $\mathbf{x}$ is the vector representing the input samples of the SVM, such that $\mathbf{x} \in \mathbb{R}^n$ with n representing the dimension of the vector ($n = 2$ represents two features in dataset $\mathbf{x}$); $\mathbf{w}$ is the weight vector representing the polynomial coefficients, $\mathbf{w}^{\mathrm{T}}$ is the transpose of $\mathbf{w}$, and b represents the so-called bias (in 2-d plane it is the intercept of the line) parameter. $\mathbf{w}$ and b are the SVM parameters to be learnt during the training phase. The SVM goal is

to find a hyperplane that maximizes the margin m between the hyperplane itself and the support vectors. Such a margin can be expressed as

$$m = \frac{1}{\|\mathbf{w}\|} \tag{2.18}$$

where $\|\mathbf{w}\| = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ represents the Euclidean norm of the weight vector $\mathbf{w}$.

Therefore, the search of the optimal hyperplane reduces to the finding of its corresponding optimal values of $\mathbf{w}$ and b for which the distance m is maximized, under the constraint that all the input data points are classified correctly. Such conditions can be mathematically derived as follows. Assuming to classify the samples belonging to the first class with 1 and those belonging to the second class with -1, one can write

$$\mathrm{y}_i = \begin{cases} -1, & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \leq -1 \\ 1, & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \geq 1 \end{cases} \tag{2.19}$$

where y_i represents the class assigned by the SVM algorithm to the i-th input $\mathbf{x}_i$. Then, the condition of making only correct decisions for all the input data points can be written as

$$\mathrm{y}_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right) \geq 1, \qquad i = 1,2,\dots,N \tag{2.20}$$

where N represents the total length of the input data.

Thus, the optimization problem related to the optimal hyperplane research becomes

$$\begin{aligned} & min_{\mathbf{w},b} \ \|\mathbf{w}\| \\ \text{subject to} \quad & \mathrm{y}_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right) \geq 1, \qquad i = 1,2,\dots,N \end{aligned} \tag{2.21}$$

Eq. (2.21) represents a quadratic optimization problem also known as hard-margin SVM. As previously mentioned, in case of nonlinearly separable data a transformation to a higher dimensional feature space needs to be considered to speed up the computation time. For such a purpose, a mapping function $\varphi(\cdot)$ needs to be used and the optimization problem described in Eq. (2.21) becomes

$$\begin{aligned} & min_{\mathbf{w},b} \quad \|\mathbf{w}\| + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & \mathrm{y}_i\left(\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \qquad i = 1,2,\dots,N \\ & \xi_i \geq 0, \qquad\qquad i = 1,2,\dots,N \end{aligned} \tag{2.22}$$

where C, also known as penalty of the error term, is a hyperparameter representing the cost assigned to any misclassification (i.e., higher C will imply a more strict data separation). The quadratic optimization problem described by Eq. (2.22) is also known as soft-margin SVM.

As previously mentioned, often the data to be classified are nonlinear separable. Therefore, a shift towards a higher dimensional features space is required. To avoid computing the mapping function $\varphi(\cdot)$, the so-called kernel trick can be leveraged. To do

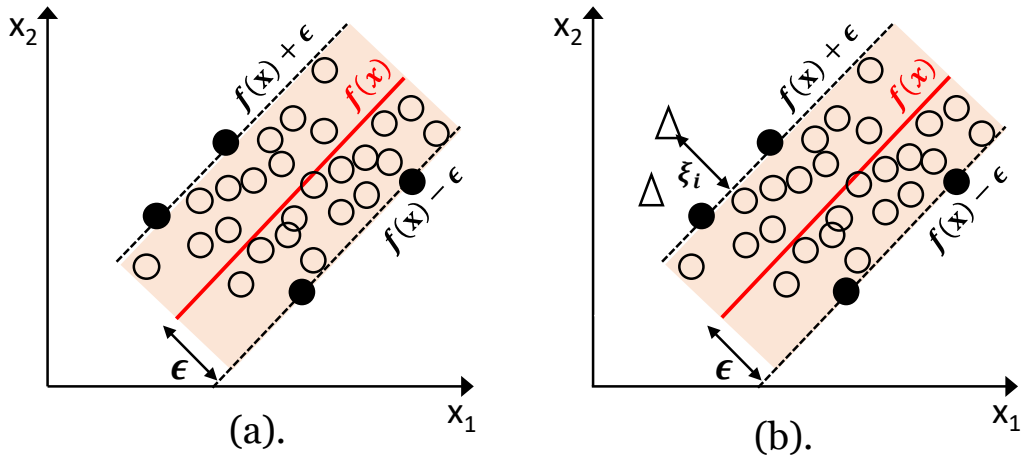


Figure. 2.10. (a) Support Vector Machines for regression. Each circle represents an input point, the solid circles represent the support vectors, the red line represents the fitting function $f(x)$, and the dashed black lines represent the error margins defined by ϵ, and (b) The triangle circles represent the input points falling outside the error margins ϵ, which are taken into account by slack variables ξ

so, the primal problem described by Eq. (2.22) needs to be transformed in a dual problem by means of the Lagrange dual formulation. This will allow the chosen nonlinear kernel function to be employed.

The most common solutions available to solve the optimization problem described by Eq. (2.22) are the CVXOPT Python package for convex optimization and the Sequential Minimal Optimization approach. For a given input $\mathbf{x}_i$, the output of the optimization problem will be a set of weights $\mathbf{w}$, whose linear combination will correspond to the predicted class y_i. The analysis presented above is valid for binary classification problems only. To address multiclass classification problems, they simply need to be reduced in a series of binary ones or the so called *one-vs-all* approach [63].

- **SVM as Regressor**

The SVM method used for solving a regression problem is generally called as SVM regressor or SVMR. Similar to the SVM as classifier, SVMR is also reduced to an optimization problem. However, unlike the previous case, now the minimization of $\mathbf{w}$ is subject to the condition that the label y_i (which now assumes continuous values) assigned to the i-th input x_i deviates from the hyperplane within the accuracy boundary ϵ, for all the samples i, as shown in **Figure. 2.10(a)**.

In other words, if the prediction errors fall within the interval defined by ϵ, the prediction is considered accurate. Therefore, by tuning ϵ, the exact accuracy level of the model can be chosen. Consequently, while in SVM for classification the hyperplane represented the optimal decision boundary, in SVMR it represents the objective fitting function $f(x)$ to be identified. Moreover to minimize any possible deviation larger than ϵ, slack variables can be also considered for those points lying outside the accuracy boundaries, as shown in **Figure. 2.10(b)**. Bearing in mind such assumptions, the optimization problem described by Eq. (2.22) can be restated as

$$min_{\mathbf{w},b} \quad \|\mathbf{w}\| + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad |\mathrm{y}_i - f(\mathrm{x})| \leq \epsilon + \xi_i, \qquad i = 1,2,\dots,N \tag{2.23}$$

$$\xi_i \geq 0, \qquad i = 1,2,\dots,N$$

where ξ_i represents the slack variable associated to the i-th input point which falls outside the error margin defined by the hyperparameter ϵ, C is the hyperparameter representing the tolerance for such points (i.e., an higher C will imply a higher tolerance for points lying outside the error margin), and $f(\mathrm{x}) = (\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i) + b)$ represents the line (or the hyperplane) fitting the input data. Again, similar to SVM as classifier fundamentals, while dealing with nonlinear data similar transformation $\varphi(\cdot)$ into a higher dimensional space should be considered. To do so, the similar kernel functions presented in the previous section can be employed.

In this thesis, we leverage SVMR to estimate the EDFA gain ripple penalties and filtering penalties inside the ROADM nodes. Such an implementation is detailed within formulation as well as proposed as a solution part in the Chapter 4.

2.9 Nonlinear Least Square Curve Fitting

Generally, curve fitting and ML regression involves approximating the data with functions. Moreover, various algorithms of ML could be applied to curve fitting. However, in most cases these do not have the efficiency and accuracy of more general curve fitting algorithms, finding a choice of parameters for a mathematical model which gives best fit (defined in various ways) to a data set. ML in a sense is more generic, it includes selection of features, models, splitting of the sets of training, cross-validation, and testing etc. However, if the model can be fitted with curve fitting tools and techniques, certain ML algorithms can also be directly used to fit specific parameters of that model. Generally, nonlinear least square curve fitting is very useful when the goal is to optimise the parameters of some nonlinear model based on some measurements. For example, these fitting techniques can be used to train QoT estimation tool or more specifically, PLM parameters with the monitoring information (this approach is presented in as we Chapter 3).

In fitting a model function $f'(\mathrm{x};\boldsymbol{r})$ of an independent variable x and a vector of n parameters $\boldsymbol{r}$ to a set of m data points (x_i, f_i), it is convenient to *minimize* the sum of the weighted squares of the errors between the data f_i and the curve-fit function $f'(\mathrm{x};\boldsymbol{r})$.

$$e^2(\boldsymbol{r}) = \sum_{i=1}^{m}\left[\frac{f(\mathrm{x}_i) - f'(\mathrm{x};\boldsymbol{r})}{\sigma_{f_i}}\right]^2 \tag{2.24}$$

$$e^2(\boldsymbol{r}) = (\boldsymbol{f} - \boldsymbol{f}'(\boldsymbol{r}))^T\,\boldsymbol{W}\,(\boldsymbol{f} - \boldsymbol{f}'(\boldsymbol{r})) \tag{2.25}$$

$$e^2(\boldsymbol{r}) = \boldsymbol{f}^T\boldsymbol{W}\boldsymbol{f} - 2\boldsymbol{f}^T\boldsymbol{W}\boldsymbol{f}' + \boldsymbol{f}'^T\boldsymbol{W}\boldsymbol{f}' \tag{2.26}$$

where σ_{f_i} is the measurement (monitoring) error for data $f(\mathrm{x}_i)$. Typically, the weight matrix $\boldsymbol{W}$ is diagonal with $W_{ii} = 1/{\sigma_{f_i}}^2$. In general, the weights W_{ii}, can be set to pursue

other curve fitting goals. This scalar-valued goodness-of-fit measure is called the *chi-squared error criterion* because the sum of squares of normally-distributed variables is distributed as the chi-squared distribution.

If the function $f'(x; \boldsymbol{r})$ is nonlinear in the model parameters $\boldsymbol{r}$, then the minimization of e^2 with respect to the parameters needs to be carried out iteratively. The goal of each iteration is to find a perturbation $\boldsymbol{h}$ to the parameters $\boldsymbol{r}$ that reduces e^2.

In the upcoming Chapter 5 and Chapter 6, we extensively rely on Levenberg-Marquardt (LM) algorithm for retraining of Qtool parameters. The LM algorithm was developed in the early 1960's to solve nonlinear least squares curve-fitting problems [64]. Generally, the least square problems arise in the context to fitting a parameterized mathematical model to a set of data points by minimizing an objective function expressed as the sum of the squares of the errors between the model function and a set of data points. If a model is linear in its parameters, the least squares objective is quadratic in its parameters. In this case, this objective may be minimized with respect to the parameters in one step via the solution to a linear matrix equation. However, most of the problems comes with a nonlinear fitting function. So, if the fit function is nonlinear in its parameters, the least squares problem requires an iterative solution algorithm. Such algorithms reduce the sum of the squares of the errors between the model function and the data points through a sequence of well-chosen updates to values of the model parameters.

LM algorithm is a combination of two numerical minimization algorithms: the gradient descent method and the Gauss-Newton method. In case of gradient descent method, the sum of the squared errors is reduced by updating the parameters in the steepest-descent direction. In the Gauss-Newton method, the sum of the squared errors is reduced by assuming that the least squares function is locally quadratic in the parameters, and then the goal is to find the minimum of this quadratic. The LM method acts more like a gradient-descent method when the parameters are far away from their optimal value and acts more like the Gauss-Newton method when the parameters are quite close to their optimal value.

We now briefly introduce the gradient descent and the Gauss-Netwon method to mathematically describe the Levenberg-Marquardt Method as follows.

2.9.1 The Gradient Descent Method

The steepest descent method is a general minimization method which updates parameter values in the *steepest* or *downhill* direction. This is the direction opposite to the gradient of the objective function. The gradient descent method converges well for problems with simple objective functions [65], [66]. The gradient of the chi-squared objective function with respect to the parameters is

$$\frac{\partial}{\partial \boldsymbol{r}} e^2 = 2\,(\boldsymbol{f} - \boldsymbol{f}'(\boldsymbol{r}))^T\, \boldsymbol{W}\, \frac{\partial}{\partial \boldsymbol{r}} (\boldsymbol{f} - \boldsymbol{f}'(\boldsymbol{r})) \tag{2.27}$$

$$\frac{\partial}{\partial \boldsymbol{r}} e^2 = -2(\boldsymbol{f} - \boldsymbol{f}'(\boldsymbol{r}))^T\, \boldsymbol{W} \left[\frac{\partial \boldsymbol{f}'(\boldsymbol{r})}{\partial \boldsymbol{r}}\right] \tag{2.28}$$

$$\frac{\partial}{\partial \boldsymbol{r}} e^2 = -2(\boldsymbol{f} - \boldsymbol{f}')^T \boldsymbol{W} \boldsymbol{J} \tag{2.29}$$

where $\boldsymbol{J}$ is the m x n Jacobian matrix which represents the local sensitivity of the function f' to the variation in the parameter $\boldsymbol{r}$. The parameter update $\boldsymbol{r}$ that moves the parameters in the direction of the steepest descent is given by

$$h_{GD} = \alpha \boldsymbol{J}^T \boldsymbol{W} (\boldsymbol{f} - \boldsymbol{f}') \tag{2.30}$$

where α is the positive scalar that determines the length of the step in the steepest-descent direction.

2.9.2 The Gauss-Newton Method

The Gauss-Newton method presumes that the objective function is approximately quadratic in the parameters near the optimal solution [67]. For moderately-sized problems the Gauss-Newton method typically converges much faster than gradient-descent methods [68]. The function evaluated with perturbed model parameters can be locally approximated via a first-order Taylor series expansion as given by

$$\boldsymbol{f}'(\boldsymbol{r} + \boldsymbol{h}) \approx \boldsymbol{f}'(\boldsymbol{r}) + \left[\frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{r}}\right] \boldsymbol{h} = \boldsymbol{f}' + \boldsymbol{J}\,\boldsymbol{h} \tag{2.31}$$

Substituting the approximation $\boldsymbol{f}'(\boldsymbol{r} + \boldsymbol{h}) \approx \boldsymbol{f}' + \boldsymbol{J}\,\boldsymbol{h}$ into Eq. (2.26) for $e^2(\boldsymbol{r} + \boldsymbol{h})$,

$$e^2(\boldsymbol{r} + \boldsymbol{h}) \approx \boldsymbol{f}^T \boldsymbol{W} \boldsymbol{f} + \boldsymbol{f}'^T \boldsymbol{W} \boldsymbol{f}' - 2\boldsymbol{f}^T \boldsymbol{W} \boldsymbol{f}' - 2(\boldsymbol{f} - \boldsymbol{f}')^T \boldsymbol{W} \boldsymbol{J} \boldsymbol{h} + \boldsymbol{h}^T \boldsymbol{J}^T \boldsymbol{W} \boldsymbol{J} \boldsymbol{h} \tag{2.32}$$

The first-order Taylor approximation results in an approximation for e^2 that is quadratic in perturbation $\boldsymbol{h}$. The Hessian of the chi-squared fit criterion is approximately $\boldsymbol{J}^T \boldsymbol{W} \boldsymbol{J}$.

The update for the parameter $\boldsymbol{h}$ that minimizes e^2 is found from $\partial e^2 / \partial\, \boldsymbol{h} = 0$:

$$\frac{\partial}{\partial \boldsymbol{r}} e^2(\boldsymbol{r} + \boldsymbol{h}) \approx -2(\boldsymbol{f} - \boldsymbol{f}')^T \boldsymbol{W} \boldsymbol{J} + 2\boldsymbol{h}^T \boldsymbol{J}^T \boldsymbol{W} \boldsymbol{J} \tag{2.33}$$

which results in the normal equations for the Gauss-Newton update are given by

$$[\,\boldsymbol{J}^T \boldsymbol{W} \boldsymbol{J}\,] h_{GN} = \boldsymbol{J}^T \boldsymbol{W} (\boldsymbol{f} - \boldsymbol{f}') \tag{2.34}$$

Note that the right-hand side vectors in normal equations for the gradient descent method (Eq. (2.30)) and the Gauss-Newton method (Eq. (2.34)) are identical.

2.9.3 The Levenberg-Marquardt Algorithm

As already discussed, in the LM algorithm, the parameter updates are adaptively varied between the gradient descent update and the Gauss-Newton update as given by

$$[\,\boldsymbol{J}^T \boldsymbol{W} \boldsymbol{J} + \boldsymbol{d}\, I\,] h_{LM} = \boldsymbol{J}^T \boldsymbol{W} (\boldsymbol{f} - \boldsymbol{f}') \tag{2.35}$$

where $\boldsymbol{d}$ is the *damping parameter* whose small values result in a Gauss-Newton update and large values result in a gradient descent update. In general, the damping parameter $\boldsymbol{d}$ is initialized to be large so that first updates are small steps in the steepest-descent direction. There are frequent checks after each iteration of parameter updates. If any iteration happens to result in a worse approximation $\left(e^2(\boldsymbol{r} + h_{LM}) > e^2(\boldsymbol{r})\right)$, then $\boldsymbol{d}$ is

increased. Otherwise, as the solution improves, $\boldsymbol{d}$ is decreased, the LM method approaches the Gauss-Newton method, and the solution typically accelerates to the local minimum.

Note that in LM's update relationship [68], the values of the damping parameter $\boldsymbol{d}$ are normalized to the values of $\boldsymbol{J}^T\boldsymbol{W}\boldsymbol{J}$ as given by

$$[\boldsymbol{J}^T\boldsymbol{W}\boldsymbol{J} + \boldsymbol{d}\, diag(\boldsymbol{J}^T\boldsymbol{W}\boldsymbol{J})]h_{LM} = \boldsymbol{J}^T\boldsymbol{W}(\boldsymbol{f} - \boldsymbol{f}') \tag{2.36}$$

Note that the LM algorithm might not find optimum for all nonlinear fitting problems (some of the problems have a single global and no local (e.g., convex), some have several locals). Depending upon the nature of the problem, the LM algorithm might get stuck. In order to overcome this, we should rerun the simulations with different initial/starting solutions (points). However, the problems tackled in the Chapter 5 and Chapter 6 of this thesis are already well proven to be convex and LM algorithm seems a suitable choice. In this thesis, we also leverage LM algorithm to train the parametric Qtool with the real world or monitoring information. Such an implementation is described within the formulation and proposed as a solution part in Chapters 5 and Chapter 6.

2.10 Conclusions

This chapter has been devoted to introducing and highlighting the required background to facilitate the understanding of the fundamental concepts tackled in this Ph.D. thesis. We started by introducing optical networks and the required components to realize it. Then, we discussed the trends in term of physical layer disaggregation. We also discussed the main optical impairments present in these networks along with the available modeling schemes that are used for the upcoming Chapters. We then briefly introduced the concept of QoT estimation with the traditional approach (LOGO) to adjust the launch power. We moved forward to present the role of design margin to cover the modeling inaccuracies of the QoT estimation tool. Finally, we introduced the required background to be integrated within an SDN architecture. We then provided background on several ML-based techniques and nonlinear least square curve fitting techniques that are used in this thesis.

The following chapter focuses on reviewing the state-of-the-art of the objectives of this Ph.D. thesis, aiming at identifying research opportunities.

Chapter 3

Review of the State-of-the-Art

In this chapter, we review the State-of-the-art related to the different objectives defined in Chapter 1 for this Ph.D. thesis. In general, the first part of this chapter is devoted to the detailed literature review on available Quality of Transmission (QoT) estimation techniques (**O1**). This Chapter also outlines the limitations of the adopted schemes in the literature. The second part of the literature review is based on the limitations when traditional single vendor PLM scheme or QoT estimation tool to multivendor partial disaggregated optical networks (**O2**). The last part of this chapter addresses the available literature on transponders (TPs) launch power optimization strategies (**O3**).

3.1 O1 - Quality of Transmission (QoT) Estimation Techniques

In this section, we summarize relevant previous research works investigating the QoT estimation problem within transport networks by means of analytical PLM and/or ML based techniques. We limit this literature overview to networks with coherent transmission, since non-coherent transmission is no longer deployed in the current transport network generation. In addition, the problems of both OPM and QoT estimation are tightly related, as monitoring is needed to feed most of the current QoT estimation frameworks available in the literature. For this reason, many ML-assisted QoT techniques based on the monitoring information are also considered in this review.

Data collection is one of the most important requirements for most of the works on ML-based QoT estimation schemes. Data collection can be difficult, time-consuming, and/or expensive. Since ML relies on the training phase, which is dependent on the amount and type of collected data, most of the studies in the ML-based QoT estimation topic are applied into *brownfield* networks. These are networks already deployed and operational. This eases the monitoring data to be collected directly from the field to train models. The most common scenario is the capacity upgrade (i.e., upgrade phase), where the network operator needs to assess that the QoT of a new service is acceptable, or determine the maximum capacity the service can carry based on its estimated QoT. Some of the techniques presented, however, also apply to networks that need to be yet physically deployed and for which field data is unavailable (i.e., planning phase)— the *greenfield* scenario. The estimation of the QoT of all the services to be established right after the network is deployed is a typical greenfield scenario. Note that resource computations such as routing and spectrum assignment (RSA), modulation format, launch power, span lengths, and so on are generally available (or can be computed) in both scenarios and are always considered to be a part of the Qtool.

In the literature three possible approaches exist to estimate QoT of connections. Thereby, we first briefly introduce these three approaches. Note that all the previous works discussed in this section are based on one of these three approaches of the QoT estimation tool or Qtool.

3.1.1 Possible Approaches for QoT Estimation

In this section, we introduce three possible approaches based on full analytical and/or ML based QoT estimation schemes. A detailed analysis of the available work on these schemes is done in the next section. However, in the following, we just briefly introduce them for the sake of both understanding and readability.

Consider a PLM (as discussed in Chapter 2) for the computation of SNR (quality metric), parameterized by a vector Θ_{PLM}. This parameter vector Θ_{PLM} could contain noise figure (NF) and gain of the EDFA, fiber parameters such as attenuation, dispersion coefficients etc. The goal of this approach is to estimate the quality metric denoted by $\mathbf{y}$ based on the input vector, $\mathbf{x}$ such as launch power, path/route of the signal, modulation format, baudrate etc. This approach is very similar to the scheme presented in Section 2.5 of last chapter (**Figure. 2.6**). However, an additional training input can be added to tune the parameter vector Θ_{PLM}. We call this approach as PLM-based QoT estimation scheme and is widely accepted in traditional optical network planning and upgrade phases. **Figure. 3.1(a)** shows the basic schematic for this approach.

In the next approach, the main idea is to replace the PLM (and its parameter vector Θ_{PLM}) with ML model. The ML model can be a parametric model (e.g. neural network, NN) with parameter vector Θ_{ML} equal to the weights and biases of the trained ML model. With proper training the resulting ML model can well approximate underlying physical layer of the optical network. This approach models the underlying system as a black box that has no prior knowledge of the underlying physical phenomenon. We call this approach as ML or black box based QoT estimation or Qtool scheme. **Figure. 3.1(b)** shows the basic schematic for this approach.

The third approach is a combination of the above two approaches. In this approach there is a PLM with parameter vector and the idea is to add ML based corrections to further tune the parameters of the PLM. This ML correction part could be a full black box model or mathematical equations based fitting schemes. This scheme is widely adopted in brownfield network planning and upgrade scenarios where monitoring information can be used to train a ML model. This trained ML model with parameter vector Θ can be then used to add corrections or to retune the parameters of the PLM. We call this approach as hybrid or ML-assisted QoT estimation scheme. **Figure. 3.1(c)** shows the basic schematic for this approach.

Note that all the above introduced three approaches have their pros and cons. More specifically, the first approach requires no data (or very low amount of data for initial training/adjust PLM parameters). Hence, this approach is cost-effective but misses the real network conditions and thus results in low accuracy [12]. The second approach requires a lot of data collection (via probing and monitoring) and hence cost-inefficient, yet highly accurate as the data comes from the real network [57]. Moreover, this approach could be feasible in brownfields as connections are already established. In

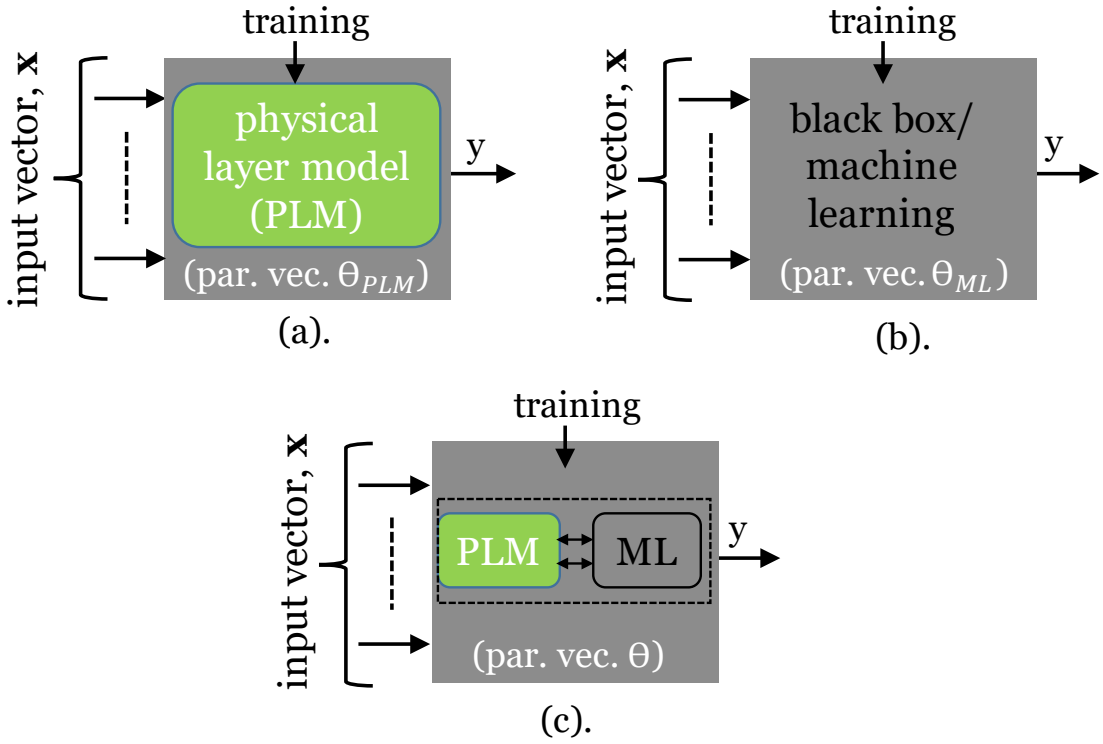


Figure. 3.1. Possible approaches for QoT estimation (a) PLM-based, (b) ML- based, and (c) hybrid or ML-assisted.

greenfield scenarios, this approach requires spare TPs (to probe and monitor) in order to generate data and can be highly expensive for greenfield planning.

The last scheme takes the benefit from both; the analytical PLM to generate data in a fast way and the monitoring information to train ML models to correct PLM parameters (or adding directly ML correction on top of PLM calculations). This scheme however needs synthetic data (from PLM) and real field monitoring data, whose amount is less than the data requirement of second approach [12]. However, this approach seems to be practically feasible in brownfields as connections are already established and monitoring their performance is possible. In terms of data requirement, fast calculations and cost-effectiveness, this scheme seems quite promising. Keeping this is mind, the work done in this thesis is based on the third approach as we will see in the Chapter 4 to Chapter 6.

Qtool has been investigated extensively in the last decades [57]. Typically, QoT estimation is performed using an analytical PLM [12] included in the Qtool. The main sources of noise accounted in Qtools are the ASE noise generated at both span and node amplifiers and the NLI noise, which considers fiber non linearities, mostly self- and cross-channel interferences (i.e., SCI and XCI). In the QoT framework, the optical impairment and signal distortion caused by passive network components are straightforward to calculate. However, fiber nonlinearities and active devices with proactive control logics such as erbium doped fiber amplifiers (EDFAs) and reconfigurable optical add-drop multiplexer (ROADMs) that have complex feedback processes, are difficult for modeling and calculation based on detailed physical models. If the data about every physical parameter throughout the system at every moment is

detailed, then we could just use these parameters with precise physics-based models. For the early point to point systems this could be done with reasonable accuracy. However, for mesh networks and more significantly in disaggregated networks, this approach is no longer practical. A common practice is to add a design margin to the Qtool to account for the modelling simplification assumptions and other uncertainty parameters [8], [9]. Removing such uncertainties would allow increasing the estimation accuracy and thus, reducing the margins. This leads to accomplish higher efficiency and/or lower cost during network planning and upgrading. A good engineering tool is to exploit ML, attain reasonable performance in predicting nonlinear and complex systems behavior with incomplete data. This is the reason that ML-based estimation techniques are gaining a lot of attention to improve Qtool accuracy [12], [16], [56]- [59],[69], [70].

In this thesis, the major objective is to improve current QoT estimation methods by using ML techniques on the monitoring information available from the established set of connections in an optical network. More specifically, we targeted ML-based modeling to estimate the penalties due to EDFA gain ripple effect (**O1.1**) and filtering uncertainties at the ROADM nodes (**O1.2**) for the Qtool. Hence in the following we limit our State-of-the-art literature survey particularly on the EDFA and ROADM filtering in context to the QoT estimation techniques.

3.1.2 O1.1 - QoT Estimation with EDFA Gain Characteristics

EDFAs are key devices in WDM transport networks that ensure the required connection QoT level at the receivers. However, EDFAs are the dominant noise source (ASE noise) in those networks. Typically, span EDFAs are operated in Automatic Gain Control (AGC) mode with near to zero tilt (first order/linear correction) to get a relatively flat gain in the C-band [71], [72]. Although the gain tilt is maintained at zero there are still fluctuations/ripples within the gain bandwidth of EDFAs. These gain ripples may be due to: *i)* residual imperfections in the gain flattening filters; and/or *ii)* wavelength dependent absorption/ emission coefficients of Er3+ ions [73], [74]. Also, typically, PLMs/Qtools assume that the EDFAs have constant NF across all WDM channels. Although NF mainly depends on the population inversion of erbium ions and follows McCumber theory [75], small NF variations across transmission band are expected.

The SNR is a function of the traversed spans and particularly depends upon the gain ripple profile of each span EDFA. The authors in [10] presented the effects of gain ripples on span EDFAs, leading to uncertain OSNR estimation. They also explored the effects of the variation in the operating conditions of span EDFAs (monitored by tapping the signal at each span EDFA) to minimize the OSNR penalty due to gain ripple. The authors in [76] proposed a pre-emphasis launch power technique to reduce the EDFA gain ripple effects. This technique is well suited for fully loaded links and requires precise control at the transmitter. The work in [76] use approximate analytical models to calculate noise at full channel loading conditions. However, [76] verified once more that EDFA gain ripple affects the quality of signal, hence SNR. A gradient descent-based launch power optimization strategy was used to improve the SNR in that work. However, all the span EDFAs gain ripple profiles were assumed to be identical, which, in general, is not realistic. Apart from these attempts, several analytical models were also proposed to estimate EDFA wavelength dependent gain and channel output power, under different

channel loading conditions. In general, all these models range from limited accuracy and simple characterization to very detailed estimations, trading-off processing time and accuracy. In [77] authors proposed an accurate ML model with cascaded EDFAs as a first step toward channel assignment/resource allocation. However, the benefits in terms of accuracy improvement or margin reduction for QoT estimation were not presented. The premises of all the above reviewed techniques are essentially the same: given the input power spectrum of an amplifier, to estimate the gain per channel or equivalently the output power spectrum using a neural network trained for instance in the lab., prior to deployment.

A most recent extension of EDFA gain spectrum modeling (in [78]) based on a hybrid (analytical and experimental dataset based neural network ML model) approach was presented in [79]. In this technique, a neural network is fed with the outputs of an EDFA analytic model in addition to the standard modeling inputs. Although the technique ultimately does not improve the modeling accuracy, it enhances the training time and reduces the amount of training samples. Some other recent ML-based works addressed the wavelength dependent gain spectra estimation [76], [80]. In [81], [82] neural network models were also proposed to recommend wavelength assignment with 99% of precision value that considers EDFA gain ripple effects. In [80] a single EDFA modeling required 18000 channel loading conditions. A more recent study in [83], additionally outputs the EDFA's settings to achieve a given response. As related work, the problem of finding EDFA's settings to achieve a target gain shape is put in the broader context of inverse system design [84],[85] whereby a ML framework is used to learn what parameters yield a desired output for some system, such as an optical amplifier. Solving the problem for a single setting implies that training may need to be done for each amplifier's settings. The cascade of EDFAs and its effect on gain ripples were also not considered. The dataset used for training those models were either *i)* analytical without considering fiber effects/NLI noise contribution, or *ii)* experimental which needs the data collection of each individual span EDFA. These requirements practically constrain the usability of those approaches. In [71], the gain uncertainties of EDFA working in AGC mode was experimentally investigated. A method was derived aiming at reducing these gain uncertainties up to 0.5 to 0.7 dB when encompassing up to five cascaded EDFA spans. The targeted scenario in [71] was dynamic with the add/drop of new/existing connections. Many recent works also rely on leveraging ML to estimate the EDFA gain profile or the signal QoT value with line modeling approach, i.e. point to point link [86]-[90]. In [89], [90] transfer learning techniques are further proposed to adjust a neural network trained on one line to another line with little amount of retraining, to estimate the OSNR [89] or Q-factor [90]. However, in terms of brownfield scenarios i.e. for fixed load EDFAs, with static network conditions, the gain ripples effect was not explored. Some works recently have been published on network wide simulations to estimate the effect of EDFA gain ripple on the signal quality to estimate the QoT of a service in the brownfield/upgrade. In this context, the approach used in [70], [91], [92] was to either learn the parameters of a physical model or to refine them when that those inputs are not accurately known. In particular, [92] focuses on learning EDFA gain ripple penalties from end-to-end measurements. The other works [70], [91] deal with end-to-end QoT estimation of the whole network based on full ML or black box approach. The problem

of handling parameter uncertainty is also tackled in [93] using the hybrid method, but without trying to refine the parameters.

3.1.3 O1.2 - QoT Estimation with Filtering Penalties inside ROADM nodes

ROADMs based on wavelength selective switches (WSSs) is another key element in WDM based optical transport networks. ROADMs support the colorless and/or directionless and/or contention less (CDC) multiplexing and demultiplexing at the network nodes [30], [31], [94], [95]. However, the WSS of the ROADMs are essentially filters that results in signal degradation. They narrow the signal bandwidth (BW), a problem which becomes severer as the number of WSSs/ROADMs are cascaded over long paths [40], [41]. Consequently, this causes optical filtering penalty that needs also to be modeled precisely and/or covered in the pre-allocated margins. Authors in [40] investigated the ROADM node penalty. Their studies found that filtering penalty increases exponentially as the number of traversed nodes increases, as well as that the penalty varies with respect to the modulation format and frequency grid spacing. In a deployed network, there are uncertainties in such penalties; slight variations in filters spectral responses are typical even for identical filters, while the transmitter laser and the filter central wavelengths could be misaligned. These uncertainties grow even higher for heterogeneous nodes with different types of filters (e.g., in multi-vendor scenarios [10], [96]). In light of the above, it is quite challenging to estimate the corresponding accumulated penalties.

The authors in [97] presented an analytical model based on a higher order SNR-OSNR relation to capture cascaded filtering effects but limited to point-to-point characterization. Moreover, the work presented in [40], [41] was also limited to characterization of cascaded effects and the network level perspectives were missing such as quality of individual filters in the cascaded chains, misalignment between filter and ITU-T grid etc. All these works focused on characterization of the cascade (either by looping single filter, or by replacing cascaded filters with a tunable BW filter) and not on the identification of the quality of the individual filters. Furthermore, almost none of these works included network level extensions. [98] proposed an approach for real time cascaded filtering penalty assessment (for different order Gaussian filters) and verified it for a commercial 200 Gb/s 16-quadrature amplitude modulation (16- QAM) transponder and WSS modules. [98] showed the effect of spectral shape variation in a cascade of filters which resulted in uncertain (as outliers in Figure. 4 of [98]) OSNR penalty estimation. However, spectral shape variations and the resulting uncertain OSNR penalty estimation were not taken into account in that study. As already discussed, this non-linear penalty is expected to exacerbate in disaggregated optical networks where ROADM/filters and transmitter (Tx.) lasers could come from multiple vendors with diverse characteristics. A recent study on disaggregation at node level indicated 3.5–5 dB (core) and 3–3.5 dB (metro) required margins [99], mainly due to uncertainties/variability of the multi-vendor components. Therefore, the filtering penalty of ROADM nodes play a significant part in those increased margins.

3.2 O2 - Multi-vendor Qtool Extensions

In the literature many traditional or non-ML based schemes for QoT estimation exist [10], [12], [70]. For such schemes, QoT estimation is performed using an analytical PLM

which is the key module of the QoT estimation tool. Partial disaggregation, where the line system is open (OLS) to multiple vendor transponders (TPs) has gained significant attention due to the ease in the control plane implementation. In this scenario, the GN PLM is a well-accepted PLM for traditional single vendor aggregated optical networks. However, several previous studies [10], [70], [100], [101] that used the GN model as the PLM for QoT estimation, resulted in a high design margin, compromising the Qtool accuracy. One of the main reasons for this is because these PLMs lack the information about TP vendor characteristics which are crucial for multi-vendor network scenarios. Even within TPs from a single vendor, statistical variations of TPs components exist [102]. For multi-vendor TPs in a common OLS, apart from the aforementioned TP statistical variances, vendor dependent factors play a significant role in the QoT/SNR estimations [99], [102], [103]. In such a network, performance variations arise from the different TP components and digital signal processing (DSP) used by the different vendors. So, if we use a typical single vendor PLM [10], [101] we might observe huge deviations in the estimated and measured QoT. Such estimation errors would affect (among other operations) the optimization of the launch power levels, the incremental planning etc. In Chapter 5, we will go through these use cases and show the impact of inaccurate PLM in the targeted partial disaggregated network scenarios.

3.3 O3 - Launch Power and Dynamic Optimization Techniques

Several works studied variations of power optimization [55], [76], [104]-[111] with the goal of minimizing intra- and inter-channel non-linear (Kerr) effects since these are the main limiting factor of the transmitted signals power in dispersion-uncompensated systems. Previous works relied on heuristics to optimize the individual [76], [104], [105], [109]-[111] channel launch power. Some works also proposed schemes to optimize the launch power for all the channel uniformly to improve the QoT of the transmitted signals [55], [106]-[108]. Authors in [104], [105] presented several approaches to optimize the launch power (along with the constellation and channel allocations) of each channel to maximize the network efficiency, taking into account NLI as well as ASE noise. Authors in [108] proposed an efficient launch power optimization method combining heterogeneous launch power control and digital nonlinear compensation to resolve the issue of fiber nonlinear impairments. However, this work was specifically focused on optimizing launch power as a function of inter/cross-channel noise with self-channel noise completely compensated in DSP. The local optimization leads to global optimization (LOGO) model [55] presented in Chapter 2, targets on maximizing SNR at the receiver by maximizing each span's SNR or minimizing each span's non-linear noise. The main benefit of this model is its flexibility to optimize the power of each span irrespectively of the span lengths (homo or heterogeneous). However, the LOGO assumes full load and the same power levels for all channels (at each span) and thus does not consider the connection distance. Authors in [109] formulated via a convex-based optimization, the problem of individual channel launch powers with the objective of maximizing the minimum SNR margin using a GN model. Extensions to take advantage of connections monitoring was presented in [110]. Advance version to improve SNR with power optimization to include higher order nonlinear effects, more specifically stimulated Raman scattering was also presented in [111].

Table. 3.1 - State-of-the-art summary

Objectives	References
O1.1 - QoT Estimation w.r.t EDFAs Gain Characteristics	[10], [12], [57], [71], [76]-[81], [92]
O1.2 - QoT Estimation w.r.t Filtering Penalties inside ROADM nodes	[10], [30], [31], [40], [41], [94]-[99]
O2 - Multi-Vendor Qtool	[10], [12], [70], [99]-[103]
O3 - Launch Power Optimization	[13], [55], [76], [104]-[111]

As discussed, the authors in [109] formulated the problem of optimizing the launch powers of all connections to maximize the sum or the minimum channel margin using a PLM based on the GN model (with fixed parameters) as a convex optimization problem. An extension of [109], that improves the SNR estimation accuracy from measurements (thus assuming an operating network / dynamic optimization) was presented in [110]. The authors proposed to probe (change the launch power) and monitor the network and use that to calculate the partial derivatives needed by the convex optimization algorithm's intermediate calculations. The limiting factors of that work were the assumption on perfect non-linear impairments monitoring, which is generally considered very hard, along with the extensive interactions with the network for probing. Additionally, the analysis was focused on a single link.

Similar Qtool accuracy issues arise in other multivariable dynamic optimization problems such as the dynamic resource allocation, automatic network reconfiguration, defragmentation operations, virtual network reconfiguration etc. [13],[112]-[118], where the optimization algorithm relies on the Qtool to perform calculations for candidate reconfigurations. Note that, the optimization algorithms for these dynamic tasks are iterative and only final solution (after series of iterations to reach convergence) is configured to the network. Hence, none of the intermediate iteration (which is calculated by the Qtool) is configured to the network. The Qtool in those related works was assumed to either have fixed parameters or aligned before the optimization task. This Qtool was used to take decisions which were afterward configured (final converged solution) to the network. For the extensive reconfigurations targeted in the above works, the Qtool can fail since its accuracy drops as the algorithm in its intermediate calculations projects the network into new states.

3.4 Conclusions

In this chapter, we reviewed the state-of-the-art of relevant works related to the goals of this Ph.D. thesis. **Table 3.1** summarizes the reviewed work upon which we build the cornerstone of the studies of this Ph.D. thesis.

The next chapters present the contributions of this Ph.D. thesis to fulfill the objectives outlined in Chapter 1.

Chapter 4

Machine Learning Assisted QoT Estimation

This Chapter is devoted for the first objective, i.e., O1 including both sub-objectives O1.1 and O1.2. To this end, we start exploring the benefits of monitoring and ML techniques to enhance the accuracy of the QoT estimation tool or Qtool. Based on the type (and quantity) of monitoring information we propose two schemes to estimate EDFA gain ripple penalties for new connection requests. Based on these two modeling schemes, we comment on the relationship between the quantity of monitoring information and the accuracy of the Qtool. We then provide a separate ML based modeling scheme to incorporate the filtering penalties at ROADM nodes to reduce design margin for the new connection requests. We initially propose independent ML based schemes (and models) for modeling of EDFA gain ripple and filtering penalties, and then a joint one. Enhancing the Qtool with the proposed supervised ML regression models yield estimates for new or reconfigured connections that account for these two effects, resulting in more accurate QoT estimation and thus a reduced design margin. Initially, we propose two supervised ML regression models, implemented with linear (with polynomial complexity) regression and Support Vector Machine Regression (SVMR), to estimate the individual penalties of the two effects and then a combined model.

4.1 Motivation and Problem Statement

For reliable and efficient network planning and operation, accurate estimation of QoT before establishing or reconfiguring the connection is necessary as discussed in detail in Chapter 2 and Chapter 3. In optical networks, a design margin is included in a Qtool to account for modeling and parameter inaccuracies, ensuring the acceptable performance. The available works in Chapter 3 clearly states that the addition of this design margin results in underutilization of the network capacity/performance. EDFAs are key the devices in WDM based optical transport networks that ensure the required connection QoT level at the receivers. However, EDFAs are the dominant source of ASE noise in these networks. Span EDFAs are operated in AGC mode with near to zero tilt to get a relatively flat gain in the C-band [119]. Although the gain tilt is maintained at zero there are still fluctuations/ripples within the gain bandwidth of EDFAs.

In Chapter 3 (Section 3.1), it is presented that the gain profile of the EDFA is not completely flat and suffers from the gain tilt and the gain ripple problems. The former is

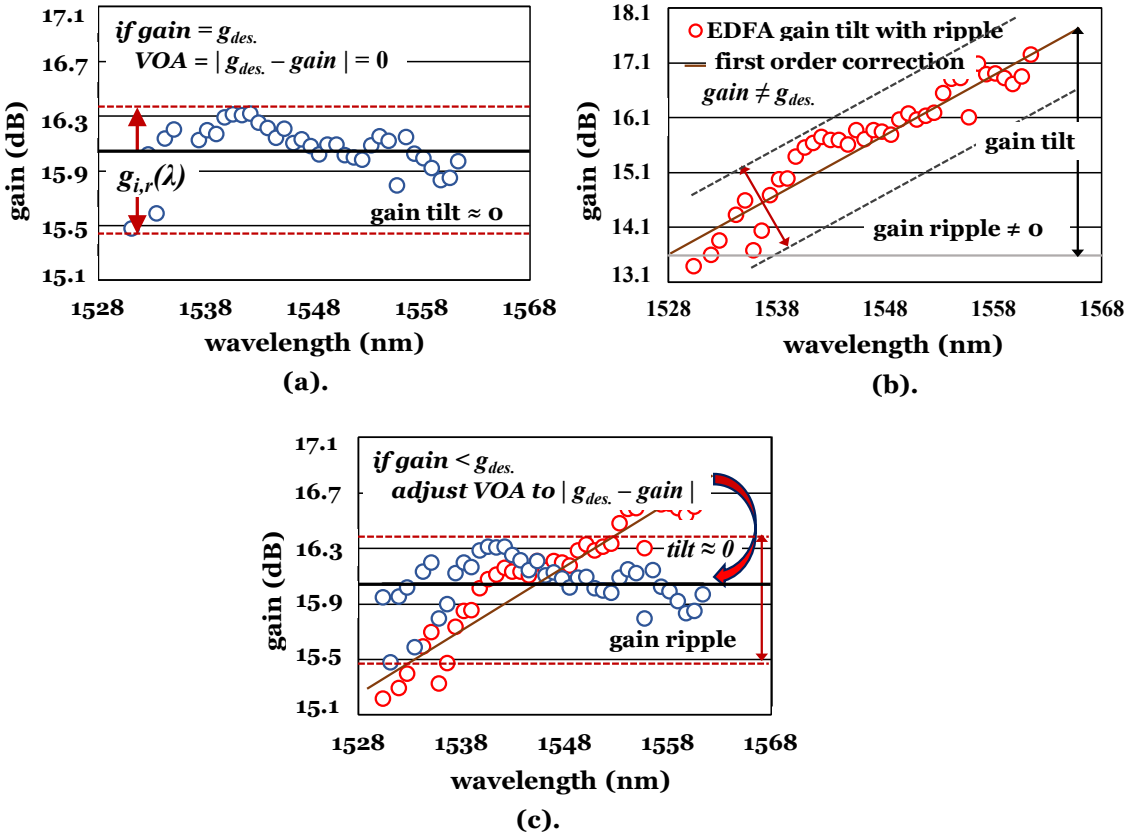


Figure. 4.1. End to end EDFA gain tilt equalization on experimentally collected (a) ripple profile with $gain = g_{des.}$, VOA = 0 dB, (b) $g_O = g_{des.}$ leading to gain tilt, and (c) first order correction by setting VOA = $|g_O = g_{des.}|$.

easily equalized with a first order/linear correction. Hence, in the following we focus on discussing the compensation of the gain tilt in this paragraph, to avoid confusion with the gain ripple addressed through the rest of the Chapter. In general, an EDFA amplifier card has a flat linear approximated output (gain ripples exist but are higher order terms, i.e. zero gain tilt, only for a specific gain value $g_{des.}$, as shown in **Figure. 4.1(a)**. This value is based on the internal design and is specified by the amplifier manufacturer. If the amplifier operating gain point g_O is not equal to $g_{des.}$ ($g_O \neq g_{des.}$), then the output suffers from a gain tilt as shown in **Figure. 4.1(b)**. This tilt can be compensated at the amplifier card level with well-known methods [72]. A variable optical attenuator (VOA) inside the amplifier is automatically adjusted to obtain the zero-tilt profile. However, even though the tilt is corrected, still gain ripples exists at the output as shown in **Figure. 4.1(c)**. From now onwards, whenever we will consider the gain ripple of an EDFA in this upcoming sections and Chapters, we will assume that the gain tilt is compensated to zero by internally readjusting its operating point to $g_O = g_{des.}$

ROADMs based on WSSs is another key element in WDM based optical networks. Current generation ROADMs support CDC multiplexing and demultiplexing at the network nodes. The WSS of the ROADMs filter and route the signal toward the assigned route that results in narrowing of transmitted signal's (connection) BW, a problem which becomes severer as the assigned path to connection traverse over more ROADM nodes.

Consequently, this causes optical filtering penalty that also needs to be modeled precisely and/or covered in the pre-allocated margins. It is well-known that filtering penalty increases exponentially as the number of traversed nodes increases, as well as that the penalty varies with respect to the modulation format and frequency grid spacing. [40], [41]. However, in deployed networks, slight variations/uncertainties in filters spectral responses (at ROADM nodes) are typical even for identical filters (procured from same vendor), while the transmitter laser and the filter central wavelengths could be misaligned. These uncertainties grow even higher for heterogeneous nodes with different types of filters (e.g., in multi-vendor scenarios [10], [96], [97]). Hence, in reality, there are always uncertainties in such filtering penalties that depends upon the spectral characteristics of the transmitted signal and the filters of the ROADM nodes. In light of the above, it is quite challenging to estimate the corresponding accumulated penalties.

In brief, to the best of our knowledge, the effects of both EDFA gain ripple and ROADM filter spectral uncertainty on the QoT estimation need further investigation. The understanding of these two effects could improve QoT estimation accuracy and reduce the margin for either reconfigured or future connection establishments. As a result, we devise estimation tools that, based on training from monitored data of established connections, can predict the independent or combined penalties of the aforementioned effects, resulting in a significant reduction in the required margins.

4.1.1 Motivation – EDFA Gain Ripple

As discussed above, typically all span EDFAs are operated in AGC mode with near to zero gain tilt, that is flat first order approximated gain in the C-band, as presented in **Figure. 4.1(c)**. To motivate our work and understand the trends of EDFA gain ripple profiles, we performed lab experiments. We used 4 different EDFAs in the optical spectrum band of ~1530 – 1563 nm with 40 WDM channels at 100 GHz spacing with central frequencies adjusted according to ITU-T standards. We operated all EDFAs in AGC mode and zero tilt by pre-adjusting their operating points. The experimental setup used is shown in the **Figure. 4.2(a)** and **Figure. 4.2(b)**. Based on the collected experimental data, we created realistic span EDFA gain profiles denoted by $g_r(\lambda)$, with maximum peak-to-peak value of ±0.5 dB. Note that, λ denotes the wavelength in C-band, and r denotes the 4 characterized EDFAs. We then simulated in VPI Transmission Maker a link with increasing number of spans, each of 80 km and chose one of the experimentally collected $g_r(\lambda)$ profile per simulation [14]. On this link, we simulated the transmission of 100 GHz spaced 40, 32 Gbaud Dual Polarization- Quadrature Phase Shift Keying (DP-QPSK) WDM channels as shown in **Figure. 4.2(a)**. We found ~1 dB of fluctuation for the channels after five spans (maximum EDFA peak-to-peak ripple of ±0.5 dB in all spans) as shown in **Figure. 4.2(c)** and **Figure. 4.2(d)**.

In general, the shape of the ripple can vary over longer time (aging), but this is a slow process. So, in both short and medium term the ripple function has a clear trend which makes its modeling possible. The observed SNR variation upto ~1 dB due to the gain ripple for only five cascaded spans (**Figure. 4.2(c)** and **Figure. 4.2(d)**) - similar findings are reported in Figure. 7 of [10]- indicating that if the gain ripple is not modeled in the QoT estimator, an equivalent margin (~1 dB) should be used to cover it.

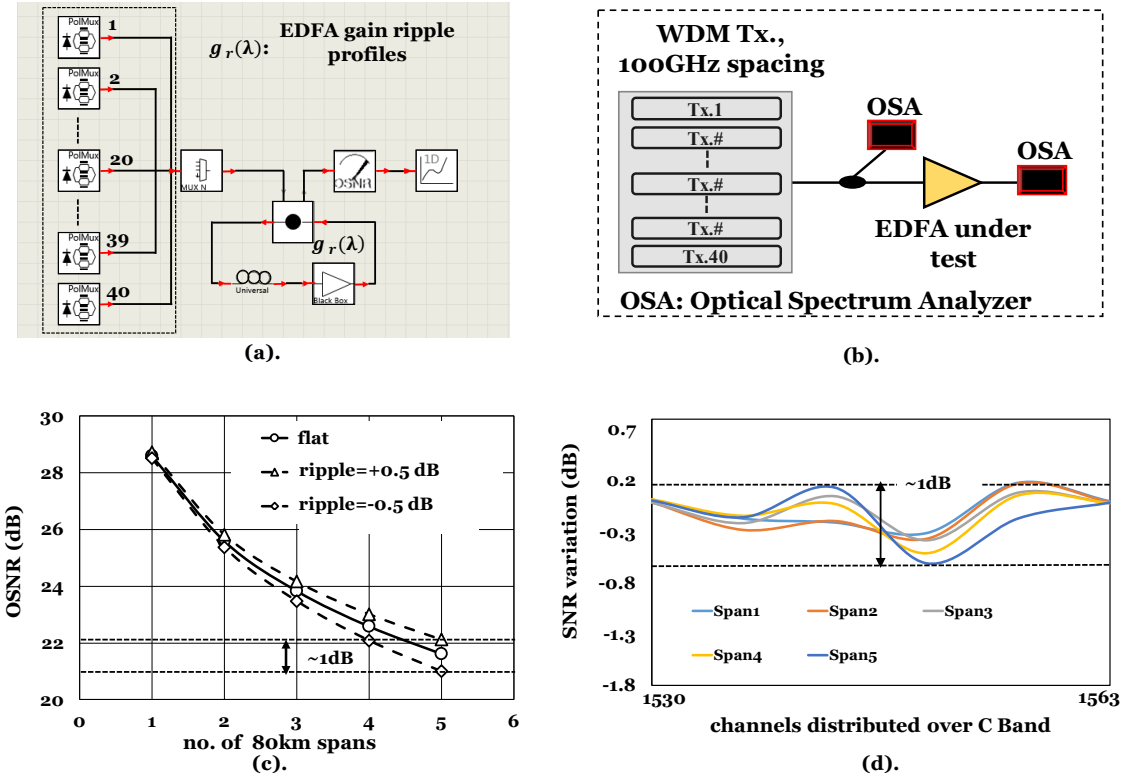


Figure. 4.2. (a) VPI Tx. Maker set up of 40 x 100G DP-QPSK WDM channels having span EDFA gain ripple profile, (b) experimental setup to capture EDFA gain ripple profile, (c) measured OSNR penalty for ripple of ±0.5dB for central channel, and (d) evolution of SNR after each span for all 40 channels.

4.1.2 Motivation – Filtering Penalties at ROADM nodes

In optical networks, connections are generally filtered and routed through multiple ROADM nodes (**Figure. 4.3(a)**) before finally detected at their receivers. We denote a connection c with central wavelength, λ_c and traversed path p_c. We also denote by $F_m(p_c, \lambda_c)$ the spectral response of the filter located at the end of m-th link along p_c. Note that the ROADM nodes might implement more than one filter. That said, we focus on just one filter per node for the sake of better clarity. In general, $F_m(p_c, \lambda_c)$ is characterized by its central frequency, pass-band bandwidth (BW), shape (Gaussian, trapezoidal etc.) and order. The central frequency of the filter is defined according to the connection's central wavelength λ_c and its bandwidth (BW) is always higher than the connection's BW and according to ITU-T grid. The filter shape and order depend on the filter type and current generation ROADMs use second order Gaussian shaped filters [30].

We simulated a setup for filter cascading in VPI Transmission Maker [14] as shown in **Figure. 4.3(b)**. **Figure. 4.3(c)** shows the spectral shape of the transmitted signal modulated at 32 Gbaud and 0.1 roll off factor with DP-QPSK modulation format (blue line) obtained with simulations in VPI Transmission Maker (**Figure. 4.3(b)**). As the

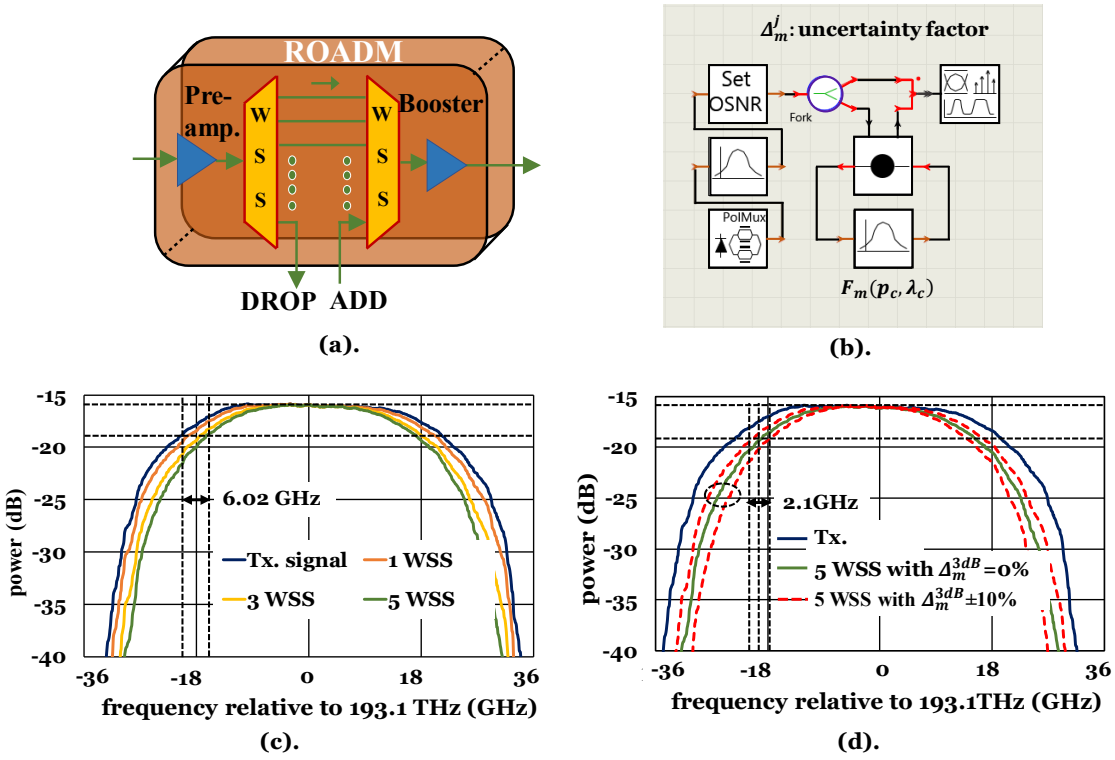


Figure. 4.3. (a) Sample S&S ROADM architecture, (b) VPI Transmission Maker set up of single channel, 100G DP-QPSK signal having uncertain filtering, Δ_m at ROADM node, (c) effective 3 dB BW reduction after cascading of 5 filter with zero uncertainty Δ_m=0, and (d) effective 3 dB BW reduction after cascading of identical filter with uncertainty, Δ_m=0%.

transmitted signal traverses the nodes/filters, its signal quality is degraded, according to the cascade of the filters' spectral responses, what is known as the filter cascading effect. **Figure. 4.3(c)** also shows the spectral shape of the received signal after one, three and five identical cascaded filters (the spectral response m is considered to be identical for all $m = 1,2,..,5$). **Figure. 4.3(c)** clearly shows that the signal quality degrades as the cascade increases, even for identical filters. In **Figure. 4.3(c)**, we observed a 3 dB BW degradation of ~6.02 GHz after 5 identical filters (solid green line). However, as already discussed, in real networks slight variations in filters spectral responses are typical even for identical filters, while the transmitter laser and the filter central wavelengths could be misaligned. Such issues result in inaccurate filtering penalty estimation. We simulated an uncertainty in 3dB BW denoted by $\Delta_{\mathrm{m}}^{3dB}$ of ±10% for each filter in VPI Transmission Maker for a cascade of five filters. **Figure. 4.3(d)**, shows the resulted signal 3 dB BW at different cascade levels. We observed ~2.1 GHz uncertainty in the signal 3 dB BW (dotted red line) at the end of the cascade/path. This uncertainty would contribute to inaccurate OSNR filter penalty estimation and should be covered by a related margin.

These preliminary results, obtained from the above described experiments and simulations for EDFA gain ripple and spectral uncertainties of filters, motivated us to explore these effects further. Our goal is to integrate them in QoT estimators to yield higher accuracy or lower design margin for future connections, as discussed in the upcoming sections of this Chapter.

4.2 GN Model Extension

This section outlines the mathematical extension on the existing GN model based standard Qtool to capture the effect of EDFA gain ripple. We outline the modeling used in a standard and extended PLM to account for the gain ripple. In general, a PLM calculates ASE and NLI noises and adds margins on top for simplifications/ noise contributing factors that it partially covers.

4.2.1 Mathematical Modeling of Ripple Unaware and Aware EDFA Gain

Let us focus on a multi-span link as depicted in **Figure. 4.4**. Current generation commercial EDFAs are dual staged having low NF, and a large dynamic range (up to 15 dB) [120]. Moreover, in real networks, a Dynamic Gain Equalizer (DGE), is used to compensate the cumulative EFDA gain ripple effect on multi-span links, resulting in almost flat output power [121]. For a typical scenario of a 6 span link the DGE is applied at the 3rd span (mid-span) as shown in **Figure. 4.4**. For longer links a new DGE is installed every ~6 spans. Even though the use of DGEs, residual gain ripple effect is still present, affecting the overall QoT estimation accuracy. The ASE noise (linear noise) estimation is straightforward to model and depend upon the gains and NFs of the EDFAs along the path. For NLI several models were proposed and validated as already discussed in Chapter 2. The GN model [50] is computationally very efficient while maintaining good accuracy. Hence, we implemented it and extended it to account for the additional wavelength dependent gain ripple penalties.

Assume now that we have a connection $c = (p_c, \lambda_c)$ and use the standard GN model that does not model the gain ripple. Assuming link c with N_s spans of **Figure. 4.4**, the power spectral density (PSD) of ASE noise at the link end accumulated over span EDFAs is given by

$$G_{ase,m} = \sum_{n_s=1}^{N_s} NF_{n_s} * h * v * \left(g_{n_s} - 1\right) \tag{4.1}$$

where, NF_{n_s} and g_{n_s} is the noise figure and average gain of n_s- th span EDFA, h is the Planks constant, v is the reference frequency (typically 193.1 THz).

Let us now assume that link m has N_{ch} WDM channels, similar to **Figure. 4.4**. For channel n, we assume transmitted power P_n and symbol rate, R_n. As stated in Chapter 2, with incoherent noise accumulation assumption, the PSD of NLI noise at the link end can be calculated as the sum of the NLI noise produced in each single span. With the assumption of noise to be additive Gaussian, the GN model calculates PSD of NLI noise at link end, as

$$G_{nli,m}(\lambda_c) = \frac{16}{27} \sum_{n_s=1}^{N_s} \gamma_{n_s}^2 L_{eff,n_s}^2 \cdot \sum_{n=1}^{N_{ch}} G_{n,n_s} G_{n,n_s} G_{c,n_s} (2 - \delta_{nc}) \Psi_{\lambda_n,\lambda_{n,n_s}}$$
$$\cdot \prod_{n_s'=1}^{n_s-1} g_{n_s'}^3 e^{-6\alpha_{n_s'} L_{n_s'}} \cdot \prod_{n_s'=n_s}^{N_s} g_{n_s'} e^{-2\alpha_{n_s'} L_{n_s'}} \tag{4.2}$$

where Ψ is the phased array factor which under the assumption of incoherent accumulation is given by Eq. (128) and Eq. (129) of [50]; L_{n_s} is the n_s span length; γ_{n_s} is its non-linear coefficient; L_{eff,n_s} is its effective length; α_{n_s} is its attenuation coefficient, g_{n_s} is the gain of the span EDFA; $G_{n,ns}$ is the PSD of the n-th WDM channel (n =1, 2… , N_{ch}) at the start of the n_s-th span; δ_{nc} is the factor that distinguishes the SCI and XCI terms as given by Eq. (122) of [50]. The detailed derivation of Eq. (4.2) along with parameters description are available in [50].

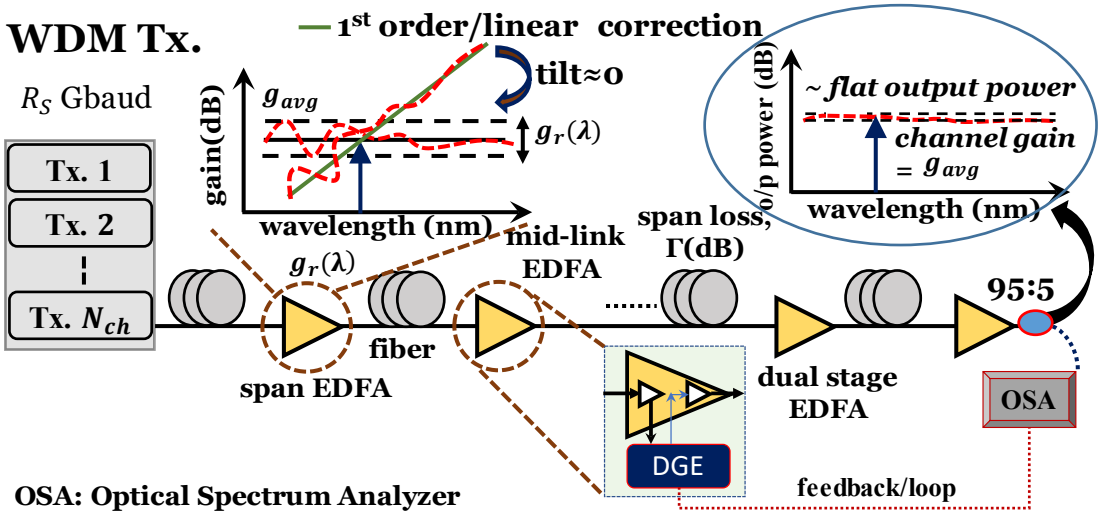


Figure. 4.4. WDM link indicating span EDFAs with gain ripple, $g_r(\lambda)$ and DGE location with dual stage EDFA resulting in almost flat output at link end.

A standard PLM based Qtool assumes that each fiber span loss ($e^{-2\alpha_{n_s}L_{n_s}}$ of Eq. (4.2)) is exactly compensated at the end of each span by the gain of span EDFA (g_{n_s}). Also, a standard Qtool assumes a flat/wavelength independent EDFA gain without ripple. Under this assumption the per span PSD of side channels ($G_{n,ns}$) and channel of interest at λ_c ($G_{c,ns}$), depends only on baudrate, R_n and per channel transmitted power, P_n. Thus, a standard PLM makes following assumptions on three terms of Eq. (4.2) as

$$\left\{\begin{array}{l} \prod_{n_s'=1}^{n_s-1} g_{n_s'}^3\, e^{-6\alpha_{n_s'}L_{n_s'}} = 1 \\ \prod_{n_s'=n_s}^{N_s} g_{n_s'}\, e^{-2\alpha_{n_s'}L_{n_s'}} = 1 \\ G_{n,n_s} = \dfrac{P_n}{R_n} \text{ for } n = 1, \ldots N_{ch}, n_s = 1, \ldots, N_s \end{array}\right\} \tag{4.3}$$

The total NLI noise at channel centered at λ_c, $G_{nli,m}(\lambda_c)$ is calculated by using the assumptions of both Eq. (4.3) in Eq. (4.2) and is independent of per span wavelength dependent gain ripple effect. Also, the distortion introduced by a span EDFA ripple in the PSD of the channels, which are input for the next span are not considered for the calculation of NLI noise of the next span. We denote the PSD of NLI calculated with the ripple unaware equations, that is, Eq. (4.3) used in Eq. (4.2), as $G_{nli\_RU,m}(\lambda_c)$.

Assuming now a network and a connection $c = (p_c, \lambda_c)$ crossing link m on its path p_c. The SNR at the end of link m calculated with the ripple unaware GN model, is given by

$$SNR_{RU,m}(p_c, \lambda_c) = \frac{G_{o,m}(\lambda_c)}{G_{ase,m} + G_{nli\_RU,m}(\lambda_c)} = \frac{G_{o,m}(\lambda_c)}{G_{Noise\_RU,m}(\lambda_c)} \tag{4.4}$$

where $G_{o,m}(\lambda_c)$ is the output signal PSD at the end of link m which is equal to P_c/R_c for the assumptions of Eq. (4.3).

A typical assumption for a connection that traverses multiple links is that the inverse SNR per link is additive (incoherent). With the ripple unaware PLM of Eq. (4.4) the total SNR at end of connection c is given by

$$[SNR_{RU}(p_c, \lambda_c)]_{dB} = \left[1 \Big/ \left(\sum_{m \in p_c} SNR_{RU,m}^{-1}(p_c, \lambda_c) \right) \right]_{dB} + design\ margin_1 \tag{4.5}$$

We now discuss how to extend the GN model to account for EDFA gain ripples. To model that, we assume known EDFA gain profiles for the span EDFAs, denoted by $g_{n_s}(\lambda)$ (for λ ranging the full C-band), which is broken down as follows

$$g_{n_s}(\lambda) = g_{n_s,avg} \cdot g_{n_s,r}(\lambda) \tag{4.6}$$

where $g_{n_s,avg} = g_{n_s}$ (= span loss) is the average value and $g_{n_s,r}(\lambda)$ is the wavelength dependent ripple.

Each $g_{n_s,r}(\lambda)$ alters the current span signal PSD G_{n,n_s} and of the next span depending upon the current span. This clearly means that to accommodate the ripple effect in $G_{nli}(\lambda_c)$ calculations, the assumptions in Eq. (4.3) need modifications and extensions at span level as follows:

$$\left\{ G_{n,n_s} = \begin{pmatrix} \prod_{n_s'=1}^{n_s-1} [g_{n_s',avg} \cdot g_{n_s',r}(\lambda_c)]_{n_s'}^3 \, e^{-6\alpha_{n_s'} L_{n_s'}} \neq 1 \\ \prod_{n_s'=n_s}^{N_s} \left[g_{n_s',avg} \cdot g_{n_s',r}(\lambda_c) \right]_{n_s'} e^{-2\alpha_{n_s'} L_{n_s'}} \neq 1 \\ \frac{P_n}{R_n};\ n_s = 1 \\ \sum_{n_s'=1}^{n_s-1} G_{n,n_s'} \cdot g_{n_s',r}(\lambda_n);\ n_s \neq 1 \end{pmatrix}, n = 1, .. N_{ch} \right\} \tag{4.7}$$

This Eq. (4.7) is then substituted in Eq. (4.2) to calculate the ripple aware NLI noise $G_{nli\_RA,m}(\lambda_c)$ that takes into account per span wavelength dependent ripple effects in gain as well as the PSD of the lighted channels. Also, DGE altered power profile (from mid span) modifies the NLI noise contribution for all onward spans. We can capture this with Eq. (4.7) by setting the DGE applied channels PSD $G_{n,n_{DGE}}$ at the specific span n_{DGE}. Using the extended GN model, we obtain the SNR calculated at the end of link m,

$G_{n,n_{DGE}}$, $G_{nli\_RA,m}(\lambda_c)$. So, Eq. (4.4), is now changed to Eq. (4.8) as

$$SNR_{RA,m}(p_c,\lambda_c) = \frac{G_{o,m}(\lambda_c)}{G_{ase,m} + G_{nli\_RA,m}(\lambda_c)} \quad or$$

$$SNR_{RA,m}(p_c,\lambda_c) = \frac{G_{o,m}(\lambda_c)}{G_{ase,m} + G_{nli\_RU,m}(\lambda_c) + G_{RA,m}(\lambda_c)} = \frac{G_{o,m}(\lambda_c)}{G_{Noise\_RA,m}(\lambda_c)} \quad (4.8)$$

Where $G_{Noise\_RA,m}(\lambda_c)$ corresponds to the total noise at the end of link m estimated by the PLM with EDFA gain ripple information. With this PLM, plus an additional margin, the total SNR calculated at end of connection $(c = p_c, \lambda_c)$ traversing L links is given by

$$[SNR_{RA}(p_c,\lambda_c)]_{dB} = \left[1\Big/\left(\sum_{m\in p_c} SNR_{RA,m}^{-1}(p_c,\lambda_c)\right)\right]_{dB} + design\ margin_2 \quad (4.9)$$

Relying on above derivations, assume now that we have a connection $c = (p_c, \lambda_c)$ and use the standard GN model that does not model the EDFA gain ripple. The SNR at the end of link m on the path p_c calculated with the standard GN model (*ripple unaware*-RU) is denoted by $SNR_{RU,m}(p_c,\lambda_c)$, respectively. We can then accumulate the inverse of SNR over the links of the path p_c to obtain the total SNR calculated at connection's end as $SNR_{RU}(p_c,\lambda_c)$ with Eq. (4.5). We denote the Qtool that uses the standard GN which is ripple unaware as Q_{RU}. Such Qtool, since it is ripple unaware, needs to use a margin (part of $design\ margin_1$) on top of $SNR_{RU}(p_c,\lambda_c)$ to account for the penalties due to neglected wavelength dependent gain ripples.

As discussed, EDFAs have gain ripples, which makes the power spectral density (PSD) of each channel to change after each traversed span (even for uniform launch power and baudrate). Assuming known ripple profiles $g_{n_s}(\lambda_c)$ of each span n_s we can calculate the PSD of each channel and also the PSD of NLI noise at the end of each span. In deployed networks with DGEs installed at certain spans (as shown in **Figure. 4.4**), the DGE alters the power profile of the applied span to flatten the output power at the end of the covered spans via a feedback loop. We can account for this in the modeling, by setting on the span on which the DGE is applied the signals PSD to be equal to the DGE feedback power profile. This is calculated by the known EDFA ripple profiles to obtain (almost) zero ripple at the end of the link.

The above described GN model is thus *ripple aware* (RA). We calculate the ASE and NLI noise (including effect of gain ripples) per link, and then, using the per link inverse SNR addition, we obtain the total SNR for connection c, denoted by $SNR_{RA}(p_c,\lambda_c)$ as given by Eq. (4.9). Then, on top of that, we add the $design\ margin_2$ to cover for modeling inaccuracies, excluding the gain ripple. We call the Qtool that uses this ripple aware PLM as Q_{RA}.

Note that in case of ripple aware Q_{RA} the total noise generated at the end of path is closer to the reality, compared to ripple unaware Q_{RU}, since it models/accounts for the wavelength dependent ripple. This leads to better accuracy and a lower design margin *i.e.* $design\ margin_2 < design\ margin_1$.

4.3 ML Assisted EDFA Gain Ripple Modeling

In the previous section we outlined the mathematical modeling of the EDFA gain ripple with some extensions on the existing GN model based modeling scheme. However, such modeling is feasible if we accurately know the gain profiles of the EDFAs $g_{n_s}(\lambda_c)$ at each span. Such assumptions are rather unrealistic. They would require the measurement of all EDFAs in the network, which would have to be repeated because they would vary with time (caused by aging, traffic changes, etc.). So, in this section we propose to use monitoring information in an operating network combined with ML to model the penalties due to EDFA gain ripple effect. The proposed ML model is trained with the current network state and then used for estimating the *unseen* penalties of future connections, achieving higher QoT estimation accuracy and requiring lower margins.

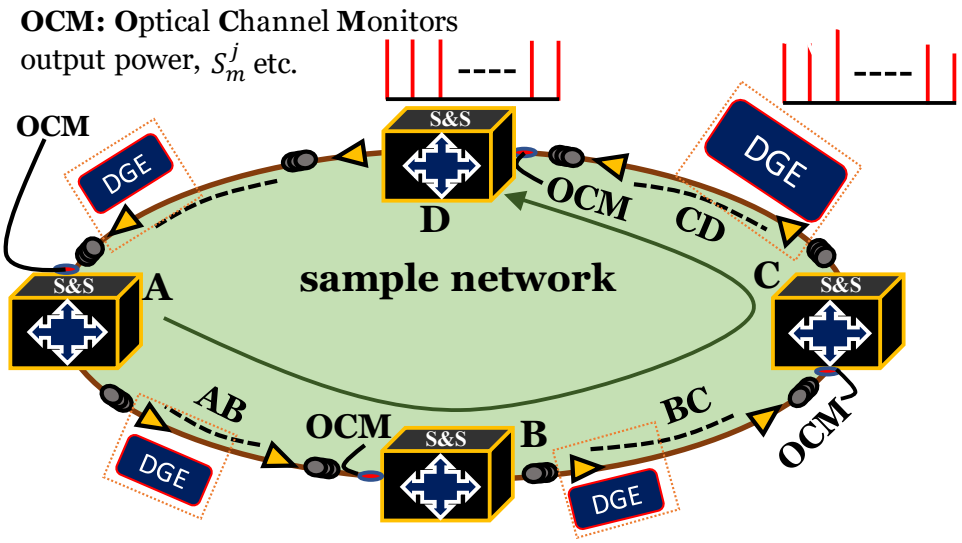


Figure. 4.5. Sample 4-node network depicting monitoring port locations and DGE placement locations.

We assume an optical network with established connections and their attributes also referred to as the state of network at a given time denoted by C. Note that C contains the attributes for each connection such as, the traversed path p_c, central wavelength λ_c, transmitted power etc. We also assume that the network has optical channel monitors (OCMs) installed at the end of each link [11], [122] that is, before each ROADM node. In **Figure. 4.5**, we show a simple network where we indicate the locations of OCMs and the main parameters extracted (output power, set of j attributes from monitored spectra $S_{p_{c,m}}^j$ etc.). In **Figure. 4.5**, we also indicate the location of the DGEs that flatten the EDFA gain ripples before every ROADM node, that is, at the end of each link. We also assume that we can monitor the feedback power profiles of the DGEs and also the electrical SNR information at the coherent receiver at the transmission endpoints [11], [15]. Finally, we assume that all monitoring information is made available through a suitable control plane architecture as the one presented in Chapter 1 and Chapter 2.

We here propose *two schemes* to estimate EDFA gain ripple penalties based on the type (or quantity) available monitoring information, *i.e.*

i) ***scheme-1*** - electrical SNR monitoring information (SNR monitors) at each node, and

ii) ***scheme-2*** - OCMs installed at the intermediate nodes and coherent receiver (electrical SNR) only at the destination node.

We start with a standard Qtool that is gain ripple unaware, Q_{RU}. We use the monitored information from established connections to calculate the estimation error of the standard unaware Qtool due to the considered EDFA gain ripple effect. Based on those we train a supervised ML regression model to estimate the error between the standard Qtool and the monitored quality values. The model is then used to calculate the related corrections for new (unestablished) connections. Then for new connections, we use the trained ML model as a correction on the standard unaware Qtool estimation. The main difference of the two considered schemes lay in the type and quantity of monitoring information, the ML feature extraction and the modeling.

4.3.1 Scheme-1 Formulation

We assume a ripple unaware Qtool Q_{RU}, as discussed in the last section, which calculates the end-to-end noise of each established connection as $G_{Noise\_RU}(p_c, \lambda_c)$, and the related per link noise as $G_{noise\_RU}(p_c, \lambda_c)$, for all connections $c \in C$. In this scheme, we assume that each node is equipped with coherent receiver (or SNR monitors) and it is possible to extract the electrical SNR information at each node along the path p_c for all connections $c \in C$. As a fundamental step for this scheme, using the per node monitored SNR information we can improve such estimations. We use the ripple aware Q_{RA} formulas with flat EDFA gain profiles, but we use the per node monitored electrical SNR information. We call this monitoring information-based ripple aware $\tilde{Q}_{RA}$ Qtool. Note that C contains attributes for a connection such as, the traversed links, central wavelength, launch power etc. We now monitor the SNR, hence the noise at the path (destination node) level $\tilde{G}_{Noise\_RA}(p_c, \lambda_c)$ and at the link level $\tilde{G}_{noise\_RA}(p_c, \lambda_c)$. We denote the set of estimated noise values by $\tilde{Q}_{RA}$ per link $\tilde{Y}_{RA\_l}(C)$ and end-to-end (path) for all established connections by $\tilde{Y}_{RA\_c}(C)$.

We then monitor the electrical SNR of the established connections and thus their noise (ASE and NLI) after each traversed link $Y_{RA\_l}(C)$ and at the coherent receiver/ path level, $Y_{RA\_c}(C)$ and store it in the Qtool database. This data serves as the ground truth, it defines the true $G_{Noise\_RA}(C)$, with zero margin. We denote the difference of $\tilde{Y}$ and G_{Noise_RU} (standard Qtool) as *i)* $E_l(C) = \tilde{Y}_{RA\_l} - G_{noise\_RU}$ which is a vector with the ripple penalties at the end of each link, accumulated over the links, spans and *ii)* $E_c(C) = \tilde{Y}_{RA\_c} - G_{Noise\_RU}$, which is a vector with the ripple penalties at connection's end, accumulated over all the used links. Note that, we assume ideal monitors with zero monitoring noise, hence $\tilde{Y} = Y$. We let E_{R_S1} be the concatenation of both penalty vectors E_l and E_c. Note that the subscript $S1$ represents the scheme-1 to distinguish between this scheme to the other one tackled in the next subsection since both share a common objective of modeling EDFA gain ripple penalty. From connections attributes, C, we extract features which depend on connections' routes and central wavelengths. To be more specific, for each connection c we assign its used wavelength on the links that it utilizes/traverses. Also, links used in the path are one hot encoded (traversed links in the path are one and rest

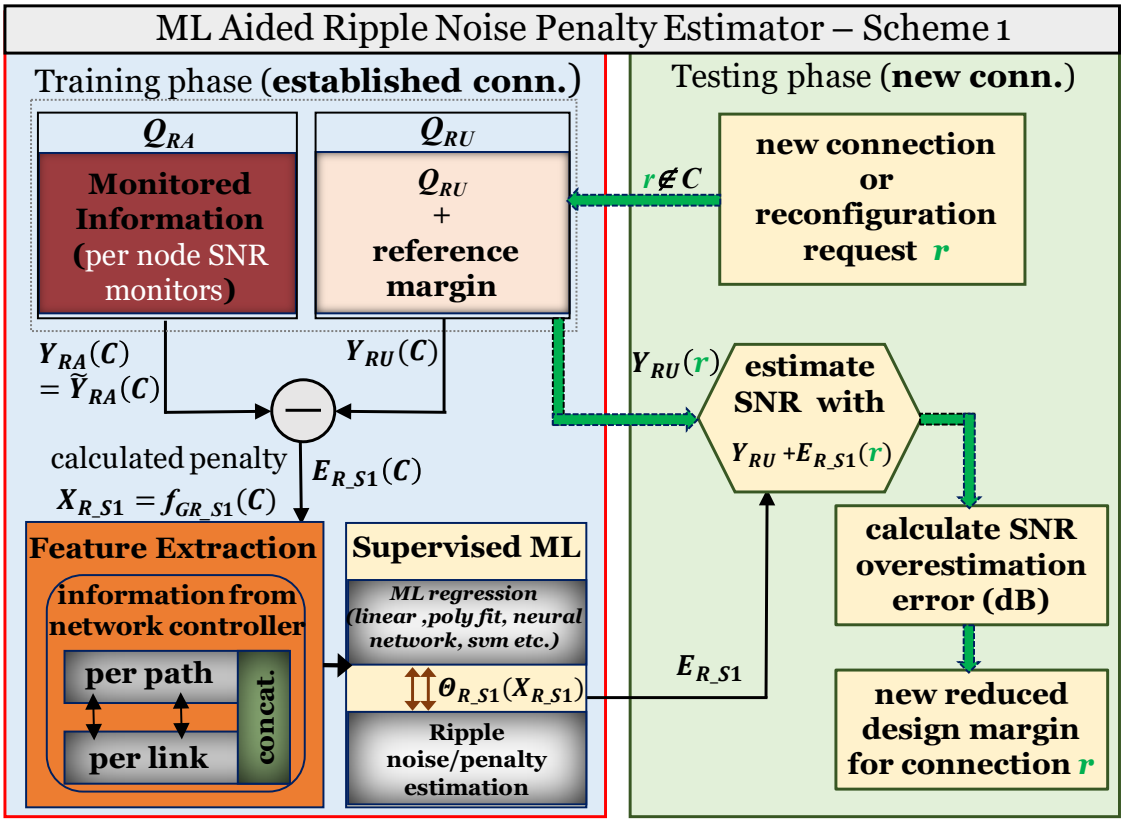


Figure. 4.6. Overall flowchart of ML-based penalty estimator utilizing Scheme-1 (i.e., train/test) for EDFA gain ripple case.

are zero). Additional to these features, a *bias* is also considered to account for any monitoring calibration error and for the non-zero equalized tilt. The per link and per path features are merged into a single gain ripple feature matrix , $X_{R\_S1} = f_{GR\_S1}(C)$. The feature matrix enables the correlation between connections crossing the same link while accounting for their utilized wavelengths. Our goal is to identify the function $\Theta_{R\_S1}(X_{R\_S1}) \approx E_{R\_S1}$ that maps well the feature matrix X_{R_S1} to the penalty E_{R_S1} generated due to the gain ripple. Based on the monitored information of established connection, we can train supervised ML models on the above features and their corresponding labels, E_{R_S1}. Under the hood, Θ_{R_S1} implements a wavelength dependent ripple penalty function per link.

We rely on ML for training and fitting of X_{R_S1} on E_{R_S1} and finding the function Θ_{R_S1}. Assuming a new connection request $r \notin C$, we will use Q_{RU} to obtain the total noise G_{Noise_RU}. Then we use our trained ML model Θ_{R_S1} to estimate the ripple noise penalty on the new connection $\Theta_{R\_S1}(f_{GR\_S1}(r))$ and estimate the total noise including ripple as $G_{Noise\_RU} + \Theta_{R\_S1}(f_{GR\_S1}(r))$. The testing error will be identified once we establish the connection, monitor the SNR and the noise at the receiver and compare it to that. The interactions between the collected monitoring information, the Qtool Q_{RU} and the ML assisted ripple noise penalty estimation are depicted in **Figure. 4.6**.

4.3.2 Scheme-2 Formulation

As a first step for this scheme, we propose to only use the per link monitored DGE power profiles to improve QoT estimation. Note that we are relying here on optical information at the intermediate nodes, *i.e.,* DGE power profile information obtained from the OCMs of **Figure. 4.5**. Compared to Scheme-1, this scheme bypasses the limitation of using electrical SNR information at intermediate nodes. Hence this scheme provides the accurate QoT estimation considering EDFA gain ripple penalties in a cost-effective way. However, there is a tradeoff between scheme-1 and scheme-2 in terms of quantity of monitoring information and accuracy, as discussed in the results section.

DGE-based Equivalent Link Model

The ripple aware Qtool (Q_{RA}) described above assumes that the gain profiles of all EDFAs are known with good accuracy. This is indeed a strong and unrealistic requirement. Therefore, we propose to use monitoring information (DGE attenuation profile) in an operating network combined with ML to model the penalties due to the EDFA gain ripple effect. Then we use our ML model to estimate the ripple penalties of the future connections. To be more specific, the first step of this scheme uses monitoring information from mid span DGEs per link to create an equivalent link model m_{REA}. The DGE monitoring information pertains to the applied attenuation profile (for established connections) to get the flatten output at the link end. Consider that the DGE is applied on span d then the attenuation profile of that span for connection c is denoted by $a(m, d, \lambda_c)$.

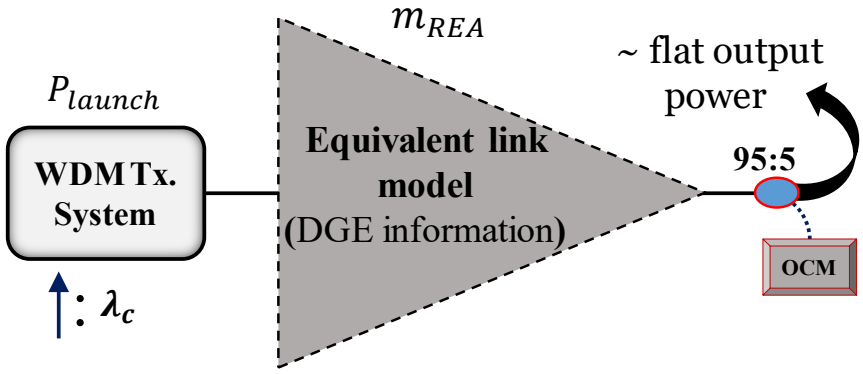


Figure. 4.7. Equivalent link model (for Figure. 4.4) based on collected DGE attenuation information.

For the WDM link m with uniform transmitted launch power (e.g. 0 dBm for all established connections) $P_{launch}(m)$, we can map an equivalent power profile (in dB scale) to span d as

$$P_{DGE}(m, d, \lambda_c) = P_{launch}(m) - a(m, d, \lambda_c) \quad (4.10)$$

Note that since the gain of the previous EDFAs (before span d) is also not flat. Thereby the actual power profile upto span d also accumulates cascaded EDFA gain ripple effect of previous span EDFAs in the output power of span d EDFA as

$$\left[\prod_{n_s=1}^{d} g_{avg} \cdot g_r\right]_{(m,n_s,\lambda_c)} \cdot P_{launch}(m, n_s, \lambda_c) \tag{4.11}$$

This is the ideal output power profile without any feedback from OCMs to DGE. Note that $P_{DGE}(m, d, \lambda_c)$ is the power profile at span d tailored by the DGE to get almost flat output power for connection c centred at λ_c. Hence the monitored $P_{DGE}(m, d, \lambda_c)$ is different from the one that is without the use of DGE (ideal one stated above). Since the number of spans within the link m is known, it is possible to replace the multispan link of **Figure. 4.4** with an equivalent link model m_{REA} as shown in **Figure. 4.7**.

For this, firstly we convert the $P_{DGE}(m, d, \lambda_c)$ to output signal PSD of span d. For the connection c, we denote PSD of output signal at span d as $G_o(m, d, \lambda_c)$. The $G_o(m, d, \lambda_c)$ depends upon the baudrate R_S of λ_c and is given by

$$G_o(m, d, \lambda_c) = \frac{P_{DGE}(m, d, \lambda_c)}{R_S(\lambda_c)} \tag{4.12}$$

We then use our extended GN model (discussed in the last section) and feed it with $G_o(m, d, \lambda_c)$ to calculate the approximate mid-span noise PSD of link m at channel λ_c, denoted by $\tilde{G}_{Noise\_mid}(m, d, \lambda_c)$. In general, the worst case of gain ripple occurs when all spans EDFAs are assumed to have the same ripple profile. Thus, we assume that all spans have the same ripple profile. Under this worst-case assumption, we calculate the approximated total accumulated noise PSD at end of link m having N_S spans as

$$G_{Noise\_REA}(m, \lambda_c) = N_S.\left(\tilde{G}_{Noise\_mid}(m, d, \lambda_c)\right) \tag{4.13}$$

Note that $G_{Noise\_REA}(m, \lambda_c)$ contains an approximation error for link m as the equivalent model is made up from monitored information at a single point, extended with a worst-case assumptions, as described above. We use this per link equivalent model along with inverse linear additive assumption as Eq. (4.5) and Eq. (4.9) to calculate the overall accumulated noise along path p_c, $G_{Noise\_REA}(p_c, \lambda_c)$ for connection c, and $SNR_{REA}(p_c, \lambda_c)$. Since, we use the link equivalent model m_{REA} which approximates multiple cascaded EDFAs and fiber spans and contains an approximation errors, our SNR estimation is not perfectly accurate. However, as shown in the results section, it is still better than the standard Qtool Q_{RU}. We call this *ripple dynamic gain equalizer aware* (REA) Q_{REA} Qtool. We denote the set of estimated values for all established connections C by $Y_{REA}(C)$.

We then monitor the electrical SNR of the established connections and thus their noise (ASE and NLI) at the coherent receiver/ path level, $Y_{RA}(C)$ and store it in the Qtool database. Similar to previous case, this data serves as the ground truth (fair comparison between two schemes), it defines the true $G_{Noise\_RA}(C)$, with zero margin. We denote the difference of the real/monitored G_{Noise_RA} and the estimated G_{Noise_REA} at path level as $E_{R\_S2}(C) = Y_{REA}(C) - Y_{RA}(C)$. The subscript $S2$ here denotes scheme-2 for EDFA gain ripple modeling. E_{R_S2} is a vector that includes the estimation errors of Q_{REA} of the established connections due to the real gain ripples. From connections attributes, C, we extract features which depend on connections' routes and central wavelengths. To be

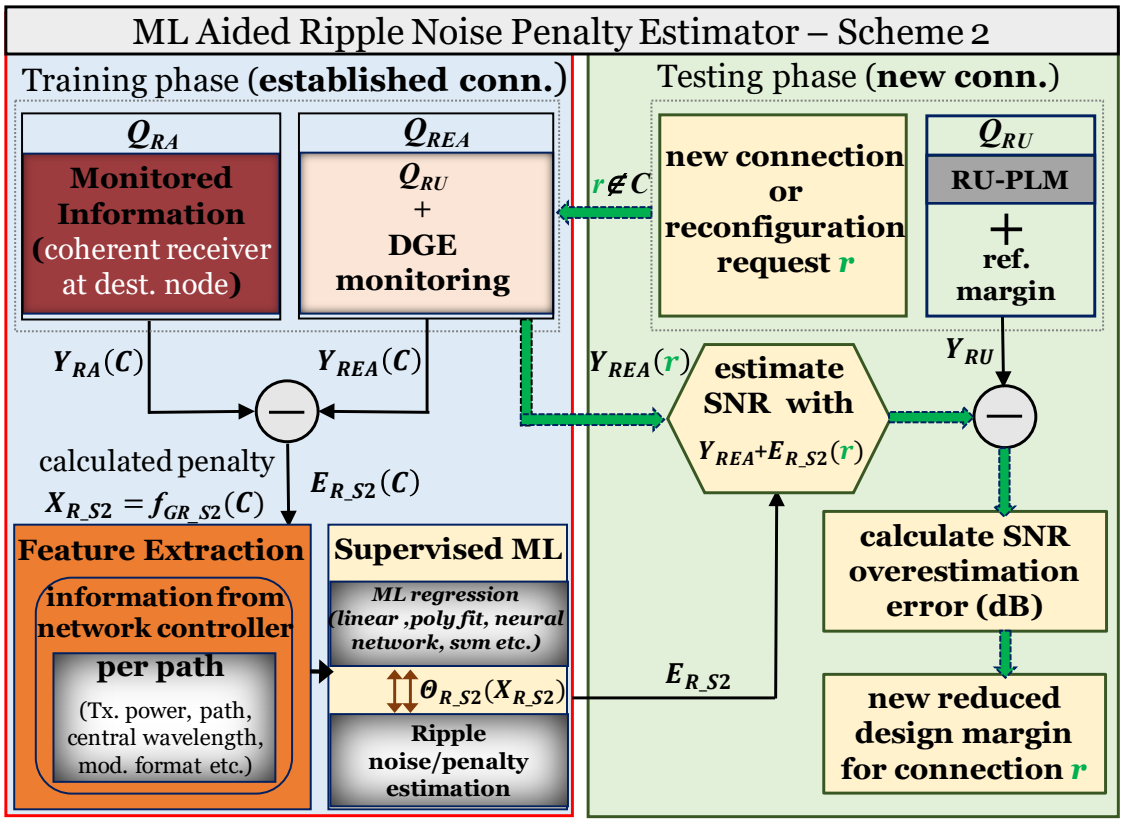


Figure. 4.8. Overall flowchart of ML-based penalty estimator utilizing Scheme-2 (i.e., train/test) for EDFA gain ripple case.

more specific, for each connection c we assign its used wavelength on the links that it utilizes (links used in the path are one hot encoded). Note that feature matrix is similar to the previous scheme with the main difference of only end to end information of SNR is accounted compare to per link information availability of Scheme-1. Similar to the previous Scheme-1, a bias is also considered to account for any monitoring calibration error and for the non-zero equalized tilt. The per connection features along with the bias term are merged into a gain ripple features matrix $X_{R\_S2} = f_{GR\_S2}(C)$. The feature matrix enables the correlation between connections (end to end) crossing the same link while accounting for their utilized wavelengths. The goal is to identify the function $\Theta_{R\_S2}(X_{R\_S2}) \approx E_{R\_S2}$ that maps well the feature matrix X_{R_S2} to the penalty E_{R_S2} generated due to the gain ripple. We again rely on ML for training and fitting of X_{R_S2} on E_{R_S2} and finding the function Θ_R. Assuming a new connection request $r \notin C$, we will use Q_{REA} to obtain the total approximated noise $G_{Noise\_REA} = Y_{REA}(r)$. Then we use our trained ML model Θ_{R_S2} to estimate the ripple noise penalty on the new connection $\Theta_{R\_S2}(f_{GR\_S2}(r))$ and estimate the total noise including ripple as $G_{Noise\_REA} + \Theta_{R\_S2}(f_{GR\_S2}(r))$. The testing error will be identified once we establish the connection, monitor the SNR and the noise at the receiver and compare it to that. The interactions between the collected monitoring information, the Qtool Q_{REA} and the ML assisted ripple noise penalty estimation are depicted in **Figure. 4.8**.

4.4 Modeling Filter Penalties at ROADM Nodes

The main assumptions regarding the network architecture and monitoring location are similar to the previous modeling scheme for EDFA gain ripple (**Figure. 4.5**). However, we assume flat EDFA gain ripple profile to focus specifically on filtering penalties. We also provide the joint scheme where both the effects are present simultaneously in next section. We now focus on the mathematical formulation and ML based modeling scheme to accurately estimate the filtering penalties generated at the ROADM nodes.

4.4.1 Mathematical Formulation

We now discuss briefly the mathematical formulas used for standard and extended PLM to account for the uncertainties in ROADM filtering penalties. As the connection traverse multiple nodes, the OSNR filter penalty accumulates in a nonlinear fashion as shown in **Figure. 4.9**. We used VPI Transmission Maker (**Figure. 4.3(a)**) to measure the penalty for three different modulation formats at 32 Gbaud with roll off of 0.1. We consider the standard CDC switch & select (S&S) architecture of ROADM node as shown in inset of **Figure. 4.9(a)**, although our approach can be also used for broadcast & select (B&S) architecture, or a mix. In S&S, two WSSs are used to route the signal to the outgoing link together with two EDFAs acting as pre and booster amplifier, respectively. For add/drop signal at source/destination node, we assume that the penalty comes from only one WSS/filter. Depending on add/drop or crossing direction, different filters are encountered in the ROADM.

To account for this and for the cascading effect, we account the filters by considering subpaths. So, extending the above notation used for the modeling of EDFA gain ripple, for connection c that uses path p_c and wavelength λ_c we denote by $p_{c,m}$ the subpath from the source/transmitter up to link m. We also denote by $S_{p_{c,m}}(p_c, \lambda_c)$ the signal spectrum at the end of link m, due to the cascade of previous filters over the subpath, $p_{c,m}$. This is given by

$$S_{p_{c,m}}(p_c, \lambda_c) = \prod\nolimits_{k \in p_{c,m}} F_k(p_c, \lambda_c) \tag{4.14}$$

where $F_k(p_c, \lambda_c)$ is the spectral response of filter k in path p_c. In the following, we drop λ_c and p_c for simplicity. From $S_{p_{c,m}}$ we can extract a set of attributes, indexed by j, that reflects the key properties of the cascade up to link m, denoted by $S^j_{p_{c,m}}$. Such attributes can be the 3dB BW, cascaded filter central frequency, signal distribution parameters, etc., or $S^j_{p_{c,m}} = \{S^{3dB}_{p_{c,m}}, S^{fc}_{p_{c,m}}, ..., S^{sym.}_{p_{c,m}}\}$. However, variations within spectral responses of filters and misalignments result in uncertainties in these attributes $S^j_{p_{c,m}}$ which we denote by $\Delta^j_{p_{c,m}} = \{\Delta^{3dB}_{p_{c,m}}, \Delta^{fc}_{p_{c,m}}, \dots, \Delta^{sym.}_{p_{c,m}}\}$.

Extending this notation to PLM/Qtool, let us consider connection $c = (p_c, \lambda_c)$. The standard PLM would assume that the filters along its path are perfectly identical, $\Delta^j_{p_{c,m}} = 0$. The standard PLM estimates an attribute $s^j_{p_{c,m}}$ (lowercase as opposed to real attributes $S^j_{p_{c,m}}$) at the end of link m, and employs an attribute dependent filter penalty

function, w^j, to calculate the attribute at the next link $s^j_{p_{c,m+1}}$=$w^j(s^j_{p_{c,m}}, F_i)$. An example of such a function for the attribute $j = 3\ dB$, w^{3dB}, is shown in Figure. 4.9.(a), coming from VPI simulations. We see that the 3 dB BW ($s^{3dB}_{p_{c,m}}$) degrades as the function of the cascaded filters. The standard PLM would also employ a function q to map the attributes $s^j_{p_{c,m}}$ to modulation format dependent OSNR penalty at each link end and at the end of the path p_c. An example of such q function that translates 3 dB BW to OSNR penalty, according to the channel modulation format is shown in **Figure. 4.9(b)**, again obtained through VPI simulations. Note that a standard PLM could do the above in one step, go directly from subpaths to filtering penalties, without calculating the attributes, but the above description gives us the intermediate step that helps us to model the uncertainties. The filtering penalty assuming identical filters is based on some predefined function/measurement (such as in **Figure. 4.3(b)**) and is unaware of the uncertainties due to the spectral variations of the filters *i.e.* $\Delta^j_m = 0$. This results in filter uncertainty unaware penalty denoted by G_{FU}. This G_{FU} results in the SNR over subpath $p_{c,m}$, at the end of the link m and before the next node as given by

$$SNR_{FU,m}(p_c, \lambda_c) = \left(\frac{G_{o,m}(\lambda_c)}{G_{ase,m} + G_{nli\_RU,m}(\lambda_c)}\right) . G_{FU}(p_{c,m}) \qquad (4.14)$$

To calculate the SNR at the end of connection c with multiple links *i.e.* $SNR_{FU}(p_c, \lambda_c)$ and $SNR_{FA}(p_c, \lambda_c)$, we extend Eq. 4.14 using the inverse SNR linear additive assumption. We denote the Qtool that uses the *filtering uncertainty unaware* (FU) PLM discussed above by Q_{FU}. This tool calculates the SNR at the end of link m, $SNR_{FU,m}(p_c, \lambda_c)$, and then at the end of the path $SNR_{FU}(p_c, \lambda_c)$, as described above. Since it is filter uncertainty unaware (FU) it needs to include in its margin $design\ margin_1$ a part to cover the filter penalty uncertainty error. Note that as in the previous section of EDFA gain ripple modeling (also in the next) we denote by $design\ margin_1$ the margin of the *unaware* Qtool and by $design\ margin_2$ the lower margin of the *aware* Qtool.

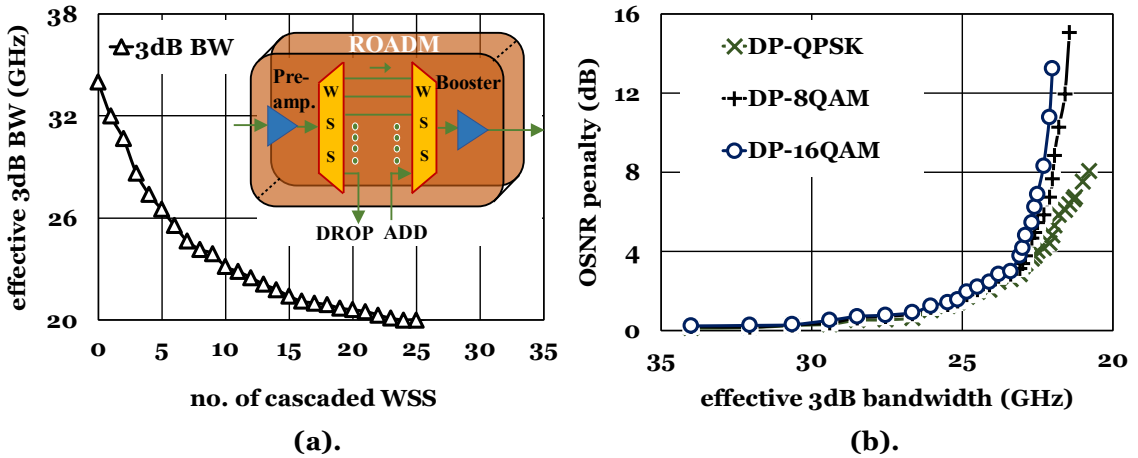


Figure. 4.9. (a) Effective 3dB BW degradation with cascading of WSSs, and (b) OSNR penalty function q (in dB) due to tight optical filtering for DP-QPSK, DP-8QAM, and DP- 16- QAM as a function of 3 dB BW.

However, in real networks there are variations in the spectral responses of filters and (grid) misalignments of transmitter (Tx.) and filters central channels. For example, as shown in **Figure. 4.3(c)**, an uncertainty of $\Delta_m^{3dB} = 10\%$ in 3 dB BW of each filter in a cascade of five filters resulted in ~2.1GHz uncertainty in the signal 3 dB BW, which according to **Figure. 4.9(b)**, results in ~0.6 dB in OSNR penalty estimation. In a real network, $\Delta_m^j \neq 0$ results in a different filtering uncertainty aware penalty, G_{FA}, and hence inaccurate QoT estimation. Assuming known Δ_m^j for all filters, and thus known subpath penalties $G_{FA}(p_{c,m})$ the SNR at the end of the link m before the next node is given by

$$SNR_{FA,m}(p_c, \lambda_c) = \left(\frac{G_{o,m}(\lambda_c)}{G_{ase,m} + G_{nli\_RA,m}(\lambda_c)}\right) . G_{FA}(p_{c,m}) \tag{4.15}$$

If we know the filtering uncertainties $\Delta_{p_{c,m}}^j$ we can calculate the SNR at the end of each link and at the end of the path $SNR_{FA}(p_c, \lambda_c)$ taking inversely additive assumption into account (similar to Eq. (4.9)). We denote the Qtool that uses this filtering uncertainties aware (FA) PLM as Q_{FA}. Such Qtool will use $design\ margin_2$ on top of $SNR_{FA}(p_c, \lambda_c)$ to cover other estimation errors, but not of filters. As discussed, it stands to reason that filter uncertainty aware Qtool, Q_{FA}, has a lower design margin *i.e.* $design\ margin_2 < design\ margin_1$.

4.4.2 ML-based ROADM Filters Uncertainty Modeling

The standard filter uncertainty unaware Qtool Q_{FU} starts from the transmitter parameters and calculates the cascaded attributes for identical filters down the path until the receiver. So, as discussed in the last subsection, Q_{FU} implements the attribute dependent function w^j which calculates the attribute of the next link based on the attribute of the previous link, $s_{p_{c,m+1}}^j = w^j(s_{p_{c,m}}^j, F_m)$ for a specific filter F_m assuming no uncertainty $\Delta_{p_{c,m}}^j = 0$ (**Figure. 4.9(a)** shows a w^{3dB} function). Based on monitoring information we understand the actual filtering uncertainties and use that for estimation of future connections requests as described in the following.

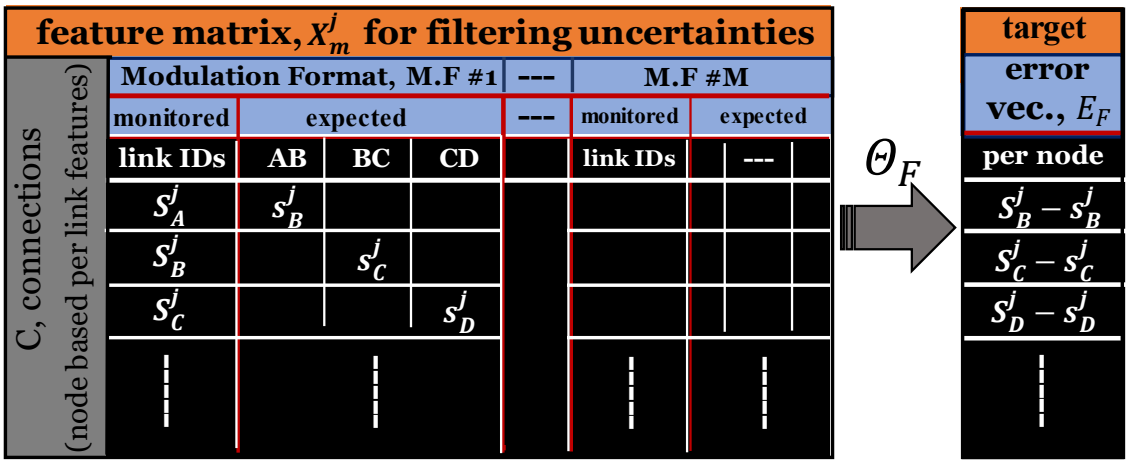


Figure. 4.10. Link based generated filters attributes features matrix X_F and target error vector E_F for the sample network shown in Figure. 4.5 (4 nodes, with established connection from A to D).

Table. 4.1 - Algorithm for link by link ML correction on testing dataset

1:	**Input:** New connection request $\boldsymbol{r \notin C}$ Filters attributes feature matrix extraction $\boldsymbol{f_{FA}}$ Trained filter uncertainty ML model $\boldsymbol{\Theta_F}$
2:	Run routing and spectrum assignment (RSA) algorithm to assign wavelength λ_r, path p_r and other transmission parameters, e.g. filters 3 dB BW Assume known filters shape F_m for all m in path p_r
3:	Assign transmitter j-th attribute to $S^j_{p_{r,0}}$ and $s^j_{p_{r,0}}$
4:	**for** $m = 0$: $N-1$ (N is the number of links on path p_r)
5:	$s^j_{p_{r,m+1}} = w^j(S^j_{p_{r,m}}, F_m)$
6:	calculate $X^j_{p_{r,m}} = f_{FA}(S^j_{p_{r,m}}, s^j_{p_{r,m}})$
7:	use trained ML model to estimate the correction, $E^j_{p_{r,m+1}} = \Theta_F(X^j_{p_{r,m}})$
8:	apply correction, $s^j_{p_{r,m+1}} = s^j_{p_{r,m+1}} + E^j_{p_{r,m+1}}$
9:	**end** for
10:	Calculate SNR estimation error at receiver with q using corrected estimations at receiver $s^j_{p_r}$

We assume that OCMs are available at links ends to monitor the signal spectrum $S_{p_{c,m}}(p_c, \lambda_c)$ for all established connections c in C. As discussed in the last section, from $S_{p_{c,m}}$ we can extract appropriate attributes $S^j_{p_{c,m}}$, which serve as the ground truth and are stored in Qtool database. Based on this OCM information $S^j_{p_{c,m}}$, we correct Q_{FU} as follows. We denote the expected attributes from the monitored data of established connection C by $s^j_{p_{c,m+1}} = w^j(S^j_{p_{c,m}}, F_m)$. The monitored-expected error is given by $E^j_{p_{c,m+1}} = S^j_{p_{c,m+1}} - s^j_{p_{c,m+1}}$, which is due to unknown uncertainties $\Delta^j_{p_{c,m+1}}$. Then, we extract a per link ML features matrix, $X^j_{p_{c,m}} = f_{FA}(S^j_{p_{c,m}}, s^j_{p_{c,m}})$. By concatenating all errors and features matrices for the different attributes j we obtain the overall error and the filters attributes feature matrix, E_F and X_F, respectively. Our goal is to identify a per link dependent (each link representing the WSS before and after it) error function Θ_F which maps the features matrix X_F to the error E_F, that is $\Theta_F(X_F) \approx E_F$. We rely on supervised ML regression techniques for training on E_F as target/label vector and finding Θ_F. **Figure. 4.10**. shows the features matrix X utilizing OCM data $S^j_{p_{c,m}}$, and the w^j expected attributes $s^j_{p_{c,m+1}}$ for the toy network of **Figure. 4.5**.

Assuming a new connection request $r \notin C$, we start with its transmission spectrum parameters to calculate the expected attributes for the next link using w^j, extract ML features for known modulation format, use the trained ML to predict the attributes correction (new E_F), apply that correction, and repeat that link by link down the path until destination. Then we translate the corrected expected filter attributes at the end of

the path to the SNR penalty using the function q, as discussed in the last subsection (**Figure. 4.9(b)**). The pseudo-code of this process is given in algorithm. The testing error is identified once we establish the connection, measure the SNR at the receiver and compare it to SNR estimated by algorithm given in **Table. 4.1**.

4.5 Combined - EDFA Gain Ripple and Filter Spectral Uncertainties

Lastly, in this section we highlight the most realistic case which merges both effects. We here provide the mathematical modeling and ML scheme to estimate the penalties for these two effects jointly and achieving a more accurate QoT estimation.

4.5.1 Mathematical Modeling

We consider a standard PLM which assumes EDFA with no ripples and no filtering penalty uncertainties. For connection, c, the filter penalty generated by identical filters for subpath $p_{c,m}$ is $G_{FU}(p_{c,m})$ and the PSD of NLI noise by flat ripple EDFAs is $G_{RU}(\lambda_c)$. The overall accumulated noise, G_{Noise_RFU} and the equivalent SNR calculated at link m end is given by

$$SNR_{RFU,m}(p_c,\lambda_c) = \left(\frac{G_{o,m}(\lambda_c)}{G_{Noise_{RFU}}(\lambda_c)}\right).G_{FU}(p_{c,m}) \tag{4.16}$$

where

$$G_{Noise\_RFU,m}(\lambda_c) = G_{ase,m} + G_{nli\_RU,m}(\lambda_c) \tag{4.17}$$

We can then calculate the total SNR of the connection (with inverse addition assumption) as $SNR_{RFU}(p_c,\lambda_c)$, and we add the $design\ margin_1$ to account for inaccurate model of ripple, filters uncertainties, and other factors. This PLM/Qtool is a combination of Q_{RU} and Q_{FU}. So, it is *ripple plus filtering uncertainties unaware (RFU)* and is denoted by Q_{RFU}.

Assuming that we know the gain ripple profiles $g_{n_s}(\lambda_c)$ for all EDFAs and also the uncertainties in the filter responses $\Delta^j_{p_{c,m}} \neq 0$ for all filters. The penalty generated with known non-identical filter responses, $\Delta^j_i \neq 0$ for subpath $p_{c,m}$ is $G_{FA}(p_{c,m})$ and the PSD of NLI noise generated by EDFAs with known gain ripples, is $G_{nli\_RA}(\lambda_c)$. The overall accumulated noise, $G_{Noise\_RFA,m}$ and SNR calculated at the end of link m is given by

$$SNR_{RFA,m}(p_c,\lambda_c) = \left(\frac{G_{o,m}(\lambda_c)}{G_{Noise\_RFA,m}(\lambda_c)}\right).G_{FA}(p_{c,m}) \tag{4.18}$$

where

$$G_{Noise\_RFA,m}(\lambda_c) = G_{ase,m} + G_{nli\_RA,m}(\lambda_c) + G_{RA,m}(\lambda_c) \tag{4.19}$$

Using the inverse SNR linear additive assumption (similar to Eq. (4.9)) we can calculate the total SNR at the end of the path p_c, $SNR_{RFA}(p_c,\lambda_c)$. Note that based on the two modeling schemes the previous sections for EDFA gain ripple, the above Eq. (4.19) can be formulated from any of those two schemes. This Qtool can be modeled in two different

ways depending upon the schemes. In general, we call this as a *ripple and filtering aware (RFA)* Qtool and denote it as Q_{RFA}. For any of the two schemes for ripple modeling inside such Q_{RFA}, it would use $design\ margin_2$ on top of $SNR_{RFA}(p_c, \lambda_c)$ to account for uncertainties other than the two under consideration.

4.5.2 ML-based Gain Ripple and Filter Uncertainty Modeling

In this case, when both effects are simultaneously present, we use all available monitors, OCM, DGE and coherent receiver (at destination node) information, together. We start by implementing the ML-based filter uncertainty model described in Section 4.4. We use OCM monitored spectra, to obtain filter attributes, $S^j_{p_{c,m}}$. Then we use ripple and filter unaware Qtool, Q_{RFU} and its functions w^j to estimate the filtering attributes for next links and create filters attributes features matrix X_F (**Figure. 4.10**). We then train the ML model Θ_F. Then, we use the OSNR filter penalty estimation function q to calculate from the monitored attributes at the end of the connections $S^j_{p_c}$ the related filter SNR penalties, denoted by $P_{FA}(C)$, which are assumed to be quite accurate since they come from the monitored data. We then used monitored SNR at the receivers (SNR_{RFA}) and subtract from those the filter penalties $P_{FA}(C)$ to obtain the SNR that includes only the gain ripple effect (SNR_{RA}). From that we obtain the accumulated noises $Y_{RA}(C)$ that include the ripple penalties. Next, we use the DGE monitored data and the process described in Section 4.2 to create the ripple features matrix X_{R_S1} or X_{R_S2} and train the gain ripple ML model Θ_{R_S1} or Θ_{R_S2}. Note that this stepwise approach works because adequate monitoring information is handled to distinguish between the two effects: OCMs provides information to understand the filter penalty which can be then removed from the monitored SNR at the receivers and focus afterwards on the gain ripple penalties. For new connection requests $r \notin C$, it is followed the same sequence, and apply first the filter penalty correction based on Θ_F and then the ripple penalty correction Θ_{R_S1} or Θ_{R_S2}.

4.6 Results and Discussions

To quantify the benefits of the developed accurate QoT estimators, we performed simulations to identify the amount of margin reduction in the presence of single or both gain ripple and filter penalty uncertainties. For this analysis, we consider the Deutsche Telekom, DT topology formed by 12 nodes and 40 bidirectional links. The link lengths range from 48 to 458 km as shown in **Figure. 4.11**.

We assumed uncompensated bidirectional fiber links with spans of 80 km of standard single-mode fiber (SSMF). We assumed 4 different traffic loads of {100, 200, 300, 400} total connections with uniformly chosen source-destination pairs. We served each demand with one wavelength, assumed to be modulated at 32 Gbaud with a modulation-tunable polarization-multiplexed transponder. We assumed that the transponder supports {QPSK, 8-QAM, 16-QAM} modulation formats attaining data rates of {100, 150, 200} Gb/s, respectively. We assumed a frequency slot size of 12.5 GHz and allocated 3 spectrum slots for each 32 Gbaud connection. We assumed a stable network state, where a specific set of connections are already established and a new set of connections need to be set up. As discussed above, supervised ML approach is used to

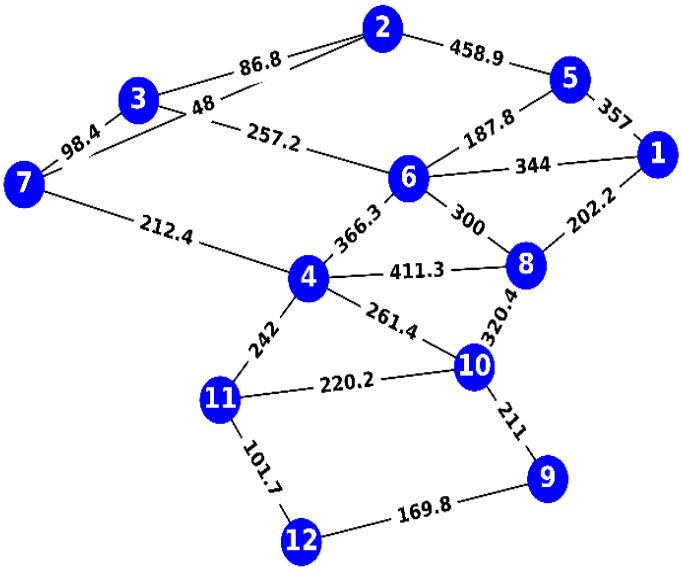


Figure. 4.11. Deutsche Telekom, DT-12 node network topology with length (in km).

train the respective ML models on the monitored connections to understand the gain ripple and filter uncertainty penalties. This trained ML model is then adopted to estimate the *unseen* penalties of the new connection requests with higher accuracy/lower design margin.

4.6.1 Only EDFA Gain Ripple

To study the effect of the gain ripple, initially it is assumed no filtering uncertainty. We assigned experimentally measured gain ripple profiles, $g(\lambda)$, to each span EDFA. To generate separate profile for each span amplifier, we applied random wavelength shifting and amplitude scaling to 4 experimentally captured ripple profiles (as discussed in Section 4.2). In total, we created around 120 different EDFA gain ripple profiles. We assumed that OCMs are installed before each ROADM node and that we can also monitor the power profiles applied by the DGEs through their feedbacks. All these were integrated in the ripple aware Qtool, Q_{RA}, that calculated the DGE power profiles and also the total noises at the receivers (Rx.) $Y_{\mathrm{RA}}(\mathrm{C})$. Taking as reference the ripple unaware Qtool Q_{RA} (see Section 4.2), **Figure. 4.12(a)** depicts the estimation error for 400 connections, which pertains to the ripple penalty. The penalties were distributed in positive and negative sides depending upon the ripple values and were ~1.8 dB in total. Positive/negative penalties result in upper/lower bounds for the design margin, which we call as "high/low margin". Typically, ~2-3 dB of design margin is used to accommodate all the uncertainties [8]. **Figure. 4.12(a)** shows that ~1 dB of QoT tool design margin would be required to accommodate the ripple penalties only (shown by histogram plot in dotted red circle). The remaining part of the design margin would cover the other uncertain effects. To improve the estimation accuracy, we used the ML model based on two schemes as described in Section 4.2. For the *Scheme-1*, we used the per link and end to end monitored SNR information from the $\tilde{Q}_{RA}$ to obtain the noise vector $\tilde{Y}(C)$. By subtracting $\tilde{Y}$ and noise monitored from ripple unaware Qtool Q_{RU}, it is

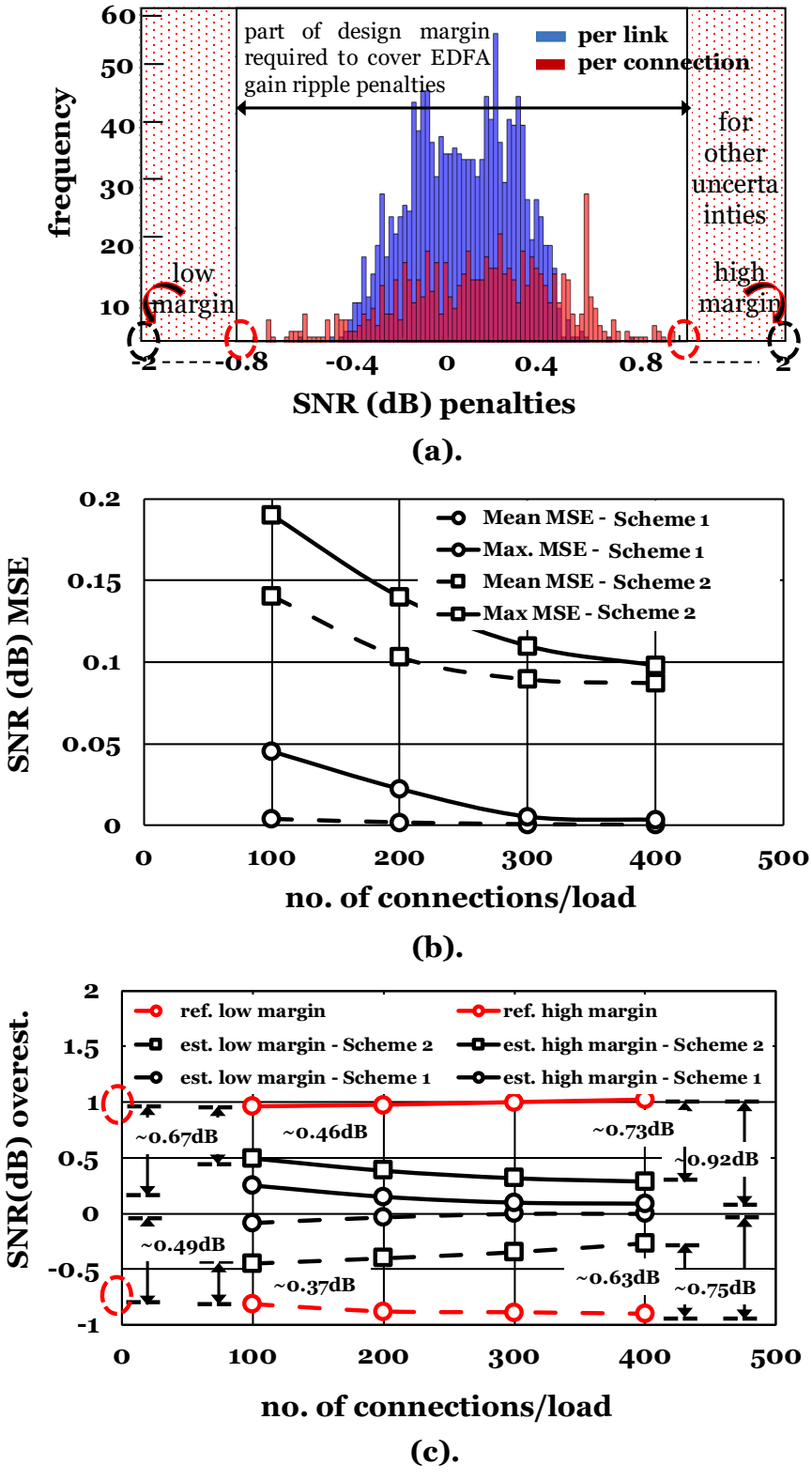


Figure. 4.12. Effect of ripple assuming no filtering uncertainty (a) penalty distribution for 400 connections, indicating min. required design margin to accommodate ripple, (b) performance evaluation (MSE of SNR (dB)) of trained ML model on testing dataset, and (c) maximum overestimation error as a function of load.

obtained the penalty vector E_{R_S1}. We then created the ripple feature matrix X_{R_S1} and evaluated several ML assisted regression techniques to fit $\Theta_{R\_S1}(X_{R\_S1})$ on E_{R_S1}.

For the *Scheme-2*, we used the DGE power profiles with the ripple-dynamic gain equalizer aware Q_{REA} Qtool to obtain the noise at the receiver, $Y_{REA}(C)$. By subtracting Y_{REA} and Y_{RA}, we obtain the penalty vector E_{R_S2}. We then created the ripple feature matrix X_{R_S2} and evaluated several ML assisted regression techniques to fit $\Theta_{R\_S2}(X_{R\_S2})$ on E_{R_S2}. For both the schemes, several ML based regression models are attempted such as linear fitting, quadratic, polynomial fitting, neural network, SVMR etc. For the *Scheme-1*, in the presented results the polynomial regressions of degree 4 is used that achieved a maximum MSE of 4.5E-2 on predicted SNR with load of 100 connections as shown in **Figure. 4.12(b)**. In the ML based regression models, MSE and root MSE (RMSE) are the key performance criteria to evaluate the estimation accuracy. With the increase in load from 100 to 400, the maximum MSE converge to a value of ~4E- 3.

However, in case of Scheme-2, we used SVMR with linear kernel function that achieved maximum MSE of ~0.19 on predicted SNR with load of 100 connections as shown in **Figure. 4.12(b)**. By increasing the load from 100 to 400, the maximum MSE converges to a value of ~0.096 in case of Scheme 2. The presented results presented for the different scenarios are averaged over 200 iterations at each load. For the above set of simulations, the maximum used peak-to-peak ripple intensity among all the span EDFAs was about ±0.5 dB, which resulted in a reference margin ($design\ margin_1$) of ~1.02 dB (maximum of dotted red circle of histogram). **Figure. 4.12(c)** shows the maximum overestimation error on the SNR, relative to **Figure. 4.12(a)** for both the schemes. This overestimation is the reduced estimated high and low margin ($design\ margin_2$). In case of *Scheme-1*, for high margin, it is found to be 0.08 dB, yielding a ~0.92 dB margin reduction with 400 connections. For low margin, this reduction value is ~0.75 dB in case of *Scheme-1*, as the distribution of penalties is less for low margin side. Similarly, in case of *Scheme- 2*, for high margin, it is found to be ~0.28 dB, yielding a ~0.73 dB margin reduction at a load of 400 connections. For low margin, we found ~0.63 dB reduction with Scheme-2. For comparison purposes we plot results from both the schemes in **Figure. 4.12(c)**.

Scheme-1 assumes electrical SNR monitors available at each ROADM node, which is indeed a strong assumption. With Scheme-1, we obtained greater than 90% margin reduction on both sides. However, it is worth highlighting that the more accuracy improvement form Scheme-1 comes with the cost of more expensive channel monitors (electrical) at each node of the network. On the other hand, Scheme-2 is based on cheap OCMs monitoring information at intermediate nodes and only requires electrical SNR information at the destination node, where the receiver is available anyways. In short, modeling is feasible form both the schemes. However, the accuracy of the models is compromised in terms of both the quantity and the type of the monitoring information.

4.6.2 Only Filtering Penalties at ROADM Nodes

To study the effect of filter spectral shape uncertainties, we assumed flat EDFAs and we randomly applied small uncertainties Δ_m at each WSS. The maximum resulted end to end 3 dB BW variation with small Δ_m was found to be ±1.5 GHz. Such uncertainties would reflect Tx. and filters-grid mismatches, and small variations in filters shapes (at

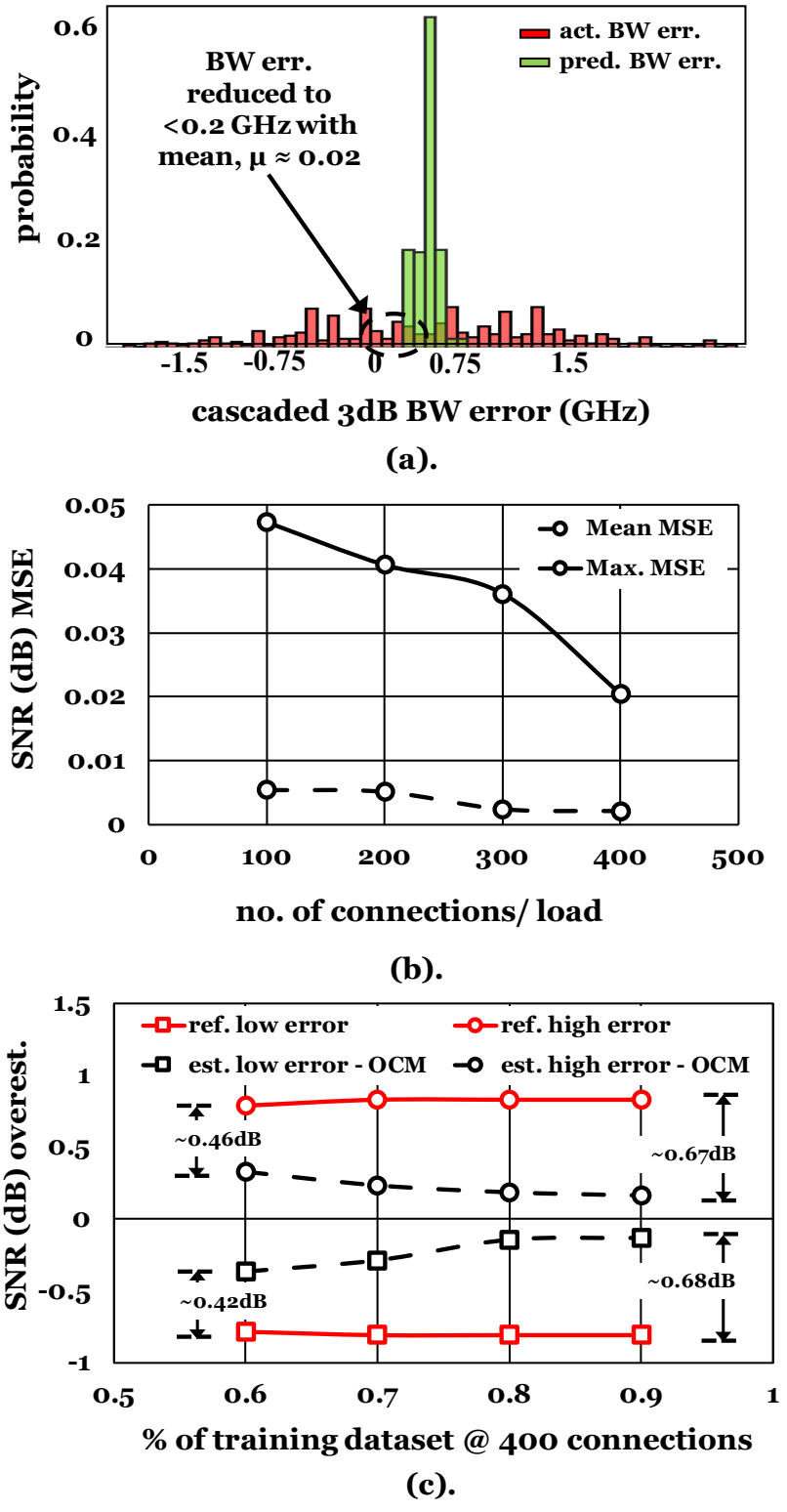


Figure. 4.13. Effect of filtering uncertainty assuming no gain ripples (a) increase in 3 dB BW prediction accuracy with ML along with error distribution (without ML), (b) performance evaluation (MSE of SNR in dB) of trained ML model on testing dataset, and (c) maximum overestimation error as a function of load.

ROADM nodes). We used the filter uncertainty aware Qtool, Q_{FA}, to calculate the ground truth, the 3 dB BW of the connections on their paths, along with their SNR filter penalties. Then, for the set of established connection C, we used the filter uncertainty unaware Qtool, Q_{FU}, to estimate the 3 dB BW and SNR filter penalties, and to calculate the errors with respect to the ground truth. **Figure. 4.13(a)** shows the distribution of the 3 dB BW error at the load of 400 connections which has both positive and negative sides depending upon Δ_m. The corresponding SNR errors are also distributed in both polarities resulting in reference high and low margins. We then used the proposed ML based method discussed in Section 4. We extracted the filters attributes features matrix, X_F and used the calculated attributes (only 3 dB BW here) errors, E_F, to train a SVMR model with Gaussian kernel. Then for a new connection the algorithm described in **Table. 4.1** is adopted, in a link by link estimation of 3 dB BW down to the receiver. This was then used to estimate the filtering penalty which finally outcomes the estimated SNR of the connection.

Figure. 4.13.(a) shows the achieved error reduction in 3 dB BW that is from ±1.5 GHz → ~0.18 GHz, using the trained SVMR model. **Figure. 4.13(b)** shows a maximum MSE of ~0.04 dB on estimated SNR at a maximum load of 400 connections using the trained SVMR model. **Figure. 4.13(c)** reflects the SNR accuracy/margin reduction as a function of the load. We observed a maximum error reduction/accuracy improvement of ~0.67 dB for high margin and ~0.68 dB for low margin, respectively. For high/low margin, it is found an overall reduction of 80.4/83.4% at a load of 400 connections as indicated in **Figure. 4.13(c)**.

4.6.3 Combined – EDFA Gain Ripple and Filter Uncertainty

For the most realistic case of both effects present together in the network, we assigned ripple profiles (perturbating the experimentally collected profiles) at each EDFA and applied small 3 dB BW uncertainties Δ_m^{3dB} at each WSS. Then we used the ripple and filtering uncertainties aware Qtool Q_{RFA} as described in Section 4 to generate the monitoring data. The SNR error distribution for 400 connections is obtained and shown in **Figure. 4.14(a)**. Note that since both effects are considered, we ended up with a wider error distribution than the individual cases discussed in the previous two subsections.

To improve the estimation accuracy, we followed the process described in Section 4.4 to train the related ML models. First it is accounted for the filtering penalties; using the monitored spectra we trained SVMR ML model Θ_F. Then the filtering penalties are removed from monitored SNR at the receivers to focus on the ripple. Note that for EDFA gain ripple penalty estimation, we tried trained models from both the previously discussed schemes, that is Θ_{R_S1} and Θ_{R_S2}. **Figure. 4.14(b).** shows the obtained SNR MSE at varying load values. It can be clearly seen that with more load the models are trained better ultimately resulting in low MSE value. **Figure. 4.14(c)** shows the reduction in the reference high and lower margins. In case of scheme-1, the overall related margin savings is found to be ~1.2 dB and ~1.3 dB for the case of high and low margin, respectively. In case of scheme-2, the overall related margin savings is found to be ~1.08 dB and ~1.1 dB for the case of high and low margin, respectively. These slight differences in Qtool estimation accuracy are generated from the type and quantity of the monitoring information as already discussed in last sections.

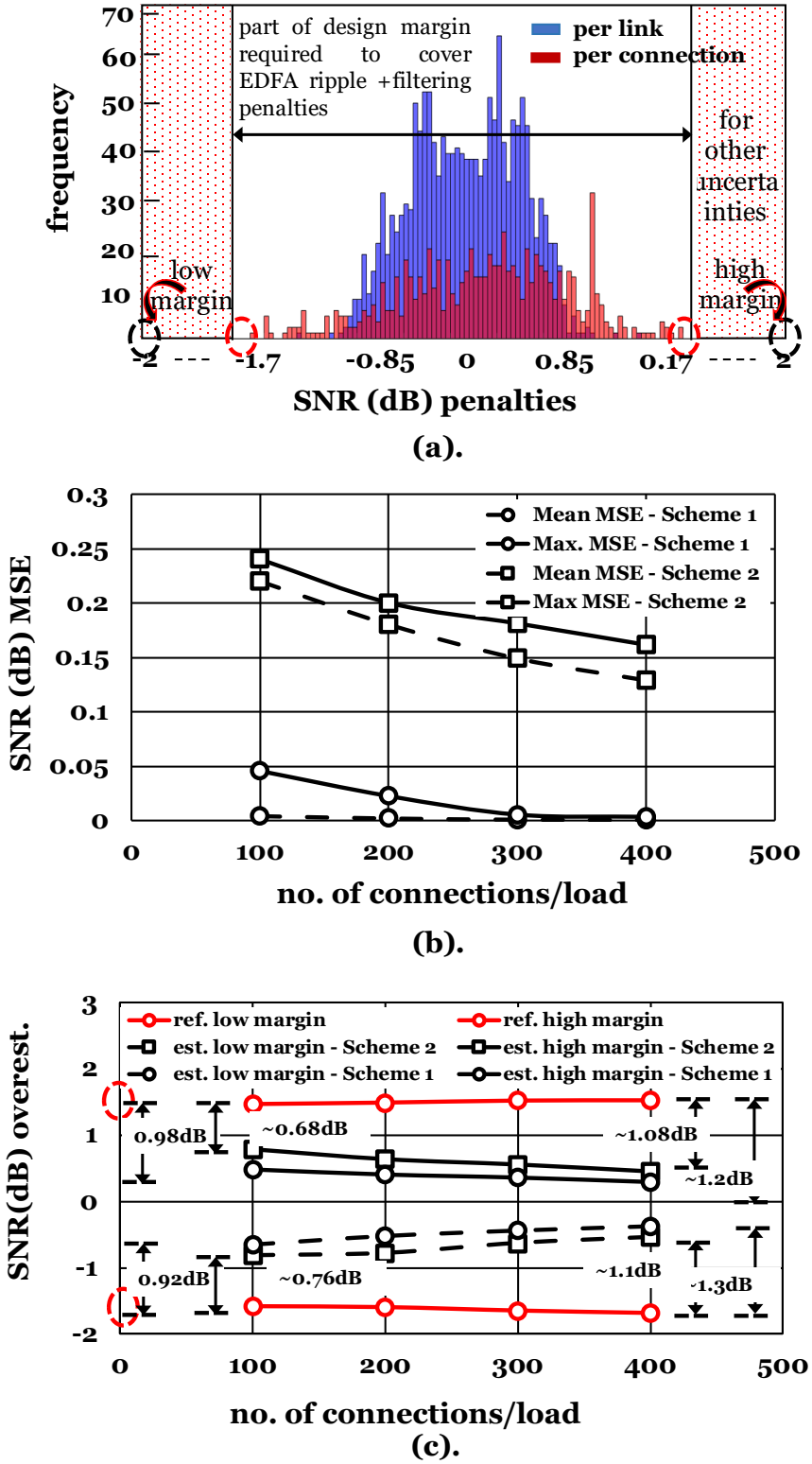


Figure. 4.14. Effects of both ripple with node uncertainties (a) penalty distribution for 400 connections, indicating min. required design margin to accommodate ripple with node uncertainties, (b) performance evaluation (MSE of SNR in dB) of trained ML model on testing dataset, and (c) maximum overestimation error as a function of load.

4.6.4 Effect of Intensity on Ripple and Filter Uncertainties

Finally, we extended our simulations to verify that the proposed ML based solution works for different intensities of ripple and filtering uncertainties and to quantify the related benefits. At first, the gain ripple intensity is varied, assuming only ripple with no filter uncertainty effect. We divided the span EDFA gain ripple profiles by a factor of 1 to 4, resulting in peak-to-peak fluctuations of ±0.5 dB to ±0.125 dB.

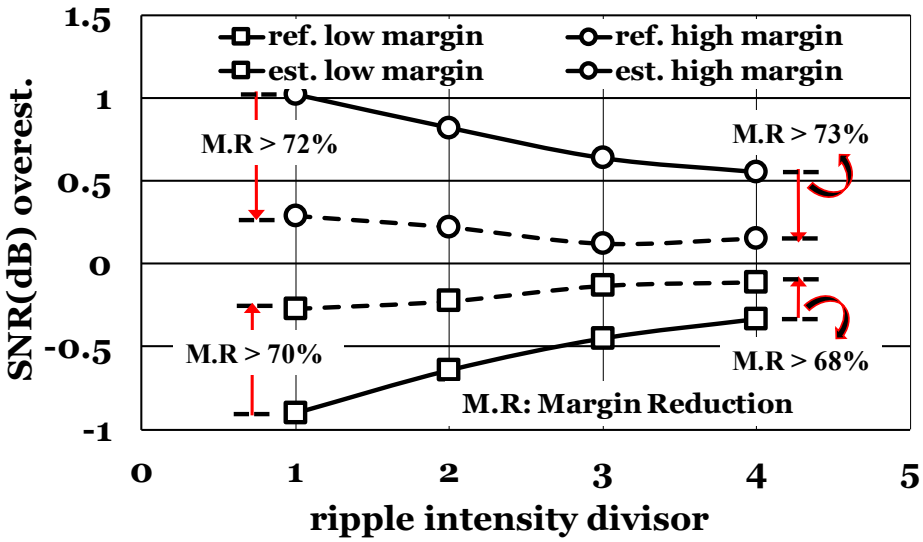


Figure. 4.15. **New margins for ripple with no node uncertainty, with different intensities of peak-to-peak gain ripple (reference as ±0.5 dB).**

We estimated the high and low margins at a fixed load of 400 connections as shown in **Figure. 4.15**. The obtained savings were higher than 70% for the examined peak-to-peak values. Note that, we tried trained models on both schemes that is Θ_{R_S1} and Θ_{R_S2}. Since the model Θ_{R_S1} is already proved to be a near to ideal Qtool from results in **Figure. 4.12**. In the presented results we only show the results obtained from the trained model (Θ_{R_S2}) with scheme-2.

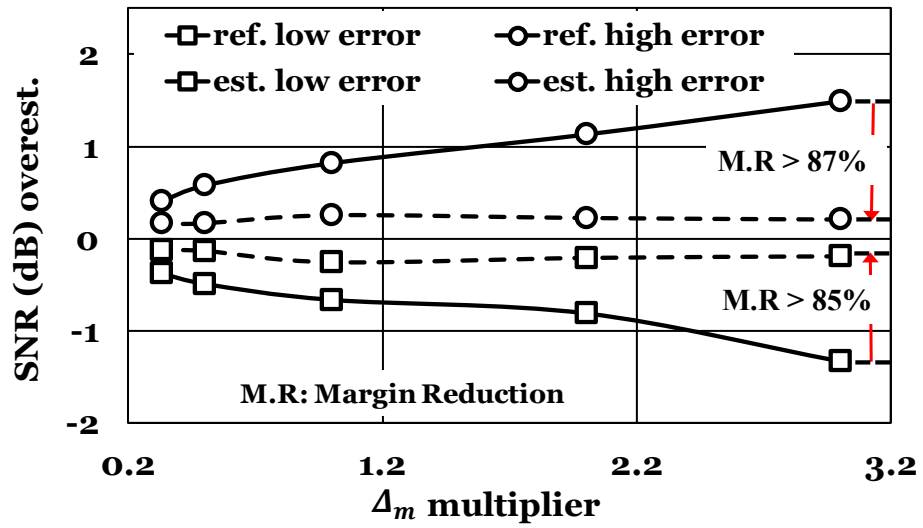


Figure. 4.16. **New margins for filter with no ripple uncertainty having reference Δ_m = ±10%.**

We then varied the filter attributes uncertainty Δ_m, assuming only filtering uncertainties. We multiplied Δ_m by a factor of 1/3 to 3 and estimated high and low margins/errors at a fixed load of 400 connections. Note that high value of Δ_m reflect ROADMs nodes with higher uncertainty, which are expected in disaggregated/ multi-vendor networks. As expected, higher reference margins are required in this scenario, and our accurate modeling results in more pronounced savings that reach > 85% and > 1.5 dB on both high and low margins as shown in **Figure. 4.16**.

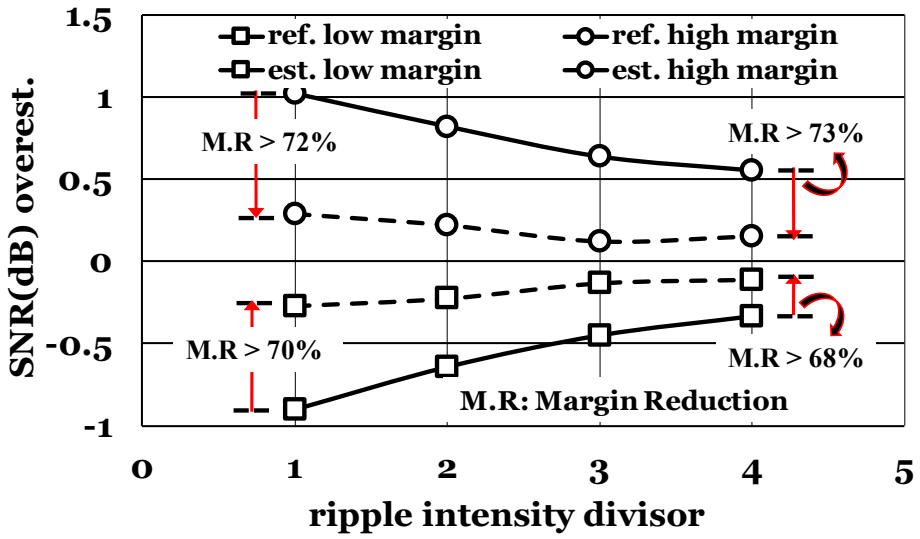


Figure. 4.17. New reduced margin for different intensities of peak to peak gain ripple with fixed uncertainties, Δ_m = ±10% inside ROADM node.

Finally, when both uncertainties are present, we again varied peak-to-peak EDFA gain ripple intensity (similar to **Figure. 4.15**) but for that it is assumed a fixed range of node uncertainties (Δ_m= ±10%.). Both schemes are tried but the presented results are only for scheme- 2 (Θ_{R_S2}). In **Figure. 4.17**, we show the reduction of the related margin at a load of 400 connections is achieved. For the examined intensities of the uncertainties, an overall margin reduction of > 75% on both high and low margins.

4.7 Conclusions

In this chapter, we have presented ML based schemes to improve Qtool estimation accuracy, specifically focusing on the EDFA gain ripple and the filtering penalty effects.

We proposed to use available monitored information from established connections and use them as a feedback to train ML model. Based on both the type and the quantity of the monitoring information, we proposed appropriate supervised ML to model the EDFA gain ripple and filtering penalties. To this end, we initially developed independent models for the two effects. Then, a joint model for both effects is adopted. The ML model is used for estimating the penalties when accommodating new connection requests, improving the Qtool estimation accuracy and thus reducing the required design margin. With combined span EDFA gain ripples and ROADM node uncertainties, it is accomplished a design margin reduction of 1.68 dB to 0.37 dB for new connection requests with respect to the reference ripple and filtering uncertainty unaware Qtool.

An accurate and fast Qtool is required for almost every optimization task of an optical network. The next chapter rely on these developed more accurate ML based QoT estimation tool. However, the focus of the upcoming Chapters is on the integration of this Qtool with optimization tasks such as transponders launch power optimization.

Chapter 5

Qtool: Extensions and Performance Evaluation

The concept of disaggregation of optical transport systems is considered by several operators as a means of higher flexibility and cost reduction. In this chapter, we propose PLM extensions that capture the performance variations of multi-vendor TPs in partial disaggregated optical networks, that is objective O.2. In particular, we propose four TP vendor dependent performance factors and we also devise a ML scheme to identify these performance factors in offline and online network planning scenarios. The proposed PLM can then be used as an extended/improved QoT estimation tool or Qtool and its performance can be evaluated with use cases of interest.

We evaluated the performance of the proposed extended Qtool on two use cases, *(i)* O2.1 - TPs launch power optimization in planning/static phase, and *(ii)* O2.2 - design margin reduction for incremental planning. In incremental planning scenario, we also show additional SNR savings that can be attained for new connection requests by proper selection of the TPs available from multiple vendors with difference performance factors.

5.1 Introduction and Traditional Qtool Limitations

The datacenter architecture is used to translate the concept of disaggregation (presented in Chapter 2) into optical networking. Interchangeable, highly flexible computing and network nodes form the foundation of datacenters [10]. This flexible datacenter architecture approach, pushes optical networking to explore multi-vendor disaggregating hardware and software with a strong focus on interoperability. Substantial effort has been placed in developing vendor-neutral software controllers, third party network orchestrators, and northbound/ southbound interfaces. Moreover, some works demonstrated line systems that are open to multiple vendor network elements [10], [101]. Such developments have an ultimate objective to enable an open or disaggregated line system (OLS), where the optical hardware from multiple vendors can be interconnected and controlled centrally through a common control plane. Thus disaggregated optical transport system is considered as a means to accomplish higher flexibility and cost reduction. We already presented different levels of disaggregation, from partial to full disaggregation in Chapter 2. Partial disaggregation, where the line system is open (open line system OLS) to multiple vendor transponders (TPs) has gained the significant attention due to the ease in the control plane implementation. While the

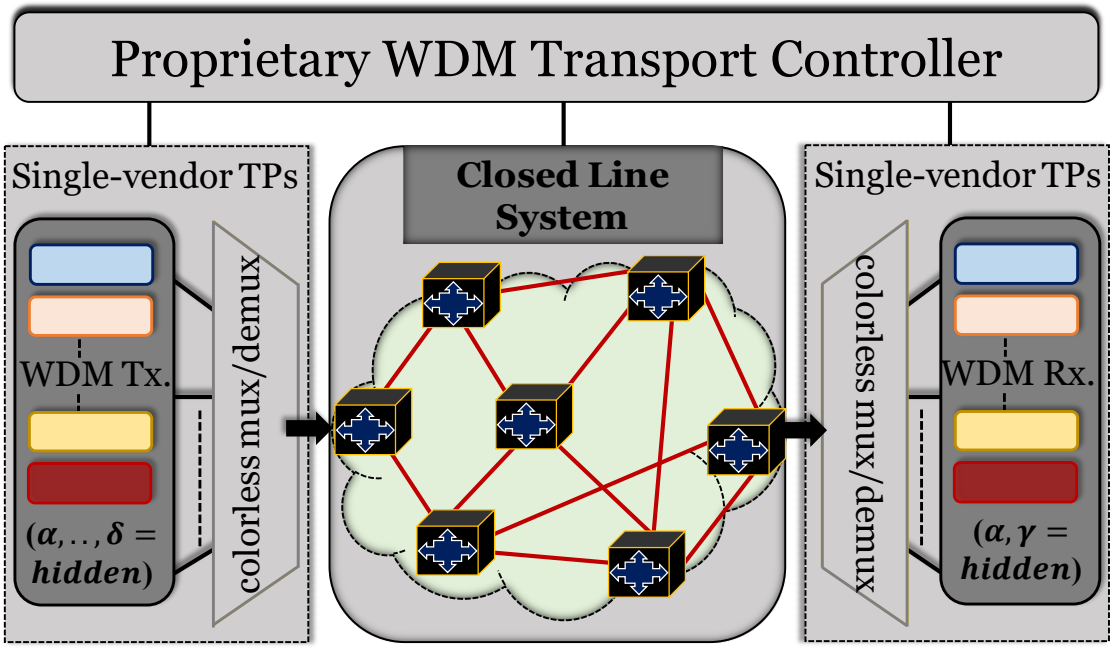


Figure. 5.1. Traditional WDM transport system: line system and TPs with proprietary controller.

introduction of disaggregated optical platforms is expected to push for equipment commoditization and generate new business models, there are still some uncertainties regarding the performance of such systems and their applicability to backbone/core networks that have stringent optical performance requirements. One of the most basic requirement for such networks is to have a PLM/Qtool that accounts for vendor dependent performance factors of network elements (e.g. multi-vendor TPs) rather than relying on traditional closed source vendor Qtool estimator and planning tools.

In traditional optical transport networks, both the line system and the TPs are aggregated (single vendor) and controlled via a proprietary Network Management System (NMS) or controller as shown in **Figure. 5.1**. For such networks, network elements operating parameters are decided by the network proprietary controller. For instance, the TPs launch power is generally set to a fixed value by the vendor itself that is optimized to achieve the best Signal to Noise Ratio (SNR) for the central channel under high load conditions. While optimizing/deciding that, several factors need to be accounted, including statistical variations of components of the TPs, even if they come from the same vendor [103]. Used margins, which account for impairment calculation uncertainties and ageing, generally cover such TP performance variations [8], [99]. In the targeted OLS scenario, the network infrastructure is expected to comprise of multi-vendor equipment such as TPs which are controlled via well-defined models and control interfaces (e.g., YANG, REST APIs, NETCONF etc.). The reference architecture for such a multi-vendor network is shown in **Figure. 5.2**.

For multi-vendor TPs in an OLS, apart from the aforementioned TP statistical variances, vendor dependent factors play a crucial role in QoT/SNR estimation. To be more specific, in such a network, performance variations arise from the different TP components, digital signal processing (DSP) implementations, forward error correction (FEC) coding

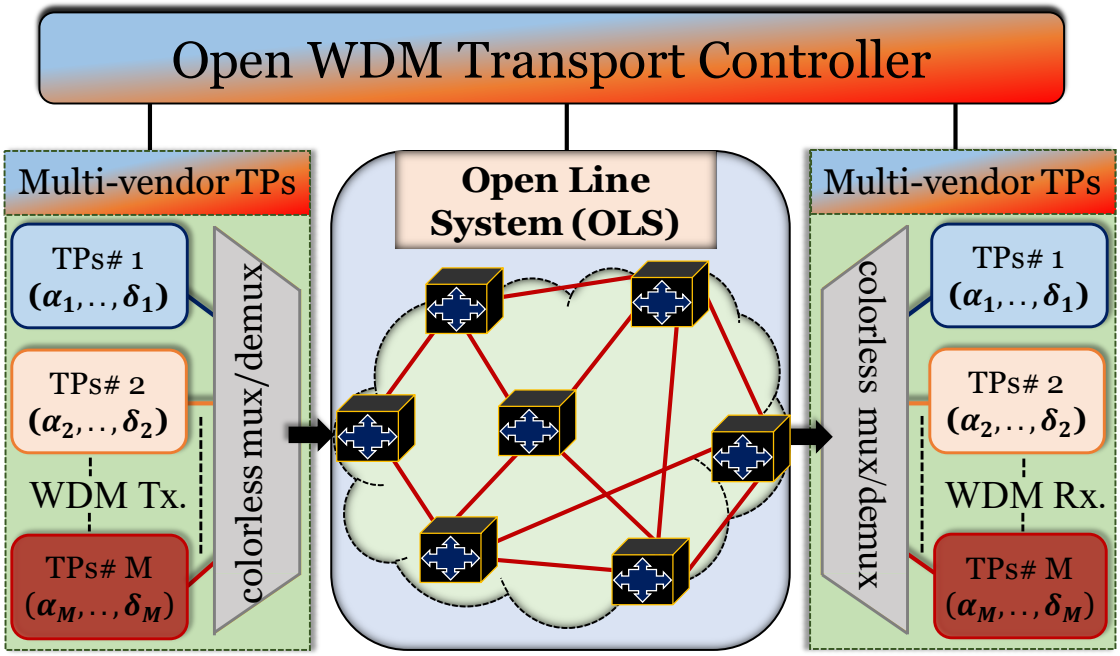


Figure. 5.2. Open Line System (OLS) for multi-vendor TPs with open transport controller.

techniques etc. used by each vendor [102], [103]. Consequently, relying on a typical single vendor PLM based Qtool to estimate the performance of a multi-vendor TP network may result in huge deviations in the estimated and actual QoT values. A solution to mitigate this is to add/increase the used margins on top of PLM estimations. However, adding margins leads to lower efficiency and underutilization of the network.

In light of the above, herein we propose extensions to the PLM to accurately model the physical layer in multi-vendor TP environment that accounts for vendor dependent performance factors. This improved PLM then can be used inside the Qtool with low design margins (compare to the standard Qtool), with overall improved Qtool accuracy. Although in this Chapter, we focus on multiple vendor TPs, the propose model can be extended to cover other line system characteristics such as amplifier ripples, filtering penalties etc. similar to the works presented in the previous Chapter 4.

5.2 Preliminary Study and Motivation

In traditional aggregated optical network with proprietary controller as shown in **Figure. 5.1**, the TPs parameters are only known by the network or domain vendor. In such a network setting, several past works already proved that the quality of transmission (QoT) of connections can be estimated quite accurately using the Gaussian noise (GN) model for fiber nonlinearities [50]. The GN model's main assumption is that, in dispersion uncompensated transmission systems where the nonlinear interference (NLI) caused by the Kerr effect is relatively small, the NLI can be modeled as additive Gaussian noise that is statistically independent of signal and amplified spontaneous emission (ASE) noise. The GN model is a well-accepted PLM for single- and multi-vendor networks [10], [12], [16], [101].

For a generic topology we assume a network with $n = 1,2,..,N$ connections. Let $\boldsymbol{p} = [p_1, p_2 \dots p_N]$ represents the transponders (TP) launch power vector of those N connections. The GN PLM is a model that takes as input several parameters and calculates the SNR values of the connections. Let $\boldsymbol{z}$ represents the *fixed* input parameters of PLM, such as routes, used wavelengths, span lengths, etc. Let $\boldsymbol{r}$ denote the set of GN model fitted parameters: *i)* fiber attenuation coefficients, *ii)* fiber non-linear coefficients, *iii)* fiber dispersion coefficients, and *iv)* a bias. Note that $\boldsymbol{r}$ here exactly represents the set of Qtool parameters that needs to be trained with the LM algorithm discussed in Section 2.9 of Chapter 2. According to the GN model, the impact of optical fiber transmission effects on the generalized signal-to-noise ratio (SNR) of connection $n \in N$ can be modeled as

$$SNR_{n,s}(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{z}) = \frac{G_{O,n,s}}{G_{ASE,n,s} + G_{NLI,n,s}} \tag{5.1}$$

where $G_{O,n,s}$ is the optical signal average power spectral density (PSD), $G_{ASE,n,s}$ is the ASE noise PSD, and $G_{NLI,n,s}$ is the contribution due to NLI noise PSD of connection n generated at span s.

Assuming incoherent noise accumulation over the spans of the path of connection n, we represent $SNR_n(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{z})$ as the generalized SNR of connection n at the path end given by

$$SNR_n(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{z}) = \frac{G_{O,n}}{G_{ASE,n} + G_{NLI,n}} \tag{5.2}$$

Then, the total SNR of connection n, calculated by the traditional single vendor Qtool Q_{SV} (using Eq. (2)) is given by

$$Q_{SV}(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{z}) = [SNR_n(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{z})]_{dB} - SNR_{b2b} - DM_1 \tag{5.3}$$

where SNR_{b2b} is the dB penalty in TP's back to back (b2b) configuration and DM_1 stands for the design margin, which is the additional margin (in dB) added on top of standard PLM calculations to cover modeling inaccuracies such as EDFA gain ripple penalties, TPs performance variations etc. Eq. (5.1) to Eq. (5.3) collectively form the traditional model to estimate generalized SNR. We call it a *single vendor* (SV) Qtool, and refer to it with Q_{SV}. Note that the single vendor Qtool is exactly similar to the standard Qtool (ripple filtering unaware, Q_{RFU} of Chapter 4). However, for the sake of clarity and understanding, we have re-defined it here with different notations.

The above described Q_{SV} considers in a coarse manner the characteristics of the transponder (TP). In general, the PSD of the output signal and NLI noise, that is $\{G_O, G_{NLI}\}$ terms, are affected by receiver characteristics of TPs such as the digital signal processing (DSP) implementation, performance variations of TP components (e.g., laser linewidth, photodiode's responsivity, etc.). The linear noise term, G_{ASE}, is mostly determined by the optical amplifiers. Parts of the TP characteristics and the uncertainties/variations of the TP performance are covered in the design margin DM_1. This can be acceptable for a single vendor TP with small variations.

5.2.1 Motivation

Considering optical networks with line system open to TPs from multiple vendors it stands to reason to expect to have higher variation in TPs performance as compare to a single vendor environment [102]. The DSP chain, which is generally implemented by different vendors in different ways, such as different algorithms or the same algorithms with different parameters (number of digital filter taps etc.), is one of the major sources of variation in multi-vendor TPs performance. Furthermore, different vendors use different components (from different third party vendor etc.) such as balanced photodiodes, local oscillator (LO) lasers (drifts, linewidths), analog amplifiers etc. As discussed, there are statistical variances within the TP components which cause performance (statistical) variations even in single-vendor TPs. However, in OLS scenarios with TPs from multiple vendors, the effects of these variations in overall network performance become more critical.

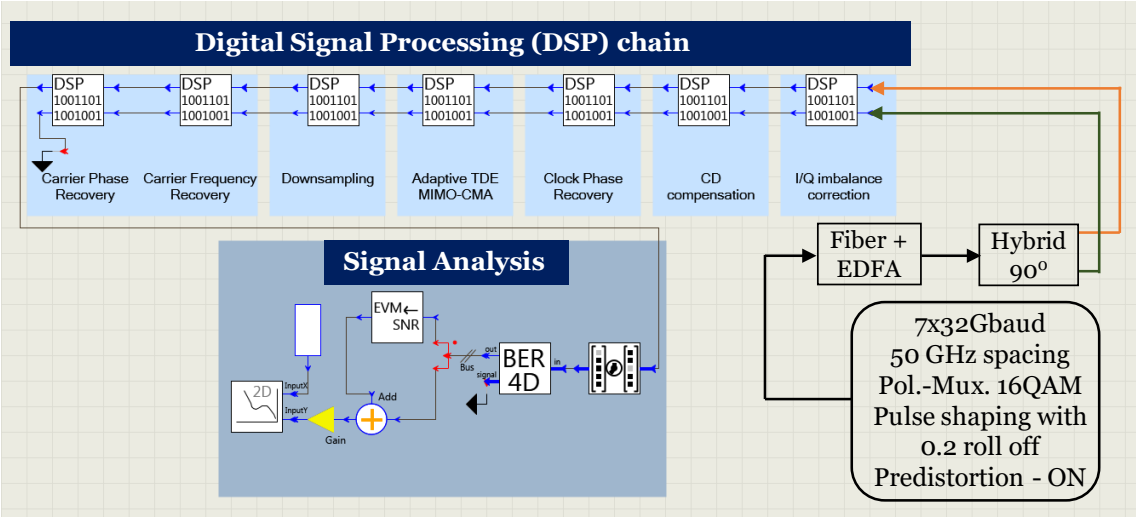


Figure. 5.3. Simulated set up in VPI Transmission Maker in order to emulate different DSP chains for different TP vendors.

Keeping this in mind, we simulated seven polarization-multiplexed 16- Quadrature Amplitude Modulation (QAM) channels, spaced at 50 GHz and modulated at 32 Gbaud symbol rate with root raised cosine (RRC) pulse shaping (roll off-0.2) in VPI Transmission Maker [14]. We used total fiber/link length of 160 km with two identical spans of length 80 km each. The EDFAs (or the line system part) were assumed to be completely flat, in order to capture the performance of the transponders. The total impairments compensated at the receiver DSP block is shown in **Figure. 5.3**. We also varied balanced photodiodes responsivity at coherent receiver front end from 0.8 (worst) to 1.0 (ideal/best) and LO laser linewidth in order to emulate component statistical variation originating from different vendor components. In chromatic dispersion (CD) compensation module, we varied the effective fast Fourier transform (FFT) size and phase noise component. For polarization demultiplexing algorithm we varied the constant modulus algorithm (CMA) and the multi-modulus algorithm (MMA) with different initial taps and number of iterations. We also varied the number of samples during clock phase recovery module, in order to emulate performance variation within different vendors DSP chains. Inside the same module, we also varied the fourth power

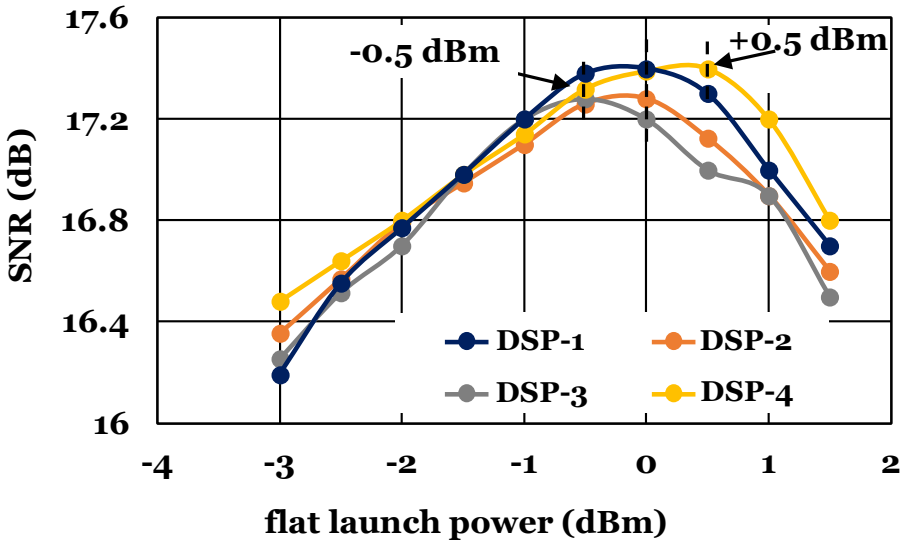


Figure. 5.4. Different value of flat optimized launch power for four different DSP chains representing four different TP vendor.

operation on the received samples, before estimating the frequency offset. Lastly, in the carrier phase recover module, we only varied phase noise parameter from 0 to 20 radians. Based on different combinations of the above DSP algorithms and parameter variations we implemented four different DSP chains at the receiver side (DSP-1 to DSP- 4), in order to emulate TPs from four different vendors.

As expected, due to the maximum nonlinear interference noise from the neighboring channels, the central channel has the worst performance/lowest SNR in a wavelength division multiplexed (WDM) system. As so, we measured the SNR at the central channel while varying the flat or uniform launch power for all channels. From **Figure. 5.4**, it can be seen that depending upon the DSP chain or TPs performance, the best optimized flat launch power is in the range of - 0.5 dBm to +0.5 dBm for the different (vendor) TPs. Note that each vendor if it would be the sole vendor (aggregated network) would perform such optimization, taking into account specificities of the network and operation load, etc., and select the corresponding optimized power. It is also worth noting that while some vendors' launch power (e.g., TP with DSP-4) is optimized, other TPs (DSP-1, DSP- 2 and DSP-3) may be in their nonlinear range.

However, assuming an environment with TPs from multiple vendors, such optimization would not be feasible, and related variations in performance would need to be covered by a corresponding margin as discussed in upcoming results section. So assuming that we use the GN model as the PLM we need to increase DM_1 in Eq. (5.3) to account for the performance differences caused by the TP's different characteristics. Increasing the margin results in underutilization of network capacity as certain TPs deployed in the network have better performance than others at certain conditions. So our goal is to extend the GN model based PLM to capture the TP characteristics in a generic way so as to reduce the margin required for multi-vendor TP environment.

5.3 Multi-vendor PLM and Qtool

To extend the GN model to capture TP characteristics we introduce four performance factor, $\boldsymbol{v} = \{\alpha, \beta, \gamma, \delta\}$, where α and δ covers vendor specific TP components variations; β covers amplifier characteristics; and γ covers vendor specific DSP implementation variations [102], [103]. In a multi-vendor TPs scenario, the performance terms $\{\alpha, \beta, \gamma, \delta\}$ would be different for the heterogeneous TPs. Though this model is also applicable for single vendor TPs (or networks with alien wavelength/TPs), its importance is more relevant in multi-vendor networks. To be more specific, we consider a scenario, where the line system is open and TPs from M vendors are deployed (or available for deployment for the new connections). For a vendor i in M, we calculate the SNR of connection n which uses TP i (denoted as $i = TP(n)$) at the end of the path, with Eq. (5.4), instead of Eq. (5.2) as

$$SNR_n(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{z}) = \frac{\alpha_i . G_{O,n}}{\beta_i . G_{ASE,n} + \gamma_i . G_{NLI,n}}$$

$$= \frac{\left(\frac{\alpha}{\beta}\right)_i . G_{O,n}}{G_{ASE,n} + \left(\frac{\gamma}{\beta}\right)_i . G_{NLI,n}}, i = TP(n) \quad (5.4)$$

where $\boldsymbol{v}$ denote the TPs' parameters vector that includes $\boldsymbol{v}_i = \{\alpha_i, \beta_i, \gamma_i, \delta_i\}$ the performance factors of transponder i.

The QoT/SNR of connection n, calculated by the Qtool that accounts for the proposed multi-vendor TPs dependent performance factors is given as

$$Q_{MV}(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{z}) = [SNR_n(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{z})]_{dB} - SNR_{b2b,i} - DM_2,$$

$$i = TP(n) \ \ and \ SNR_{b2b,i} = SNR_{b2b} + \delta_i \quad (5.5)$$

where $SNR_{b2b,i}$ is the total dB penalty for the i –th TP in back to back (b2b) configuration, and δ_i is its vendor dependent variation to some reference SNR_{b2b} value (similar to Eq. (5.3)).

The performance factor βcorresponds to the amplifier performance and its contribution comes from the line system (OLS). An important effect that can be captured in the β parameter is the wavelength dependent penalty (additional ASE noise) due to EDFA gain ripple effect. Several works were published targeting to estimate the penalties contributed by this effect [15]-[17], [76]. However, this would be wavelength dependent and common for all TPs, and thus it will be a network (line system) and not a TP depend factor. Thus, for the remaining of this chapter we will assume $\beta = 1$ for all transponders.

Eq. (5.4) and Eq. (5.5) accounts for vendor specific performance factors. We call this as a multi-vendor Qtool and refer to it with Q_{MV}. Note that in past, a bias term was added in the accumulation of linear/ASE noise to account for TP implementations [123]. Our model, on the other hand, is more generic and captures a broader range of implementation factors, including such bias. Furthermore, while we started with the GN model and described our extensions to it in the preceding definitions, the concept is generic, and our proposed extensions can be applied to other PLMs/Qtools.

Q_{MV} includes performance factors coming from different vendor TPs. Hence, when modeling a multi-vendor TP network, the margin DM would be higher if we use a single vendor/traditional Q_{SV} compared to the margin used with the proposed multi-vendor Q_{MV}, that is, $DM_1 \gg DM_2$ (proved in upcoming results section). Note that in this work, we assume that for the connections, the transmitter – receiver (Tx. -Rx.) are from the same vendor. In more diverse network disaggregation scenarios where interoperability between Tx. -Rx. from different vendors is possible, such communication would follow a specific standardized configuration (modulation format, DSP, etc.). In such case, we should have specific $\boldsymbol{v_i}$ for the standardized interoperable configurations, allowing the proposed Q_{MV} to achieve good accuracy in such a network.

5.3.1 Case Study: Flat Power Optimization using Q_{SV} and Q_{MV}

In this section, we discuss the discrepancies in the SNR values, if Q_{SV} is used as a Qtool in networks where TPs from multiple vendors are deployed (or available for deployment) and how this could affect the optimization of the launch power of the channels.

In traditional or single-vendor optical networks, all network elements are *aggregated* and under the control of the proprietary controller, as shown in **Figure. 5.1**. Every network element configuration such as launch power of TPs, operating points of amplifiers etc. are decided by the vendor. One of the most crucial decision by the vendor is to adjust the TPs launch power. There are several ways to decide the launch power of the TPs in a wavelength division multiplexed (WDM) system. The most common practice is to set a flat launch power to all TP which is found to be the optimum for the central channel for full load on a multi-span link [10], [101]. Such power optimization results in low efficiency or excess margins for side channels or diverse network paths and can be improved using more sophisticated techniques [109]- [111]. In any case, setting a conservative launch power is definitely a good strategy for networks with limited knowledge about the used elements such as the disaggregated scenario [101]. In general, the impact of network disaggregation is accounted for by considering an additional penalty or margin as discussed in [8], [99] and also in the previous paragraphs.

Let us now evaluate the discrepancies in SNR values if we have a network where TPs $M = 4$ different vendors are deployed (or available for deployment). Note that the four different DSP chains implemented in VPI (described in Section 5.2) are treated as four different TP vendors in this section. **Figure. 5.5(a)** shows the VPI measured SNR values for the seven polarization-multiplexed 16-QAM channels at 0 dBm of flat/uniform launch power, when DSP chain from TP#1 is considered. As can be seen, the minimum SNR is obtained at the central channel, which is to be expected. **Figure. 5.5(a)** also shows the SNR values for these seven channels estimated after training the parameter vector $\boldsymbol{r}$ (fiber coefficients and bias) of the standard Qtool Q_{SV}. We also indicated the SNR (dB) estimation error between trained single-vendor Qtool and VPI measured values in the same plot, with mean MSE of ~7E-3 (details in upcoming results section). Note that, in order to improve fitting accuracy with more samples, we also used VPI monitored SNR values at different spans lengths (during fitting). This is the typical outcome, we should expect when training the standard Q_{SV} for the single vendor environment. A key point that needs to be highlighted here is that the results would be the same whether we assumed a single TP or multiple TP vendors. In other words, a single DSP chain (of TP

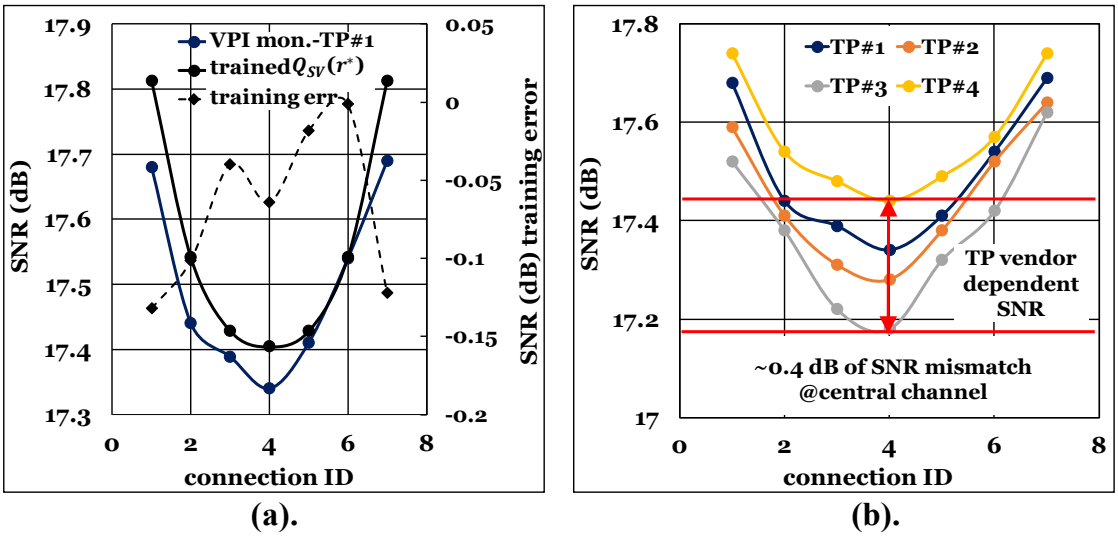


Figure. 5.5. (a) SNR (dB) values for VPI measured and trained Q_{SV} for seven channels considering DSP-1 chain, and (b) TP vendor dependent (different DSP chains) SNR (dB) values at flat optimized launch power values as presented in Figure. 5.4.

from vendor 1, assumed to be single vendor network scenario) is used for $M = 4$ TP vendors (with different DSP chains). This scenario mimics to the use of single vendor Qtool Q_{SV} for QoT/SNR estimations in a multi-vendor TP environment, which fails to account for differences in TPs performance among different vendors.

Now considering the scenario with $M = 4$ TP vendors (with different vendor dependent DSP chains). If we would use the above trained single vendor Qtool Q_{SV} for estimating their SNR values, since there is no TP/vendor dependent parameters in Q_{SV}, the SNR of all TP would be estimated as being the same, as shown in **Figure. 5.5(a)**. Instead, as shown in the **Figure. 5.5(b)** the real SNR values of the four TPs vary, and in particular we observe a ~0.4 dB difference in the SNR value of the center channel. Note that all the four set of SNR values for different TP vendors is measured at their corresponding flat optimized power as already shown in **Figure. 5.4**. So, we would require a ~0.4 dB of higher margin to cover this estimation error. The above-mentioned trained standard Qtool, which lacks TP vendor-dependent performance parameters, would only fit for one TP or one of the curves (other available TPs) of **Figure. 5.5(b)** (or an average via global training). In any case, the point here to emphasize is that the accuracy of the Qtool is compromised due to the lack of TP vendor dependent performance parameters. It is important to note that this is a special case, a linear network of two spans where all paths have the same length. For another/realistic network with diverse links and paths this margin value is higher, as discussed in the results section.

By definition, no TP vendor dependent performance factors are accounted in Q_{SV} calculations, as opposed to Q_{MV}. The vendor dependent factors accounted in Q_{MV} can improve the QoT estimation accuracy and enables the use of a low design margin. In this Chapter, we tackled the problem of TPs launch power optimization, where we assumed that we can choose for each channel a different launch power, according to its needs, a more sophisticated solution than the flat power optimization we discussed above. Also, we used the proposed multi-vendor QoT modeling scheme in a different optimization

problem, that of incremental planning. For both the use cases, we showed the improvements that we can obtain in their corresponding objective functions. This showcases that the proposed PLM/Qtool modeling scheme is generic and can benefit a variety of use cases.

5.4 Machine Learning Assisted Model Training

In this section, we provide the details on how to obtain the TP vendor specific performance factors/parameters $\boldsymbol{v}$, which can be done through ML based fitting technique in a greenfield deployment phase or while the network operates. Note that we already provide the brief discussion about TP characteristics and these factors through VPI simulations in the previous section. Once these performance factors are determined and plugged in the Q_{MV}, they improve the accuracy of QoT/SNR estimation, which in turn would be used for use cases such as incremental planning and transponders launch power optimization.

The idea in the training phase is to fit the parameters/coefficients of the Q_{MV} with measurements/monitoring in a testbed or in the operating network (reality/ground truth), so that proposed Q_{MV} behaves similar to the real world. Note that in this work, the term calibration is used to derive the TP vendor dependent performance factors using machine learning (ML) based fitting method. Hence the terms calibration or fitting refer at the end to the same goal, and, as so, are used interchangeably in the text. In this work, we relied on ML based nonlinear fitting techniques to do this. Since we consider QoT estimation, the SNR and the (bit error ratio) BER are the typical estimation targets. We chose the former, the SNR as the targeted value for our study. Regarding the input data/features, in PLMs the optical impairments depend linearly or nonlinear to the channels powers and also on several of the model input parameters. So, the input space in this work includes features such as connection's route, modulation format, symbol rate, launch power vector etc. However, the most important feature is the power level of connections as many other parameters such as interference noise, amplifier gain values etc. depends directly on this. Thus, to identify the related PLM or Qtool coefficients we should include in our input set scenarios with variable power levels.

To be more specific, in our proposed model, we assume that the input data includes the TPs launch power vector $\boldsymbol{p}$ of the N connections, whereas the target data is the monitored SNR vector (on input power vector $\boldsymbol{p}$). Note that in Section 2.9 of Chapter 2, we denote by **x** the independent variable that needs to be optimized with the LM algorithm based on the measured dataset. However, for the sake of clarity and to be aligned with the notation of Chapter 4, we here replaced the independent variable (**x** of Chapter 2) with $\boldsymbol{p}$ as the targeted problem is power optimization. Let $Y_n(\boldsymbol{p})$ denote the monitored SNR value of connection n which uses transponder $i = TP(n)$, and $\boldsymbol{Y_N}(\boldsymbol{p})$ denote the vector for all the connections n in N. In this work such monitoring is assumed to be done at the coherent receivers which would implement the vendor dependent DSP chains. Note that in the specific example that we consider we have seven established connections for which the monitored data is gathered from VPI set up as described in Section 5.2.2; in VPI we implement a DSP chain that is configured with four sets of parameters to emulate the effect of four different TPs/vendors. According to the used PLM, and in particular whether it calculates NLI based on the actual utilization or in more coarse calculations,

a change in the launch power of a single connection changes the SNR of several others, the ones that interfere/cross the same connection. In our case the implemented PLM model is the analytical version of the GN model with detailed NLI calculations per span.

To make a multi-vendor PLM, we first need to identify which parameters of the PLM model, and in particular the Q_{MV}, needs to be fitted/trained. In this work, we select following two sets of parameters to be trained based on the monitored information:

(i) $\boldsymbol{r}$– fiber coefficients- attenuation, non-linear coefficients, dispersion coefficients, and a bias (TP independent).

(ii) $\boldsymbol{v_i}$ – i-th TP vendor dependent performance factors $[\{\alpha_i, \gamma_i, \delta_i\}$ for all $i = 1, .., M]$, where M is the number of different TPs (four in our simulated scenario).

The generic goal is to apply a ML based non-linear fitting technique to fit the parameter vector $\boldsymbol{r}$ and/or $\boldsymbol{v}$ of the Q_{MV} with the monitored SNR information, that is $\boldsymbol{Y_N(p)}$. For this nonlinear fitting part, we relied on the Levenberg- Marquardt (LM) algorithm which is suitable for solving nonlinear least squares fitting problems [64]. The details about the algorithm are already presented in Chapter 2. The LM algorithm finds

$$\boldsymbol{r^*, v^* = (r, v)^* = argmin_{r,v}\left(Q_{N,MV}(p, r, v, z) - Y_N(p)\right)^2} \tag{5.6}$$

where the asterisk (*) symbol represents the corresponding trained parameter vector. Note that we assume known allocation of transponders to the connections, which could be included in $\boldsymbol{z}$. If we focus in a particular transponder type i, then above Eq. (5.6) can be modified to

$$\boldsymbol{r_i^*, v_i^* = (r, v)_i^* = argmin_{r_i,v_i}\left(Q_{N_i,MV}(p, r_i, v_i, z) - Y_{N_i}(p)\right)^2} \tag{5.7}$$

where N_i denotes the set of connections using transponder i, $\boldsymbol{Y_{N_i}}$ is the vector of monitored values and $\boldsymbol{v_i} = \{\alpha_i, \gamma_i, \delta_i\}$ for fixed $\beta_i = 1$, correspond to transponder i.

Note that we have two training options: *i) joint training* where we combine all training sets for all transponders and train globally, as described in Eq. (5.6), and *ii) separate training* where we train independently for each transponder type i, as described in Eq. (5.7), in which case we obtain $\boldsymbol{v_i^*}$ and M different $\boldsymbol{r_i^*}$ vectors. There are various methods to combine these $\boldsymbol{r_i^*}$ vectors and improve this fitting. In particular, we average the $\boldsymbol{r_i^*}$ vectors, make them constant, and rerun the fitting of Eq. (5.7) to only obtain the TP performance factors $\boldsymbol{v_i^*}$ for each TP i.

We now focus on two different ML assisted calibration schemes, depending upon the use case, for deriving the parameter vectors $\boldsymbol{r^*}$, $\boldsymbol{v^*}$.

5.4.1 Offline Training

This approach is applicable to scenarios where the TPs from the M different vendors are available to characterize prior to the field deployment, such as the planning phase or greenfield deployment. So, we envision to perform this in a laboratory prior to the field deployment of TPs. Each vendor TP is characterized by a possible set of modulation formats, symbol rates, launch power range etc. To plan the network, a PLM/Qtool would

Offline Training
available information:
(*no. of available TPs from M vendors and their characteristics such as possible mod. Format, sym. rates, launch power range etc.*)
for i = *1 : no. of TP vendors, M*
- *fix parameter vector* $\boldsymbol{r}$
- *configure* N_c *(e.g. 7) connections with central channel,* $n_i = \frac{N_c+1}{2}$ *being of type i*

for j = *-4 : 0.5 : 4*
adjust $\boldsymbol{p} = j.\boldsymbol{1}$*,where* $\boldsymbol{1}$ *is all ones vector monitor SNR,* $\boldsymbol{Y}_{n_i}(\boldsymbol{p})$ *for central channel*
end
- *find* $\boldsymbol{v}_i^* = argmin_{\boldsymbol{v}_i}(\boldsymbol{SNR}_{n_i}(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{v}_i, \boldsymbol{z}) - \boldsymbol{Y}_{n_i})^2$ *with LM algorithm*
- *use* $\boldsymbol{v}_i^*$ *for vendor i in* PLM_{MV}

end
use *M* trained PLMs for *M* vendors

Figure. 5.6. Pseudo code for offline training to determine TP vendor dependent performance factors.

be available and used by the network planner to access the quality of the connections prior to their establishment. In our case, this tool is the GN model Q_{MV}.

We assume that we generate N_c connections with the central channel $\left(n_i = \frac{N_c+1}{2}\right)$ being of TP type i at power vector $\boldsymbol{p}$, and we repeat this for all available TP types. We then measure the SNR value $Y_{\boldsymbol{n}_i}(\boldsymbol{p})$, of the central channel. The launch power vector $\boldsymbol{p}$ is then varied (for all channels) over a range of values such as from – 4 dBm to +4 dBm at a step of 0.5 dB. The SNR is monitored for the central channel n_i over the tuned power range and stored in a vector denoted by $\boldsymbol{Y}_{\boldsymbol{n_i}}$. The next step is to choose the parameters to train, which we identified to be the sets $\boldsymbol{r}$ and/or $\boldsymbol{v}$ in the previous subsection. Generally, in an offline training phase, accurate information about parameter vector $\boldsymbol{r}$ of the real network is pretty hard to obtain as there is no monitoring information from the network. Hence the goal in such an offline calibration phase is to fit only parameter vector $\boldsymbol{v}$ for a fixed/known $\boldsymbol{r}$ vector which is obtained by knowing the specifications of the calibration setup (spans, attenuators, EDFAs etc.) or by calibrating these parameters independently from the transponder.

To identify $\boldsymbol{v}$, we change the launch powers of the transponders and monitor the SNR vector as described above. If available, we repeat with another transponder of the same type in the center and/ or move the transponder to different locations apart from the center to enrich further our training set. After training with LM, the fitted $\boldsymbol{v_i}^*$ is obtained for vendor i and used in the Q_{MV}. **Figure. 5.6** provides basic pseudocode for offline calibration phase for TPs from M different vendors.

We now discuss the results of applying this ML assisted training method to fit the measurements obtained through VPI for the four TPs and presented in Section 2.2 and **Figure. 5.5(b)**. For this specific example, **Figure. 5.7(a)** and **Figure. 5.7(b)** shows the trained multi-vendor Qtool on TP#1 and TP#3 monitored data of VPI, respectively. The LM based training clearly fitted the Q_{MV} parameters ($\boldsymbol{v_1}$ and $\boldsymbol{v_3}$ and the others TPs, not presented in a figure for the sake of brevity) very well with maximum absolute training

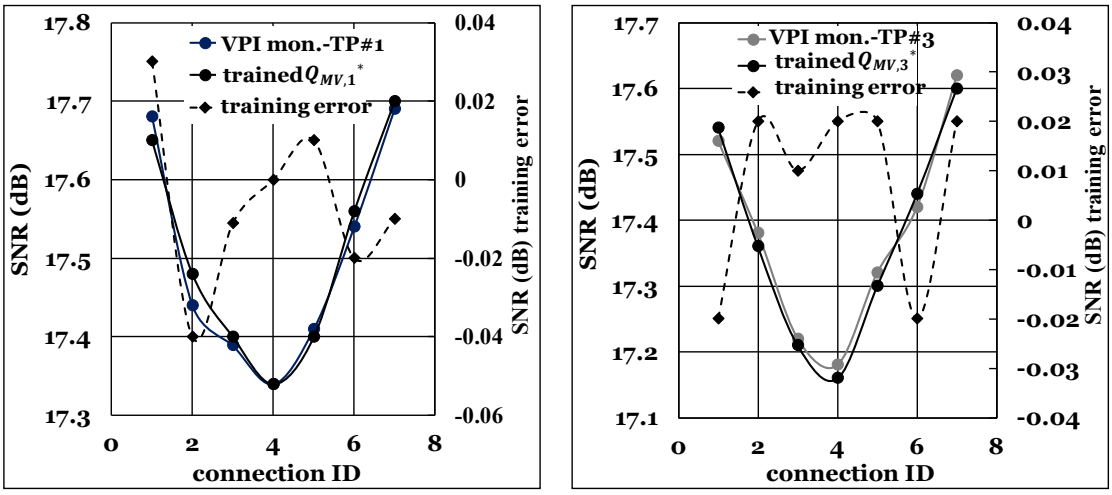


Figure. 5.7. standard and trained (a). $Q_{MV,1}$, and (b). $Q_{MV,3}$ with LM algorithm on VPI monitored dataset for TP#1 and TP#3 respectively along with training error.

error less than 0.05dB (for all TPs). It is worth noting that in this performed fitting we trained GN-based Q_{MV} with data measured in VPI; this indicates that the proposed TP modeling and the GN-model extensions can capture the performance variations of another more realistic model and gives us confidence that they can capture the related effects in real networks.

Table. 5.1 - Ground Truth/ Reality (Q_{MV}^{GT}) parameters for multi-vendor Qtool

Q_{MV}* model fitted parameters	fiber coefficients and bias ($\boldsymbol{r_i} = \boldsymbol{r} = fixed$)				TP performance factors ($\boldsymbol{v_i}$)		
	att. (dB/km)	non-linear coefficient (W/km)	dispersion ($ps/(nm.km)$)	bias (dB)	α_i	γ_i	δ_i
TP1	0.21	1.36	17.19	-2.6	0.81	0.78	0.85
TP2					0.82	0.96	0.72
TP3					0.96	0.86	0.91
TP4					0.82	0.84	0.94

Table. 5.1 indicates the trained/fitted set of $\boldsymbol{v_i} = \{\alpha_i, \gamma_i, \delta_i\}$ or the four TP vendors. Note that $\boldsymbol{r_i}$ vector was assumed to be known/fixed for all TPs (both at VPI and GN model), as discussed for the offline calibration above and also indicated in **Table. 5.1**, to specifically capture the TP vendor dependent performance factors. In particular, to get the optimized $\boldsymbol{r_i}$ vector of **Table. 5.1**, we initially trained all four TPs independently and then we averaged the $\boldsymbol{r_i}$ vector (including bias) over M TPs.

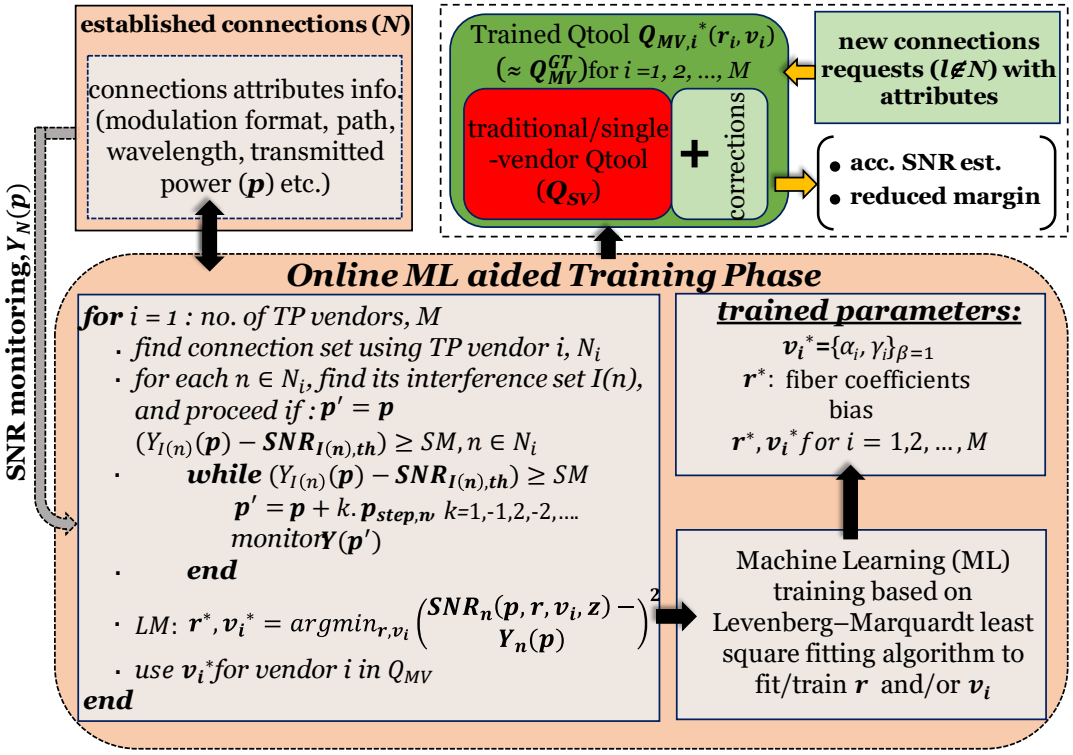


Figure. 5.8. Online training phase to determine TP performance factors along with the pseudo-code.

5.4.2 Online Training

The online training of the Qtool is applicable for more accurate QoT/SNR estimation in an operating network. The idea is to derive the TP performance factors from the established connections and use them in the Q_{MV}. One of the main assumption of this calibration technique is to know the TP category for the already established connections from the pool of M vendor TPs. Let us assume N established connections with power vector $\boldsymbol{p}$. Again the goal is to find $\boldsymbol{v} = [\{\alpha_i, \gamma_i, \delta_i\}, \text{for all } i = 1,2,..,M]$similar to offline calibration. However, the network is operational, and calibration, similar to that of an offline network, is an issue, as changing powers may render some connections infeasible, resulting in the suspension of some services.

We denote by $i = TP(n)$ the transponder type of connection n. We also denote by N_i the set of connections using TP from vendor i (with total TP vendors M). Then for the connection n let us denote by $I(n)$ the set of connections that n interferes. We choose connection n so that

$$(Y_{I(n)}(\boldsymbol{p}) - \boldsymbol{SNR}_{\boldsymbol{I(n),th}}) \geq SM, n \in N_i \tag{5.8}$$

where SM stands for safety margin (e.g. = 1 dB) and is chosen to avoid connections in $I(n)$ to reach infeasibility level. $\boldsymbol{SNR}_{\boldsymbol{I(n),th}}$ is the SNR threshold vector for $I(n)$ connection, depending upon their modulation format.

The next step is to vary $\boldsymbol{p}$ and collect the training dataset. We denote by $\boldsymbol{p_{step,n}}$ the vector that includes all zeros and a value of p_{step}(e.g. = 0.5 dBm) for connection n. For each TP vendor i, we identify candidate connections n in N_i with Eq. (5.8) and then change the initial $\boldsymbol{p}$ vector to $\boldsymbol{p} + k \cdot \boldsymbol{p_{step,n}}$, $k - 1,1, -2,2, \ldots$ This change in launch power vector $\boldsymbol{p}$ is performed for values of k until Eq. (5.8) stops to be satisfied. So we start with small positive and negative values of k and increase/decrease it as long as the criterion of Eq. (5.7) is met. This is described in more detail in **Figure. 5.8**. In this way we obtain a set of $\boldsymbol{Y_n(p)}$ for different $\boldsymbol{p}$ and the next step is to apply fitting as discussed in previous paragraphs (Eq. (5.6) and Eq. (5.7)).

The evaluation of this online training is quite more complicated than the offline one described above and necessitates network level simulations, presented in previous Section 5.4. Since VPI simulations cannot achieve network wide simulations, it cannot be used as the ground truth. As so we replaced it with a faster model, the proposed Q_{MV}^{GT}. Then the ground truth and the fitted model are both the same. The model used as the ground truth has several parameters (span fiber coefficients, bias etc.) that are hidden/unknown and are fitted in the second model. Note that this might have lower level of realism than using two different models, but we have verified in Section 5.3 and Section 5.4 that the proposed model fits very well to VPI, considered to be a very realistic and accurate model.

5.5 Use Cases

In this section, we present two use cases that verifies the importance of the proposed Qtool along with some basic mathematical description.

5.5.1 Launch Power Optimization

Adjusting launch power of the transponders is one of the crucial problem in traditional and next generation optical networks. There are two possible scenarios to adjust the launch powers of transponders, *i)* static - during network planning phase, *ii)* dynamic - while the network operates. However, in this chapter, we limit our study to static planning scenario, whilst a separate (next) chapter is dedicated to the dynamic version of this problem. In particular, in this section, we discuss about the first use case that emphasize on the amount of possible discrepancies on launch power optimization problem if we use Q_{SV} over Q_{MV}.

- Convexity and Optimization Objectives

For a generic topology we assume that we have to establish a set of $n = 1,2,..,N$ connections (planning phase). Let $\boldsymbol{p} = [p_1, p_2 \ldots p_N]$ represents the transponders (TP) launch power vector of those N connections. As the vector $\boldsymbol{p}$ represents powers that are non-negative, we can express $\boldsymbol{p} = e^{\boldsymbol{y}}$ in terms of a vector $\boldsymbol{y} \in \boldsymbol{R^N}$. As presented in the last chapters according to the GN model, the impact of optical fiber transmission effects on the generalized signal-to-noise ratio (SNR) of connection $n \in N$ can be modeled as

$$SNR_n(\boldsymbol{p}) = \frac{p_n}{P_{ASE,n} + P_{NLI,n}}, n = 1,2, \ldots, N \tag{5.9}$$

where p_n is the optical signal average power level, $P_{ASE,n}$ is the power of the ASE noise, and $P_{NLI,n}$ is the contribution due to power of NLI noise of connection n, with symbol rate of $R_{s,n}$.

Note that all terms in Eq. (5.9) are in linear domain. Also, according to the GN model presented in the last chapter, the power/contribution of the NLI parameter highly depends on the launch power vector $\boldsymbol{p}$. Moreover, $P_{NLI,n}(\boldsymbol{p})$ is a convex function of the scalar variable $\boldsymbol{p}$ for the one-dimensional subspace where the launch powers of all connections are equal, that is, $\boldsymbol{p} = p.[1,1,\dots,1]^{\boldsymbol{T}}$[109]. It is a locally convex function of the launch power vector variable $\boldsymbol{p}$ near this sub-space, where the power per connection is approximately equal, but becomes non-convex as the difference in power between neighboring (adjacent) connections increases. However, posynomial functions of this form, which are generally not convex, may be transformed into convex functions [124], [125].

The SNR is quasi-concave in linear power vector $\boldsymbol{p}$ for the region, near the equal-power subspace, where $P_{NLI,n}(\boldsymbol{p})$ is convex. On the contrary, where $P_{NLI,n}(\boldsymbol{p})$ is not convex, the SNR is not quasi-concave and has disjoint sub-level sets [124]. Using the transformation to the logarithmic power variables $\boldsymbol{y} = \log(\boldsymbol{x})$, a ratio of convex functions can be obtained and is given as

$$SNR_n(e^{\boldsymbol{y}}) = \frac{e^{\boldsymbol{y}_n}}{P_{ASE,n} + P_{NLI,n}(e^{\boldsymbol{y}})}, n = 1,2,\dots,N \tag{5.10}$$

The expression for SNR in Eq. (5.9) is neither convex nor concave, but taking a logarithm yields a concave function as given as

$$\log(SNR_n(e^{\boldsymbol{y}})) = \boldsymbol{y}_n - \log\left(P_{ASE,n} + P_{NLI,n}(e^{\boldsymbol{y}})\right) \tag{5.11}$$

where $P_{NLI,n}(e^{\boldsymbol{y}})$ is log-convex in $\boldsymbol{y}$, and $P_{ASE,n}$ is wideband and a constant, making the $SNR_n(e^{\boldsymbol{y}})$ in Eq. (5.11) log-concave in $\boldsymbol{y}$.

In order to define the quality of the connection n in the optical network, threshold SNR, $SNR_{th,n}$ is generally defined depending mostly upon the buadrate and modulation format. The difference between the SNR value of connection n at launch power vector $\boldsymbol{p}$, that is $SNR_n(\boldsymbol{p})$ and the threshold SNR $SNR_{th,n}$ is generally defined as the feasibility margin f for the connection n. The log of feasibility margin f is concave in the logarithmic power vector $\boldsymbol{y}$ as given by

$$feasibility\ margin\ f = \boldsymbol{y}_n - \log\left(P_{ASE,n} + P_{NLI,n}(e^{\boldsymbol{y}})\right) - \log(SNR_{th,n}) \tag{5.12}$$

Based on the Eq. (5.12), the objective is to optimize the launch powers of the N connections (transponders) in order to maximize

i) **Objective 1:** sum of connections margins

$$\max. f(\boldsymbol{p}) = \sum_{n=1}^{N}\left(log\ SNR_n(\boldsymbol{p}) - log\ SNR_{th,n}\right) \tag{5.13}$$

By maximizing Eq. (5.13), one can directly relate the capacity improvements in the network.

ii) **Objective 2:** minimum margin

$$\max. f(\boldsymbol{p}) = min_{n\in[1,N]}\left(log\, SNR_n(\boldsymbol{p}) - log\, SNR_{th,n}\right) \tag{5.14}$$

The channel with the least margin is the most likely to fail, so maximizing the minimum margin (Eq. (5.14)) serves to minimize the probability of the most likely failure.

subject to:

$$log\, SNR_n(\boldsymbol{p}) - log\, SNR_{th,n} \geq 0, \forall n \in [1, N]$$

$$\boldsymbol{p_{min}} \leq \boldsymbol{p} \leq \boldsymbol{p_{max}}$$

where p_{min} and p_{max} are the lower and upper power limits of the transponders' launch power; $SNR_n(\boldsymbol{p})$ is the SNR of connection n for the corresponding power vector $\boldsymbol{p}$; and $SNR_{th,n}$ is the SNR threshold required for the modulation format of n.

Note that, at high SNR, the log-concave in $\boldsymbol{y}$ is locally concave, allowing the possibility of the above two maximization objectives with optimization variable: power vector $\boldsymbol{y}$ (or $\boldsymbol{p}$) by solving the convex optimization problem [109].

5.5.2 Incremental Planning

This is the second use case to verify the working accuracy of the proposed multi-vendor Qtool. This use case is exactly similar to what we presented in Chapter 4 but with different modeling objective, that is performance variation of multi-vendor transponders. In this use case, we consider a stable network state, where a specific set of connections, N are established and the goal is to establish a new set of connections, $l \notin N$ in a network upgrade / incremental planning phase. From the monitoring information of establish set of connections N, we first train the multivendor Qtool in order to capture the above discussed performance factors with scheme presented in Section 5.3. Once the Qtool is trained, the goal is to show the performance improvements (design margin reduction) for new connection requests by using trained $Q_{MV,i}{}^{*}$ over standard Q_{SV}. In addition, we will also explore the possibility of achieving additional SNR savings for new connection requests by properly selecting the TPs.

5.6 Results and Discussions

In this section, we present the results for both the use cases discussed above. To quantify the benefits of the developed Qtool in multi-vendor TPs network scenarios, we performed network level simulations. For this analysis, we consider an Italian backbone topology with 27 nodes and 43 bidirectional links. The link lengths range from 80 to 480 km. We assumed standard single mode fiber (SSMF) spans of 80 km, a traffic load of 500 connections with uniformly chosen source-destination nodes. Note that each connection request consists of a source destination node pair, path/wavelength, modulation format and assigned TP. Each demand was modulated at 32 Gbaud with PM- {QPSK, 8-QAM, 16-QAM} leading to {100, 150, 200} Gb/s of data rate. We assumed attenuation coefficient of 0.2 $dB.km^{-1}$, dispersion coefficient of 16.7 $ps.(nm.km)^{-1}$ and

fiber non-linearity coefficient $1.3\,W.km^{-1}$ for the SSMF. We considered $M = 4$ TP vendor classes with $\{\alpha, \gamma, \beta\}$ performance factors $\{\alpha_i, \gamma_i, \delta_i\} = [\{0.81, 0.78, 85\}, \{0.82, 0.96, 0.72\}, \{0.96, 0.86, 0.91\}, \{0.82, 0.84, 0.94\}]$ and fixed $\beta = 1$ obtained from proper training of multivendor Qtool ${Q_{MV,i}}^*$ on VPI monitored data (**Figure. 5.6**). **Table. 5.1** shows the values for $\boldsymbol{r}$ and $\boldsymbol{v_i}$ that were set to Q_{MV}^{GT} which was used as the ground truth in these simulation. Each demand was served with one wavelength, assumed to be modulated at symbol rate of 32 Gbaud with uniformly chosen modulation-tunable polarization-multiplexed transponders from $M = 4$ vendors.

5.6.1 Results – Launch Power Optimization

To solve the above described two optimization objectives of Eq. (5.13) and Eq. (5.14) for transponders launch power optimization, we firstly implemented a heuristic algorithm. The heuristic starts from a uniform launch power of 0 dBm and uses Q_{MV} to estimate SNR value of each connection. It categorizes connections into "low margin" which need better performance, and "high margin" which have sufficient margin w.r.t. the threshold. Next, the heuristic iterates, and in each iteration the launch power is adjusted. In particular, at each iteration we increase/decrease the launch power values of low/high margin connections by a chosen step size Δ. We tried different values of Δ ranging from coarse steps of ±1 dBm to fine steps of ±0.25 dBm.

The optimization of channels' launch powers with the above objectives is known to be convex and of polynomial complexity as already presented in Section 5.4 and more details are available in [109]. Hence, we implemented an *interior-point algorithm* to solve it under the class of convex optimization techniques. It is well-known that such algorithm converges faster with a good starting/initial solution. Hence, we used the launch power vector returned by the heuristic as its starting point or initial solution.

Finally, we also applied an adaptive weight neural network (NN) inspired from [126]. The NN composed of single layer with neurons size equals to the number of connections N (size of optimization variables). It operates with a randomly generated initial population. The best obtained solution at each epoch (*i.e.,* temporal optimal solution) is set as the target and NN weights are updated accordingly. The NN adapts its weights by moving other predicted solutions towards the target solution, so that the objective improves with iterations

Note that we averaged the results over 20 iterations for all three implemented algorithms. **Table. 5.2** shows the computation time of the examined algorithms (heuristic, convex, NN) and standard and trained Qtools (Q_{SV}, ${Q_{MV,i}}^*$). Note that, in this use case the focus is proving the importance of the proposed Qtool over the standard single-vendor Qtool. Hence, for this particular use case, we assumed that we train the Q_{MV} form the previously discussed training techniques (known TP vendor dependent performance factors), which leads to ${Q_{MV,i}}^* \approx Q_{MV}^{GT}$ with MSE of $3.8\text{x}10^{-03}$. We observe a trade-off between computation time and objectives. The performance of the heuristic varies for the chosen step size Δ, the objective is better for smaller Δ, while its computation time increases almost linearly with Δ. Moreover, we also noticed that going below ±0.25 dB of step size did not make any substantial improvement although it took more time. Heuristic with the finest step size of ±0.25 dB took around 120 sec for obj#1

Table. 5.2. - Computation time (sec) for implemented algorithms

Implemented algorithms computation time (sec)	Obj.#1		Obj.#2	
	Q_{SV}	$Q_{MV,i}^{*}$	Q_{SV}	$Q_{MV,i}^{*}$
heuristic, Δ=±1dBm	56.5	68.8	69.5	72.4
heuristic, Δ=±0.5dBm	95.1	99.3	104.2	107.9
heuristic, Δ=±0.25dBm	120.2	131.6	134.8	147.2
convex+ heuristic	247.9	242.5	241.9	257.8
neural network (NN)	421.7	400.4	483.6	493.9

and ~140 sec for obj#2 to coverage with both Qtools (Q_{SV}, $Q_{MV,i}^{*}$). The interior point algorithm fed with heuristic (Δ =±1 dB) as initial solution took almost twice the time than heuristic to converge but showed substantial improvements in the objective goals compare to the heuristic.

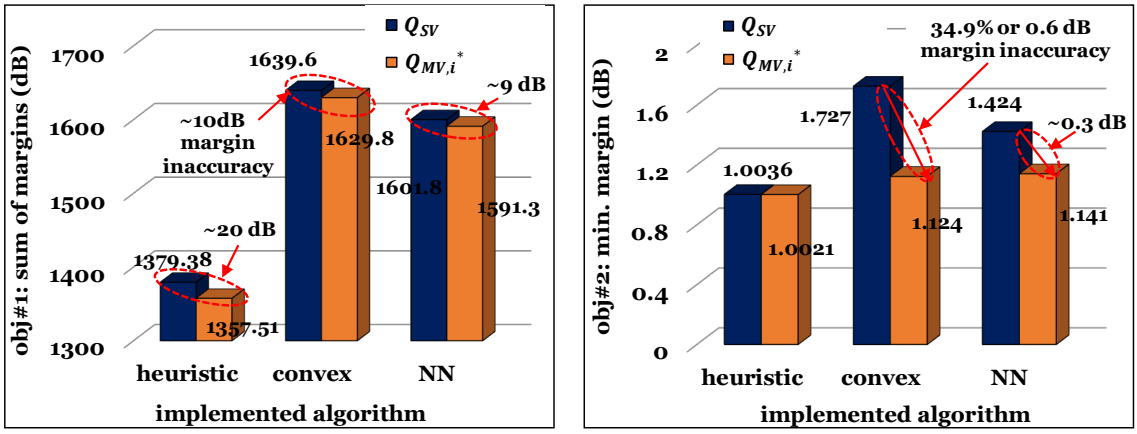


Figure. 5.9. (a) obj.#1, and (b) obj.#2 values and related savings when using $Q_{MV,i}^{*}$ instead of Q_{SV} for the multi-vendor scenario.

In general, NN is quite good in finding nonlinear mapping functions. However, for the multi-vendor launch power optimization problem at hand, convex optimization performed much better both in terms of optimization objective improvement and computation time as shown in **Figure. 5.9**. However, the NN algorithm achieved good performance (better than the heuristic) but with the highest execution time (even higher than the convex) as indicated in **Table. 5.2**. The convex (interior point) algorithm achieved the best performance in reasonable/low time. We observed ~10 dB of obj.#1 improvement and ~0.6 dB improvement of obj.#2 when using trained $Q_{MV,i}^{*}$instead of

Q_{SV} for the convex algorithm in the studied multi-vendor scenario. Note that the minimum margin

(obj.#2) reflects connections' infeasibility. These ~0.6 dB of minimum margin improvement can be reflected as ~34.9% reduction in infeasibility probability of the minimum margin connection. Hence the use of proposed multivendor Qtool reduces substantially the infeasibility probability of connections in a multi-vendor network. Varying improvements, depending on algorithm's performance and PLM inaccuracy were also observed for the other examined algorithms as indicated in **Figure. 5.9**.

5.6.2 Results – Incremental Planning

This use case is similar to the ones we presented in the Chapter 4. We consider a stable network state, where a specific set of connections are established and the goal is to establish a new set of connections, in a network upgrade/incremental planning phase. Each demand was served with one wavelength, assumed to be modulated at symbol rate of 32 Gbaud with uniformly chosen modulation-tunable pol.-mux. transponders from $M = 4$ vendors. We assume a specific set of connections, which we establish and are routed using a routing and spectrum assignment (RSA) algorithm based on shortest path and first fit slots. Since VPI simulations cannot achieve network wide simulations, it cannot be used as the ground truth. As so we replaced it with a faster model, the proposed Q_{MV}^{GT}. Then the ground truth and the fitted model are both the same. The model used as the ground truth has several parameters (span fiber coefficients, bias etc.) that are hidden/unknown and are fitted in the second model. Note that this might have lower level of realism than using two different models, but we have verified in previous sections that the proposed model fits very well to VPI, considered to be a very realistic and accurate model. Then we generate monitored data using Q_{MV}, Eq. (5.4) and Eq. (5.5). That is, we use the Q_{MV}^{GT} as the ground truth with parameters of **Table. 5.1** that are then assumed unknown and are estimated through the proposed ML training technique as discussed in Section 5.3.

So, we trained our PLM (or Qtool parameters) on these established connections and their monitored dataset. We then assume that we have a new set of connection requests of 10% of the total load. For example, at a load of 100 established connections which are used for training, additional 10% (10 new connections) are generated and need to be established. For those we estimate their QoT with our trained PLM. In other words, we assume that the training set is the set of established connections N and the testing set correspond to the new connections to be established, $l \notin N$. We averaged the results over 100 iterations (random sources-destinations) at each load.

We randomly assigned a TP, chosen uniformly among the pool of M vendors, to each connection. We assume a 0 dBm value of TP launch power is adjusted independent of TP vendor category for fair comparison. Taking as reference the SNR obtained by using Q_{SV} from Eq. (5.3), **Figure. 5.10** depicts the estimation error for 500 connections ($Q_{MV}^{GT} - Q_{SV}$), which pertains to the lack of knowledge about the TPs performance factors. The penalties were distributed in positive and negative sides depending upon the TP performance factors and were ~1.5 dB in total. Positive/negative penalties result in upper/lower bounds for the design margin, which we call as *high*/*low* margin. In

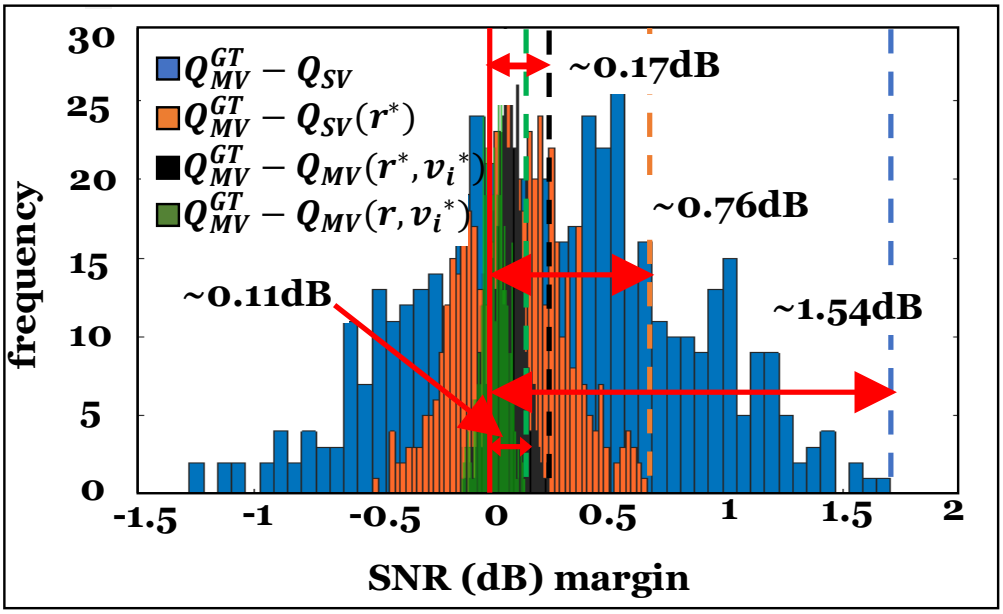


Figure. 5.10. Penalty distribution for 500 connections, indicating minimum required design margin to accommodate TP performance variations.

standards ~2-3 dB of design margin is typically used to accommodate all uncertainties [8]. Figure. 5.10. allows verifying that ~1.5 dB of QoT tool design margin would be required to accommodate these TP dependent penalties. The remaining part of the design margin would cover the other uncertain effects such as EDFA gain ripple, filtering penalties at nodes etc. To improve the estimation accuracy, the very first step is to find the trained $(\boldsymbol{r}, \boldsymbol{v})_i^*$ parameter vector for each i-th TP vendor separately (online calibration). We relied on LM algorithm to extract those fitted $(\boldsymbol{r}, \boldsymbol{v})_i^*$ parameter vector at the load of 500 established connections (new/unestablished connections: 10% of 500= 50).

The parameter vector $\boldsymbol{r}^*$ plays a crucial role in attaining good estimation accuracy. In [16], [70] the authors trained the main parameters of GN model based PLM treating it as a standard (single-vendor Qtool), such as fiber attenuation, nonlinear and dispersion coefficient ($\boldsymbol{r}$) along with a bias. In the following it is shown and discussed how much accuracy can be attained by proper selection of these parameters in case of partial disaggregated network scenarios. In particular, we applied online calibration approach to train $\boldsymbol{r}$ parameter vector. In this way we obtained the trained $\boldsymbol{r}$ parameter vector for all M TPs separately. However, the addition of $\boldsymbol{v}$ parameter vector on training phase affects the QoT estimation accuracy. Therefore, in the following 3 cases, we trained $\boldsymbol{r}$ and/or $\boldsymbol{v}$ for each TP vendor.

A. Case 1 (Reference) - Training GN Parameters Vector, r^*

This case is used as a reference case in our work and is inspired from [20], [16], [70]. In this case, only fiber coefficients ($\boldsymbol{r}$) and an additional bias is trained with the LM algorithm to minimize the nonlinear least square fitting error between Q_{SV} and the ground truth Q_{MV}^{GT}. In total we only have parameter vector $\boldsymbol{r}$ inside GN based PLM (Q_{SV}) that needs to be optimized in the online calibration phase.

Table. 5.3 - Single vendor Qtool training with only r^*

$Q_{SV}{}^*$ model fitted parameters	fiber coefficients ($\boldsymbol{r}^*$) and bias			
	attenuation ($dB/\ km$)	Non-linear coefficient ($W/\ km$)	dispersion ($ps/\ (nm.\ km)$)	bias (dB)
Same for all M=4 TPs	0.19	1.39	16.94	-1.85

As discussed in the previous section that there are two option of separate and joint training. However, we tried to fit all parameters of $\boldsymbol{r}$ together in this multi-vendor simulation environment similar to [70] for fair comparison. **Table. 5.3** shows the trained (green) parameter vector $\boldsymbol{r}^*$ (along with bias) obtained during online training phase at load of 500 connections. It can be clearly seen from the optimized parameter values that the parameter bias tried to capture the information about the TP vendor characteristics. However, the rest three parameters are also adjusted accordingly to minimize the error by Eq. (5.5). This case is used as a reference case for our work.

B. Case 2 - Training GN Parameters Vector $r_i{}^*$ and TPs Parameters Vector $v_i{}^*$

This case corresponds to our proposed solution. We propose to additionally train parameter vector $\boldsymbol{v_i}$ (per TP vendor) along with the $\boldsymbol{r}$ parameter vector, which was trained in the previous reference case. Note that the proper selection of trained parameters during online training phase plays a crucial role in overall estimation accuracy. In this case, the fiber coefficients and bias ($\boldsymbol{r_i}^*$) trained for each transponders and then averaged $\boldsymbol{r}^*$ parameter vector is obtained after taking the average on independently trained vendor dependent $\boldsymbol{r_i}^*$ and bias values. These $\boldsymbol{r}^*$ is then kept constant and only $\boldsymbol{v_i}^*$ is fitted to observe the effect of TP vendor dependent performance factors.

Table. 5.4 – Multi-vendor Qtool training with $r_i{}^*$ and $v_i{}^*$

$Q_{MV}{}^*$ model fitted parameters	fiber coefficients and bias ($\boldsymbol{r_i}^*$)				TP performance factors ($\boldsymbol{v_i}^*$)		
	att. dB/km)	non-linear coefficient (W/km)	dispersion ($ps/\ (nm.km)$)	bias (dB)	$\alpha_i{}^*$	$\gamma_i{}^*$	$\delta_i{}^*$
TP1	0.19	1.34	17.28	-2.1	0.78	0.78	0.83
TP2					0.81	0.92	0.76
TP3					0.92	0.82	0.88
TP4					0.84	0.81	0.91

Table. 5.4 shows all trained (green) parameters obtained during online calibration phase. From the table, the maximum information about the TP features (for a particular vendor) is available in trained $\alpha_i{}^*$, $\gamma_i{}^*$and $\delta_i{}^*$. From the table one may observe that the obtained values of $\alpha_i{}^*$, $\gamma_i{}^*$and $\delta_i{}^*$ are not exactly similar to what was plugged in the Q_{MV}^{GT}, also shown in **Table. 5.1**. This happened because 4 additional parameters ($\boldsymbol{r}$ vector and bias) were used and the training algorithm needs to find the best combination of all the 6 parameters for each TP vendor with nonlinear least square fitting technique.

C. Case 3 - Training TPs Parameters Vector $\boldsymbol{v_i}^*$

This case assumes that we have perfect knowledge of fiber coefficients (that is $\boldsymbol{r}$ vector of **Table. 5.5** exactly similar to **Table. 5.1**) and the goal is to trace optimum parameter vector $\boldsymbol{v_i}$ for each TP vendor separately. In a sense this is an unrealistic case mainly used as an upper performance bound. The training process here is quite straightforward and similar value of fitted $\alpha_i{}^*$, $\gamma_i{}^*$and $\delta_i{}^*$ can be attained as shown in **Table. 5.5**.

Table. 5.5 - Multi-vendor Qtool training with only $\boldsymbol{v_i}^*$

$Q_{MV}{}^*$ model fitted parameters	fiber coefficients and bias ($\boldsymbol{r}^*$)				TP performance factors ($\boldsymbol{v_i}^*$)		
	att. (dB/km)	non-linear coefficient (W/km)	dispersion ($ps/(nm.km)$)	bias (dB)	$\alpha_i{}^*$	$\gamma_i{}^*$	$\delta_i{}^*$
TP1	0.21	1.36	17.19	-2.6	0.80	0.79	0.82
TP2					0.84	0.95	0.74
TP3					0.94	0.88	0.92
TP4					0.81	0.84	0.91

As shown the fitted $\boldsymbol{v_i}^*$ (green) parameter vector is almost similar to the one used in ground truth Q_{MV}^{GT} (**Table. 5.1**) and can result in most accurate SNR estimation compare to rest of the cases. However, the major limitation of this approach is the availability of highly precise and accurate information of other physical layer parameters (fiber coefficients in this case), which make this case not feasible in practical network scenario. We also calculated the mean square fitting error (MSE in dB^2) on the SNR (dB) values for all the above discussed three cases with the trained $Q_{MV,i}{}^*$ for each TP vendor i. As expected, we noticed that the minimum value of MSE =$8.82\text{x}10^{-04}$ for the Case 3. For Case 1 and Case 2, the MSE varies from 0.0018 to $3.82\text{x}10^{-03}$ respectively. Note that this MSE is calculated on the training dataset at online calibration phase and it only represents how accurately the model is trained on the training dataset.

Figure. 5.11 shows the maximum and minimum overestimation (positive) and underestimation (negative) error on SNR in dB for the three examined Qtool cases, and the improvements relative to the traditional untrained Q_{SV}. Note that we define as maximum high margin, the maximum overestimation and as minimum high margin, the minimum overestimation error. Similarly, we denote maximum low margin, the maximum underestimation and as minimum low margin, the minimum underestimation

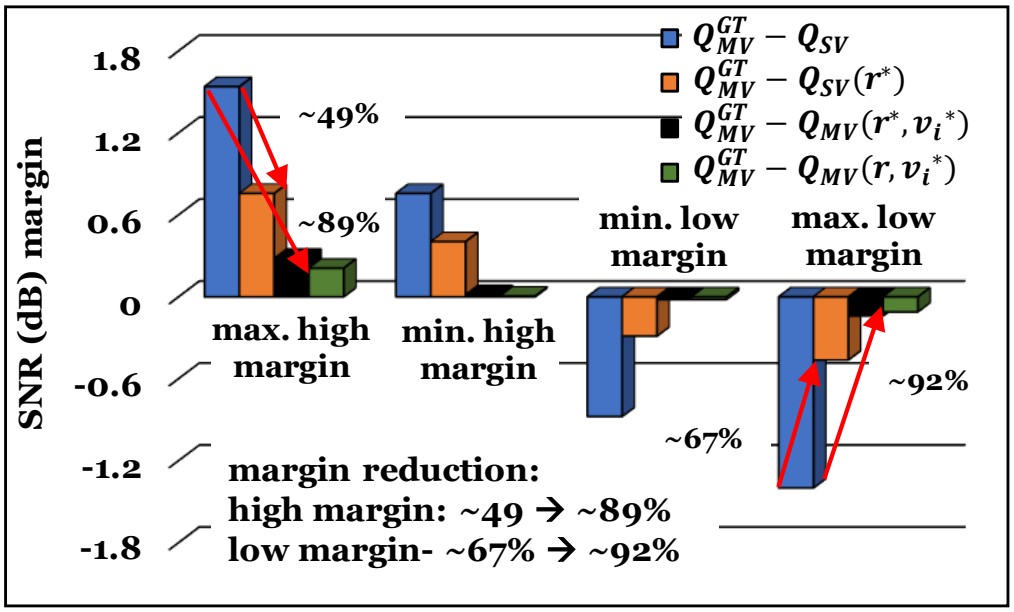


Figure. 5.11. Minimum-maximum overestimation and underestimation error range for both high and low margin sides: for the untrained-standard Q_{SV}, for the trained $Q_{SV}(r^*)$, for the proposed $Q_{MV}(r_i^*, v_i^*)$, and the Q_{MV} with perfect physical layer knowledge (r, v_i^*) with respect to ground truth Q_{MV}^{GT}, on testing dataset at a load of 500 connections.

error. This maximum overestimation/underestimation is the reduced estimated high/low margin for the Qtool implemented with trained $Q_{MV,i}{}^*$.

For high margin, it is found to be ~0.76 dB, yielding a ~0.78 dB margin reduction when only trained fiber coefficients along with an additional bias are used (Case 1 as reference). For low margin, this margin is reduced to ~0.46 dB ultimately resulting in margin reduction of ~0.94 dB. For Case 2, where three additional parameters $\alpha_i{}^*$, $\gamma_i{}^*$ and $\delta_i{}^*$ (per TP vendor) are also trained result in ~1.3 dB of both high and low margins savings as shown in **Figure. 5.11**. Similar percentage of savings are also observed in the minimum high and low margins as shown in **Figure. 5.11**. As we already discussed that Case 3 is not very realistic, since it assumes that precise information of fiber coefficients are already known with high accuracy and only parameters α^*, γ^* and δ^* (per TP vendor) needs to be trained. Based on parameters obtained during training process (**Table. 5.5**), we comparatively noticed a little bit higher margin savings. In total, with Case 2, we achieved additional ~40% and ~25% margin reduction compared to Case 1 for maximum high and maximum low margin respectively. Similar savings are also indicated through **Figure. 5.10** for better understanding of readers.

D. TPs Assignment and SNR Savings

In continuation to Qtool training and margin reduction for unestablished connections, we also examined how we can use such information to decide the appropriateness of the available TPs from the M vendors for unestablished connections. For each new connection request $l \notin N$, the QoT/SNR is estimated for all the available options from the available TP vendors. For the SNR estimation, we used trained Qtools $Q_{MV,i}{}^*$ for the

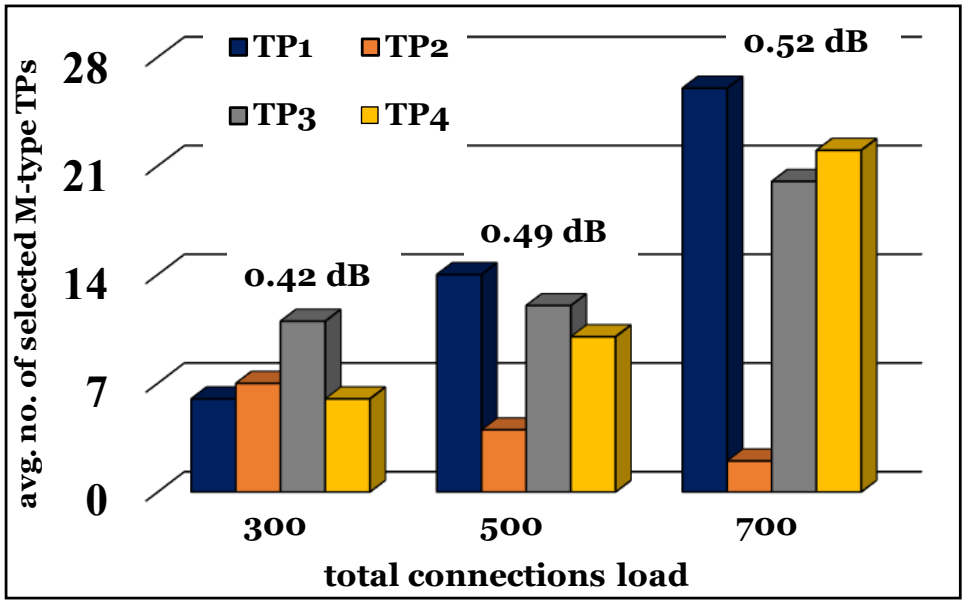


Figure. 5.12. Additional SNR (dB) improvements by proper selection of TPs from *M*=4 vendors at different traffic loads.

particular TP vendors (**Table. 5.4**). Out of those TPs, the one with the maximum SNR was chosen over the other ones. **Figure. 5.12** represents, the number of newly established connections (10% of the load) that used each available TP from $M = 4$ vendors at different loads. Note that the TPs are uniformly assigned for total connection requests (established and unestablished). Hence our interest in this work is to showcase the importance/role of vendor dependent TPs performance factors in TPs assignment to new connection requests. For this we uniformly generated and averaged the traffic (load) instances 100 times to have a fair selection of TP from $M = 4$ vendors.

At low loads, TPs from vendor 2 and vendor 3 are more utilized as the nonlinear noise contribution is low and thus TPs with better α^* parameter (more important/dominant) are selected as indicated in **Figure. 5.12**. At high loads, the amount of TPs assignment form vendor 2 is substantially decreased compared to the low loads due to worst value of γ^* resulting in high penalty in the nonlinear noise. Also, the utilization of TPs from vendor 3 and vendor 4 is increased from medium (500) to high load (700) compared to vendor 2 for the considered network topology. It is because the links are highly occupied and now the nonlinear noise is more prominent at high loads. **Figure. 5.12** also shows the additional maximum SNR (dB) value that can be achieved by proper selection of TPs for unestablished connection requests. We observed an additional minimum-maximum range of SNR improvement from 0.42-0.52 dB is available by proper selection of TPs from four different vendors at different traffic loads.

5.7 Conclusions

We proposed PLM extensions that capture the performance variations of multi-vendor TPs in this chapter. It is worth noting that, the next-generation of high symbol rate TPs (90 to 140 Gbaud) comes with an integrated amplifier, which could have a non-negligible impact on the proposed multi-vendor Qtool performance. We also devised a monitoring and ML based scheme (based on non-linear fitting) to estimate the TP vendor dependent

performance factors for accurate Qtool modeling in partial disaggregated optical networks. The trained Qtool would be then used for various use cases. However, in our study we verified the working accuracy and improvements obtained with the proposed Qtool modeling scheme in two different use cases, that is, transponders launch power optimization (static) and incremental planning. Using this multi-vendor Qtool with convex algorithms for TPs launch power optimization, we observed ~0.6 dB higher SNR estimation accuracy and ~35% minimum margin savings compared to a traditional Qtool modeling scheme. On the other hand, for incremental planning use case, with the proposed approach, we accomplished a design margin reduction of ~1.54 dB to ~0.18 dB for new connection requests with respect to the standard Qtool. We also showed additional SNR benefits of upto ~0.5 dB that can be achieved by proper selection of TPs from different vendors on top of the previously reduced design margins. This indicates that we can reduce overprovisioning of emerging disaggregated optical networks by combining the proposed Qtool with the resource allocation algorithm to achieve such savings.

Chapter 6

QoT Estimator Retraining and Dynamic Optimization

This Chapter is devoted to investigating the stated objective O3. To this end, we extend the exploration of the QoT estimation tool accuracy in context of dynamic optimization operations. In general, optical network optimization involves an algorithm and a PLM to estimate the QoT of connections while examining candidate optimization operations. In particular, the algorithm typically calculates intermediate solutions until it reaches the optimum which is then configured to the network. If it uses a Qtool that was aligned once to reflect the starting network configuration, then the algorithm within its intermediate calculations can project the network into states where the Qtool suffers from low accuracy, resulting in a suboptimal optimization. Keeping this in mind, we propose to solve dynamic multivariable optimization problems with an *iterative* and *adaptive* closed control loop process, where after certain algorithm steps the intermediate solution is configured and monitored to realign/retrain the QoT tool to follow the projected network states. The QoT estimation tool is used as a Digital Twin (DT), a digital representation of the real system which is realigned during the dynamic optimization process. In particular, we discuss the dynamic version of the launch power optimization problem presented in Chapter 5. Specifically, there is a set of established connections whose launch powers are optimized while the network operates. We observed substantial improvements in the sum and the lowest margin when optimizing the launch powers with the proposed approach over optimization using a one-time trained Qtool. The proposed approach achieved near to optimum solutions as found by optimizing and continuously probing and monitoring the network, but with a substantial lower optimization time.

6.1 Role of QoT Estimation Tool in Dynamic Optimization

An accurate and fast QoT estimation tool (or Qtool) is required for almost every optimization task of an optical network. Today most of the optimization tasks are static. For example, the network setup and upgrading calculations are performed in advanced. The Qtool relies on a PLM and includes margin that covers modeling uncertainties of the PLM along with the evolution of the physical layer conditions over the targeted lifespan. Moreover, as soon as the connection is provisioned/ established, the vendor can measure its QoT (e.g. the SNR), and correct/improve the configuration. The upgrades that involve dynamic operations such as the establishment of new or reconfiguration of established

connections were classified as static, since typically they are carried out in maintenance windows and not on the operating network. Dynamic reconfigurations for resiliency involve protected/restored connections which were probed beforehand. In any of these optimization tasks the QoT estimation tool needs to be accurate. However, the dynamic operations are not directly applied on the network, which results in an indication of the lack of certainty for such operations. As discussed in the Chapters 2 to Chapter 5, both monitoring and ML techniques play an essential role in improving the Qtool estimation accuracy. This in turn improves the efficiency of static optimization and paves the way to reduce overprovisioning and realize some dynamic optimization use cases [12], [13], [56], [116], [127]- [129].

Let us consider a network upgrade/incremental planning task which involves calculations for new establishments and possible reconfigurations of the already established connections [12], [70]. Traditionally a Qtool with high margins is used, e.g. considers pessimistic fiber coefficient parameters, full spectral load, high modelling inaccuracy etc. Consequently, the achieved optimization may be inefficient entailing considerable resource overprovisioning. Using monitoring feedback and ML techniques, the Qtool parameters (or its PLM) can be fitted/trained so that its estimated SNR values are close to those monitored in the network. We refer to this process as the alignment of the Qtool to the physical layer of the network. The Qtool accuracy is even more critical when used for dynamic optimization tasks, where the target is to achieve high efficiency in an operating network. This can be effectively combined with the intrinsic programmability advantages brought by a SDN controller [130], [131]. By doing so, the SDN controller can implement the optimization logic and interfaces with a Qtool (reference architecture similar to the one presented in Chapter 2). This enables to handle closed control loops leveraging the monitoring data as input or feedback to conduct the targeted optimization task [13], [112].

Similar problems arise in almost every industry. To keep up with the rapid advancements of the systems and harvest their improvements in terms of productivity, the Digital Twin (DT) concept is gaining a lot of attention. The DT is a digital representation of the real / physical system, used to understand and optimize the targeted system [132]. According to the definition of [133], the DT is more than a model of the system; it includes an evolving set of data, and a means of dynamically adjusting the model. The DT concept was originally introduced in 2003 [133] and first put to public by the NASA [134]. Different industry sectors are taking advantage of DT's ability to simulate real-time working conditions and perform autonomous and intelligent decision-making operations. DT provides an alternative way in today's manual interaction-based design, operation, and service paradigms, to solve the related challenges autonomously and in real-time [132], [135]. Depending on the dynamicity of both the system and the optimization process, the DT needs to represent the real system with certain accuracy. To do this, the DT is integrated and realigned with the physical system. Such a realignment mechanism typically involves monitoring and ML schemes.

Turning our attention back to the optical network, the target is to use the Qtool as a DT. That is, a model with an appropriate set of parameters along with a mechanism to adjust them to support the optimization task at hand. For static optimization tasks, such as the incremental planning, the only option is to train the Qtool once, just before taking the

decisions for the entire optimization task. This results in lower margins and increased network efficiency. But the main target and benefits of DT comes in dynamic optimization. In dynamic optimization, we would like to squeeze the margins and achieve higher efficiency, making the accuracy of Qtool a critical factor. For example, the accuracy of the Qtool deteriorates as connections are established/ released/ rerouted/ change their power. In other words, in dynamic optimization tasks involving required calculations and actions (e.g. the establishment or reconfiguration of a single connection) the accuracy of the Qtool would be acceptable if it was realigned before the calculation. However, realignment of the Qtool is expensive. It requires one or more control loops, including monitoring that can be time consuming and thus it might not be feasible. Moreover, for more complex/multivariable dynamic optimization tasks, that require multiple reconfigurations the accuracy of the Qtool can become critical. The algorithms adopted for such cases are typically iterative. That is, they calculate several intermediate solutions and improve over them to find the optimum, which is then configured in the network [128]. However, the Qtool accuracy deteriorates after several intermediate calculations and indeed after a point it may fail to support the optimization calculations. The key advantage is that the network operates and thus we can realign the Qtool/retrain its parameters, so that it follows the projections to states intermediately calculated by the algorithm.

We present the dynamic version of the launch power optimization problem discussed in Chapter 5. We assume that there is a set of established connections whose powers need to be optimized while the network operates. From Chapter 2 and Chapter 5, it is clear that the optimization of launch power of TPs is a well-defined convex problem of polynomial complexity. Therefore, the optimum launch powers can be found with a convex optimization algorithm that performs several intermediate calculations. To solve the problem, three methods are investigated

i) having the optimization algorithm probe and monitor the network at each intermediate iteration,
ii) using a one-time trained Qtool for all optimization iterations,
iii) implementing an iterative closed control loop process that after a number of intermediate iterations (fixed or adaptive) configures the network, monitors and retrains the Qtool.

The last option is the *proposed solution*, exploiting the optimization with a DT. It includes, apart from the Qtool, the evolving network conditions, appropriate choice of parameters for the Qtool and the process to align it to support the dynamic optimization at hand. Although we applied our proposed solution to the dynamic launch power optimizing problem, the proposed iterative closed control loop which includes the realignment of the Qtool is generic. It can be applied to other dynamic multivariable optimization problems such as dynamic resource allocation, automatic network reconfiguration, defragmentation, virtual network reconfiguration etc. [13], [107], [112]- [118], It also provides ideas about how to realign the Qtool in simpler dynamic and even static optimization tasks.

Finally, we would like to point out that our study is quite more realistic than most of the previous works that use the same Qtool as both the estimator and the ground truth, to

generate the information to train the estimator (Chapter 3). Specifically, in our simulations we used VPI as the ground truth and the GN model (similar to last two Chapters) as the estimator. VPI is quite more detailed and complex and closer to a real system than the GN model. This choice was made to capture the mismatch between the real network and the Qtool that would be used in the optimization process, an additional difficulty which is neglected in most previous works.

6.2 Use Case and Motivation

In this section, we investigate how the Qtool accuracy affects the optimization calculations. To do this, we focus on a dynamic version of the launch power optimization problem. We assume that a set of connections are established, and our goal is to optimize their launch power. Thus, no connections are established or released, but the existing connections are reconfigured (their launch powers are adjusted) as the network operates. To motivate and better understand the problem, we discuss the optimization of the launch powers of 25 channels transmitted over a single link.

6.2.1 QoT Estimation Tool Training

We created a single link with six identical spans setup in VPI Transmission Maker [14] as shown in **Figure. 6.1**. On this link we simulated the transmission of 25 channels at 32 Gbaud with polarization multiplexed (PM)-16QAM and assumed SNR threshold of $SNR_{th} = 13.9$ dB for each channel [107]. Note that these simulations were time consuming due to the high computational complexity, as VPI uses split-step Fourier propagation simulations to model the nonlinear signal propagation of the channels. We considered the VPI setup as the '*real-world*', the actual optical network. We also implemented a Qtool, and in particular the GN-model [50], with a similar setup of six identical spans and 25 channels. We found approximately 1 dB of max. SNR difference between the Qtool and VPI, when all parameters of the Qtool and VPI (i.e., dispersion coefficient, slope, attenuation coefficient of fiber, non-linearity coefficient etc.) were set equal. Then we aligned the Qtool with the real world (VPI). This alignment can be done with various methods. In our case, we monitored the channels SNR values (in VPI) and adjusted the Qtool parameters so that its SNR estimations match with the real world (VPI), using ML.

To be more specific, we implemented the following alignment process for the GN model. We assume a network with N connections. The GN PLM based Qtool is a model that takes as input several parameters and calculates the SNR values of the connections. Keeping the same notation of Chapter 5, let $\boldsymbol{r}$ denote the similar set of GN model fitted parameters: *i)* fiber attenuation coefficients, *ii)* fiber non-linear coefficients, *iii)* fiber dispersion coefficients, *iv)* a wavelength dependent penalty term, implemented as a 4^{th} order polynomial, to cover transponder loss mismatches, amplifier ripple etc. [16], and *v)* a bias.

Similar to Chapter 5, let $\boldsymbol{p} = [p_1, p_2, \dots p_N]$ be the launch power vector of the N connections, which are the variables that will be optimized later, and let $\boldsymbol{z}$ represents the unchanged input parameters for our optimization, such as routes, used wavelengths, span lengths etc. We denote by $Q_n(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{z})$ the GN model SNR estimation for connection n, and with $\boldsymbol{Q_N}(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{z})$ the SNR vector for all connections N. The SNR calculation

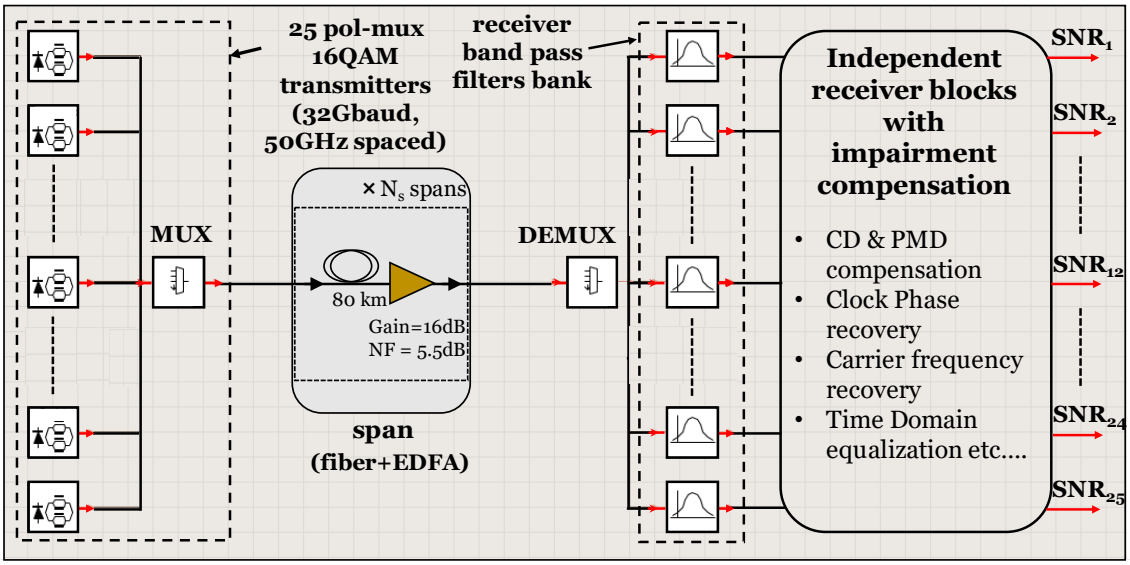


Figure. 6.1. Simulated single link setup in VPI with 25 x 32 Gbaud, PM- 16QAM, 50 GHz spaced transmitters and six identical spans.

function is nonlinear in its parameters $\boldsymbol{r}$ (and also $\boldsymbol{p}$). Finally, let $\boldsymbol{Y_n}$(p) denote the monitored SNR value of connection n and $\boldsymbol{Y_N(p)}$ be the vector for all the connections N. In this work such monitoring is assumed to be done at the coherent receivers. The training error vector is given by $\boldsymbol{Q_N(p,r,z) - Y_N(p)}$, and the objective of the training/fitting is to identify the parameters $\boldsymbol{r}$ that minimize the squared error. To fit this, we relied on the Levenberg- Marquardt (LM) algorithm which is suitable for solving nonlinear least squares fitting problems as already presented in Chapter 2. The LM algorithm finds

$$\boldsymbol{r_0 = \mathrm{argmin}_r (Q_N(p,r,z) - Y_N(p))^2} \qquad (6.1)$$

When we perform this PLM alignment once, before the optimization task, the PLM reflects with good accuracy the starting state of the network prior to optimization. We refer to this as one-time trained Qtool and denote it by $\boldsymbol{Q_N(p,r_0,z)}$.

We studied two types of EDFA: *i)* whose gain is perfectly flat/ideal, and *ii)* another with a gain ripple profile of a peak-to-peak (p2p) value of ±0.2 dB, similar to Chapter 4. Note that the EDFAs were assumed to be operated in AGC mode with average gain equal to the previous span loss. We call the setup with the flat span EDFAs as Case 1, and the setup with EDFAs having gain ripple as Case 2. Case 1 represents an ideal network with relatively stable physical layer conditions. On the other hand, Case 2, with rippled EDFAs, represents a more realistic scenario with more dynamic physical layer conditions. The physical layer dynamicity comes from the fact that an EDFA with a gain ripple introduces SNR variations when changing the connections powers. These variations are hard to estimate, unless the gain profile of the EDFA is perfectly known. This profile is hard to be found in an operating network and it might change over long time.

Figure. 6.2(a) and **Figure. 6.2(b)** show the estimated SNR values of the connections from the one-time trained Qtool $\boldsymbol{Q_N(p,r_0,z)}$ and the real network (VPI) $\boldsymbol{Y_N}$(p) at uniform

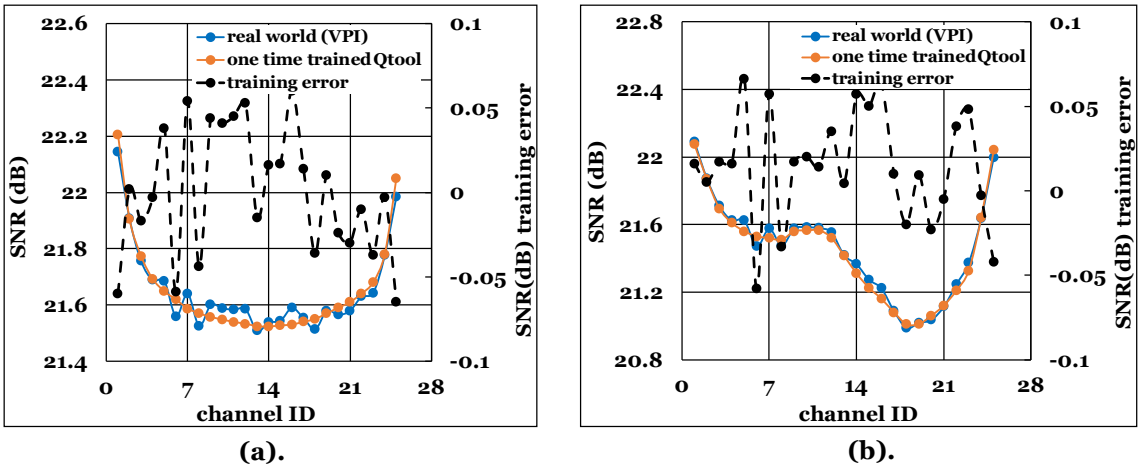


Figure. 6.2. Estimated SNR values from the one-time trained Qtool and VPI at uniform 0dBm launch power and the related training error for (a) flat EDFA, and (b) EDFA with gain ripple.

launch power of $\boldsymbol{p} = 0\ dBm$, for both flat and rippled EDFAs, respectively. The corresponding training errors are also displayed in the same figures. With one-time training, the PLM parameters were adjusted quite well and its estimated SNR values matched those of the actual optical network/real world at the initial state. This is deduced by the very low errors, less than 0.1 dB for both, Case 1 and Case 2.

6.2.2 Dynamic Launch Power Optimization

In this section, we specifically tackle the dynamic launch power optimization problem. Note that the optimization objectives are similar to the ones presented in Chapter 5. However, for the sake of completeness and better understanding, we again present them here. For a generic topology, a set of N connections are established. The objective (similar to Chapter 5, but for an operating network) is to optimize the launch powers of the N transponders to maximize

(i) **Objective 1:** sum of connections margins

$$\max. f(\boldsymbol{p}) = \sum_{n=1}^{N} \left(log\ SNR_n(\boldsymbol{p}) - log\ SNR_{th,n} \right) \tag{6.2}$$

(ii) **Objective 2:** minimum margin

$$\max. f(\boldsymbol{p}) = min_{n\in[1,N]} \left(log\ SNR_n(\boldsymbol{p}) - log\ SNR_{th,n} \right) \tag{6.3}$$

subject to:

$$log\ SNR_n(\boldsymbol{p}) - log\ SNR_{th,n} \geq 0, \forall n \in [1, N]$$

$$\boldsymbol{p_{min}} \leq \boldsymbol{p} \leq \boldsymbol{p_{max}}$$

where $\boldsymbol{p} = [p_1,\ p_2, \ldots\ p_N]$ is the launch power vector of the N connections; p_{min} and p_{max} are the lower and upper power limits of the transponders' launch power; $SNR_n(\boldsymbol{p})$

is the SNR of connection n for the corresponding power vector $\boldsymbol{p}$; and $SNR_{th,n}$ is the SNR threshold required for the modulation format of n.

The optimization of channels' launch powers with one of the above objectives is known to be convex and of polynomial complexity [109]. Moreover, we already presented in the last Chapter 5 the objective improvements with convex optimization techniques over heuristic and neural networks. Hence, we used the interior-point algorithm (convex optimization technique) to solve it. In general, a convex optimization algorithm performs intermediate calculations/iterations. At each intermediate iteration the algorithm decides on new transponders launch powers to move towards the optimum. However, these intermediate steps are internal. That is, only the final (optimal) is configured in the network. The algorithm decides these steps by using the knowledge of the partial derivatives (first and sometimes second order, depending on the algorithm) of the objective and constraints with respect to the variables (launch powers $\boldsymbol{p}$).

The first option to calculate such derivatives is to interface the optimization algorithm with a Qtool. In this case, the $SNR_n(\boldsymbol{p})$ values in the algorithm come from the Qtool calculations $Q_n(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{z})$. If the PLM used within the Qtool has closed form partial derivatives, then the optimization process is straightforward. However, typically the Qtools (e.g. GN model based PLM) do not have closed form derivatives. Then we can use a derivative identification subroutine based on finite differences. This subroutine makes changes in the launch powers and uses the Qtool (or PLM with design margin) to calculate the outcomes (connections' new SNR values). It is assumed that we use a Qtool that was aligned/trained once before the optimization, as discussed above, so we use the fitted parameters $\boldsymbol{r} = \boldsymbol{r_0}$. We will refer to this as *optimization with one-time trained Qtool.*

An alternative option is to probe the real network, that is, to interface the algorithm and, in particular, the derivative identification subroutine with the network, bypassing the PLM. In this case the $SNR_n(\boldsymbol{p})$ values in the algorithm come from the monitors of the network $Y_n(\boldsymbol{p})$. The derivative identification process would configure through the control plane the launch powers of the transponders, and would monitor the outcomes (connections' SNR values) to calculate the derivatives. This would be repeated at each algorithm's iteration. We will refer to this option as *optimization with monitoring probes.*

Note that the former option is fast. The Qtool is trained once and used thereafter to compute the derivatives. Although the Qtool is called several times, it has low computation complexity (at least the GN model [50]), resulting in low overall optimization time. However, this option suffers from accuracy issues. Specifically, several parameters such as the amplifier gain ripple, non-linear interference (NLIs), crosstalk at switches, etc., change for different network configurations/states. So, the one-time trained Qtool which is quite accurate at the beginning/initial state (**Figure. 6.2(a)** and **Figure. 6.2(b)**) behaves rather inaccurately as the iterative optimization algorithm projects the network into states that are away from the initial. The inaccuracy of the Qtool results in inaccurate estimation of the derivatives which in turn results in suboptimal optimization of the launch powers. This accuracy problem is

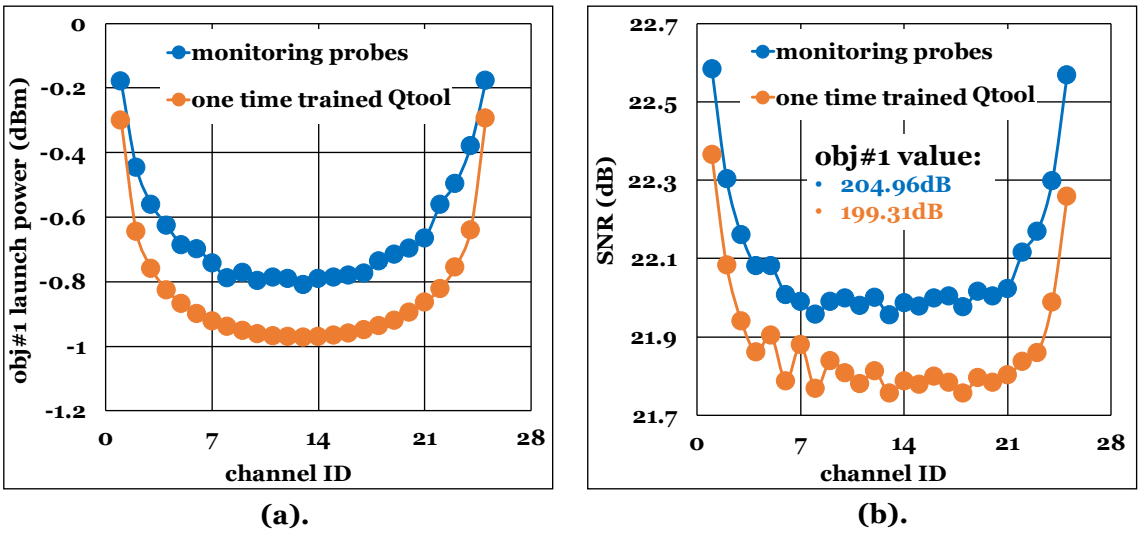


Figure. 6.3. (a) Optimized launch powers, and (b) corresponding SNR and obj#1 value, evaluated in the real network (VPI), for Case 1 (flat EDFAs).

expected to be more profound when the network physical layer is more dynamic, as in Case 2, where EDFA gain ripples result in SNR variations as the launch powers change.

On the other hand, the latter option, optimization with monitoring probes, involves several interactions with the actual network at each intermediate step, which typically take long time and is also susceptible to monitoring errors. Note that, the term monitoring probes refers to the capability of the network to change the launch powers and monitor the outcomes. In other optimization problems, e.g. involving establishment/ release of connections, such capability would probably not be present. Finally, note that in the results presented here and in Section 6.5 up to **Figure. 6.16**, the monitoring error was assumed to be zero. Thus, the results obtained with the monitoring probes and zero monitoring error are optimal and set as the reference for all other cases.

6.2.3 Deviation of Optimizing with the One-time Trained Qtool

Figure. 6.3(a) and **Figure. 6.3(b)** show the optimized launch powers for obj#1 with the one-time trained Qtool and the monitoring probes approaches for flat EDFAs (Case 1). We see that the one-time trained Qtool did not support well the optimization, since the algorithm using it identified quite different power levels. That is, although the algorithm using the one-time trained Qtool identified the optimum, this was optimum for the Qtool and not close to the optimum in the real network (VPI). The reason for this is that the Qtool could not follow/predict with good accuracy the real SNR values at the power levels calculated by the algorithm despite, the accuracy was very good for the initial state of the network, right after the (one time) training. A margin on the Qtool could cover this, but again would result in suboptimal calculations.

The maximized sum of SNR margins (obj#1) was optimized to 204.96 dB when the algorithm used monitoring probes and interacted with the real network, and to 199.31 dB when it interacted with the one-time trained Qtool. There exists a mismatch of ~5.6 dB in obj#1 value between these two optimization approaches. Note that the SNR values and the objective (**Figure. 6.3(b)**) were and should be evaluated in the real world (VPI), so

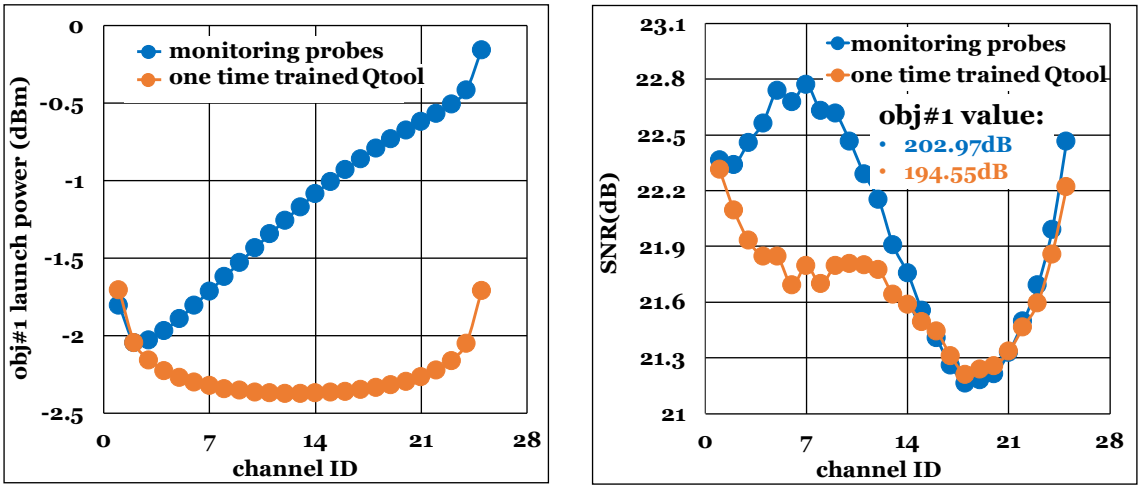


Figure. 6.4. (a) Optimized launch power, and (b) corresponding SNR and obj#1 value, evaluated in the real network (VPI), for Case 2 (EDFA with gain ripple of ±0.2 dB).

that we can see the deviation. This also explains the ripples in SNR seen in **Figure. 6.3(b)**, since VPI models some wavelength dependent factors not covered by the GN model. Similar behavior was observed for obj#2. Note that optimization with obj#2 results in choosing the launch powers that result in almost flat SNR values, since maximizing the minimum margin iteratively pushes the lowest SNR value and reduces the higher. The maximized minimum margin was optimized to 7.64 dB with monitoring probes, and to 7.36 dB with the one-time trained Qtool.

To emulate a more realistic scenario we assigned a gain ripple profile to all EDFAs having a peak-to-peak ripple value of ~±0.2 dB (Case 2). In such a scenario when the algorithm used the one-time trained Qtool, it reached an optimization objective (evaluated in the real network - VPI) quite worse than when it used monitoring probes and interacted with the actual network at intermediate optimization iterations. **Figure. 6.4(a)** and **Figure. 6.4(b)** shows the optimized launch powers and their corresponding SNR values respectively, for obj#1. A maximum input power difference of ~1.5 dBm was observed, resulting in ~8.4 dB of SNR difference for obj#1. Similarly, for obj#2, we observed a maximum input power difference of ~1.2 dBm, resulting in ~0.62 dB of SNR difference for obj#2. Note that the mismatch is higher than previously (case 1/flat EDFAs). This is because we now have a more volatile physical layer (EDFA gain ripples affect the SNR values) and the Qtool we use does not cover this additional volatility.

Concluding, for any optimization problem the Qtool accuracy is important. For planning/static problems, we cover inaccuracy issue with margin, while several papers have targeted the reductions of margin, by aligning the Qtool to the physical layer conditions e.g. through monitoring and ML as presented in Chapter 3. However, there have been limited discussions on dynamic optimization problems; the disadvantage is that to justify dynamic optimization we should target to achieve high efficiency, making the accuracy of the Qtool more critical. For complex/multivariable dynamic optimization tasks, such as the dynamic launch power optimization problem discussed above, an iterative algorithm is typically used that calculates several intermediate solutions. One

option is to interface the algorithm with the network to probe and monitor it to carry out the intermediate steps until it achieves the optimum. This, however, is cumbersome and very slow. On the other end, we can train the Qtool (by adjusting PLM parameters) before the optimization and use it in all intermediate calculations. Since Qtool calculations are fast the optimization will finish quickly. However, the accuracy of the Qtool can deteriorate and result to suboptimal optimization as seen in the preliminary results discussed above. This motivated us to address the limitations of the optimization with one-time trained Qtool by exploring the operating network and its feedback. Our goal is to appropriately realign the Qtool at intermediate optimization calculations, so that the difference between the optimization objective achieved with monitoring probes (interacting with the real-world) and with the retrained Qtool is negligible, while the whole optimization is much faster.

6.3 Network Dynamic Optimization and Qtool Retraining

Qtools (or PLMs), which can be analytical, semi-analytical, ML models, etc., have certain accuracy. The modeling assumptions impact the estimation accuracy. For example, many Qtools neglect EDFA gain ripples, partially model NLIs (e.g. consider full load), filters' crosstalk (inside ROADMS), residual dispersion, specific parameters of transponders, etc. Note that a detailed PLM (resulting Qtool) is slower in the calculations and requires more input parameters. Then, a second factor comes in play: the input parameters might not be known with good accuracy, which eventually reduces the accuracy of a detailed Qtool.

Optimization tasks result in network changes and typically use a Qtool to estimate the effect of such changes. However, these changes also modify the physical layer itself, they move the network to a new state. Depending on the PLM of the Qtool, such changes are covered to a certain degree. For example, when changing the power of a connection, NLIs, crosstalk, but also the penalties due to EDFAs' gain ripple profiles change. The PLM model of the Qtool could for example, cover the effect of NLIs and crosstalk, but not the evolution of gain ripples. For these reasons margins are used. However, in dynamic use cases the aim is to be more efficient and thus the accuracy of the Qtool is a crucial factor. To improve that we can take advantage of the operating network. Hence a basic need for dynamic optimization is to have a Qtool that follows the network changes. In the ML era, a way to do this is to choose an appropriate set of parameters and retrain the Qtool at certain points. However, retraining is cumbersome and thus we cannot retrain it before every dynamic task. On the other end, training the Qtool once before a multivariable optimization task can result in suboptimal optimization, since the accuracy of the Qtool deteriorates after several intermediate calculations.

In this section, we focus on dynamic multivariable optimization problems, and, in particular, we discuss the launch power optimization problem of established connections as introduced in the previous section. Note, however, that the proposed solution is generic and applicable to other dynamic simple or multivariable optimization problems as well. We propose to use an iterative closed control loop process to solve such dynamic multivariable optimization problems. At certain intermediate iterations of the algorithm we close the loop, configure the network and monitor to retrain the Qtool (with ML) to follow the projected network conditions. The target is to make the Qtool a digital replica,

that is, a DT, of the optical physical layer for the dynamic optimization task at hand. The frequency of the Qtool retraining depends on both the PLM model (used within the Qtool) and on the optimization task. As discussed in the previous section alternative options for the algorithm are to avoid using a Qtool and allow the algorithm to interact with (probe and monitor) the network or use a one-time trained Qtool. All three options are formally described in the following subsections.

6.3.1 Optimization with Monitoring Probes

The scheme that is considered in this subsection assumes that the optimization algorithm interacts directly with the actual network and follows a closed control loop process. The algorithm employs a subroutine to specify the probes, the configurations that are applied to the network. Then it monitors the outcomes to identify the information that it needs for an intermediate optimization step. A representation of this scheme is shown in **Figure. 6.5(a)**.

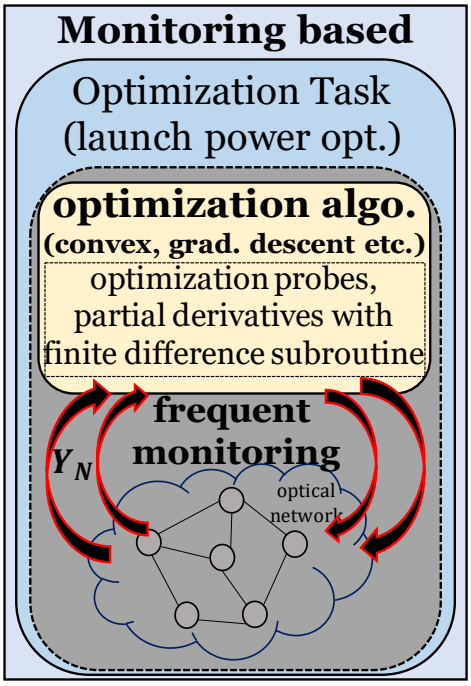
To be more specific, we focus on the dynamic launch power optimization problem with a typical objective such as maximizing the sum of SNR margins, or minimum margin, as discussed in Section 6.2. This problem is known to be convex and of polynomial complexity. The convex optimization algorithms, such as (sub)gradient methods, interior point, trust-region-reflective etc., are iterative; at each iteration they need to calculate Jacobians and/or Hessians for the objective and constraint functions [136], [137]. Actually, the related algorithms are classified into first or second order depending on the order of the partial derivatives they use. For optimization problems that involve PLMs (that resides inside Qtool) without closed form partial derivatives a way to calculate them is to use a subroutine that implements finite differences [137].

For example, for the power optimization problem at hand, to calculate the gradient for an objective function f we need to find/monitor the changes in the SNR values of all connections, assumed to be done through the coherent receivers, with respect to changes in the powers of the transponders. To give an example, assuming a network with a set of N established connections with a launch power vector $\boldsymbol{p}$. We denote by $\boldsymbol{\delta p_n}$ the vector with all zeros apart from element n whose value we set to p_{step}, what we refer to as the *power probe step*. As a matter of fact, the change in launch power of the single connection n results in changes in the SNR values of all interfered connections (those sharing a common link). So, we denote by $\boldsymbol{SNR_N(p)}$ and $\boldsymbol{SNR_N(p + \delta p_n)}$ the SNR vector of all the N connections for the respective power vectors. If f is the objective function, then the first order partial derivative for n is given by

$$g_n = f(\boldsymbol{p}) - f(\boldsymbol{p} + \boldsymbol{\delta p_n})/p_{step} \tag{6.4}$$

Depending upon the function f this involves certain operations with the vectors $\boldsymbol{SNR_N(p)}$ and $\boldsymbol{SNR_N(p + \delta p_n)}$. The gradient $\boldsymbol{g}$ is the vector of all partial derivatives, that is, g_n for all n. To calculate the gradient with the finite difference method we need to probe with $\boldsymbol{p} + \boldsymbol{\delta p_n}$ and monitor $\boldsymbol{SNR_N(p + \delta p_n)}$, and repeat this probe/monitoring process for all connections n = 1,2,..., N.

Generalizing this, the optimization algorithm calculates (first or second order) partial derivatives through a finite differences subroutine at each intermediate iteration. Let us



(a).

Pseudo-code – 6.1

Start with initial launch power vector $\boldsymbol{p_0}$, iteration number i=-1
While not converged
  Increase i
  Finite differences process, time: t_{mon} per probe to monitor all connections $\boldsymbol{Y_N}$
    Probe d_i times (configure new launch powers and monitor)
    (e.g. for the first order partial derivative of connection n, probe with $\boldsymbol{p_i + \delta p_n}$ and monitor $\boldsymbol{Y_N(p_i + \delta p_n)}$
  Calculate the derivates and next launch power vector $\boldsymbol{p_i}$, time: t_{calc}
  Evaluate convergence (e.g. compare objective improvement with a threshold), when converged: $L_{mon\_prob} = i$
  Apply the calculated power vector p_i to the network

(b).

Figure. 6.5. (a) Optimization with actual deployed monitors (monitoring probe based approach), and (b) Pseudo-code for optimization with monitoring probes.

assume that the algorithm calls the finite differences method d_i times at iteration i. For the above example with the gradient, we have $d_i = N$. This is the simplest case; we typically have $d_i \geq N$ depending on the algorithm. Note also that we might have different number of probes per iteration, that is different d_i per i. However, to simplify our analysis we assume that this is constant, $d_i = D$, for each iteration i.

The convex optimization algorithm with monitoring probes performs L_{mon_prob} iterations to find the optimum. We also denote by t_{mon} the monitoring time, assumed here to monitor simultaneously all N established connections. The monitoring time can range from minutes to hours, depending upon the network size, the monitoring plane, the targeted monitoring error etc. [138]. However, once the monitoring information is forwarded to the algorithm, the time t_{calc} to calculate the gradients/ Hessian and also the next launch powers is quite lower (few milliseconds range) compared to the monitoring time ($t_{mon} \gg t_{calc}$). So, with the monitoring probes-based approach, under the assumption that N connections are monitored in parallel, the total optimization time T_{mon_prob} is given by:

$$T_{mon\_prob} = L_{mon\_prob} \,.\, (D \cdot t_{mon} + \ t_{calc}) \approx L_{mon\_prob} . D . t_{mon} \qquad (6.5)$$

Optimizing with monitoring probes is described with pseudo-code 6.1 in **Figure. 6.5(b)**.

In general, the optical monitors have certain measuring accuracy. Herein, we assume that the monitored SNR at the coherent receivers is quite accurate. Note that higher accuracy can be achieved through time averaging; to reduce the effect of short term time impairments, e.g. polarization, the monitoring measurements could be averaged over

time resulting in higher accuracy but also higher monitoring time. Depending on the monitoring error, we might end up to a different and worse objective value instead of the optimum. Another aspect to be accounted for in a real network is that the power probe steps (p_{step}) cannot be very small because fine-tuning of the equipment is not feasible. Thus, in a real network, there are two factors that hinder the monitoring probe optimization process:

(i) the monitoring errors

(ii) the minimum power probe step p_{step} that can be configured

We call the SNR vector obtained from monitors with errors as the noisy monitored vector, and denote it by

$$\widetilde{\boldsymbol{Y}}_{\boldsymbol{N}}(\boldsymbol{p}) = \boldsymbol{Y}_{\boldsymbol{N}}(\boldsymbol{p}) + \boldsymbol{v} \tag{6.6}$$

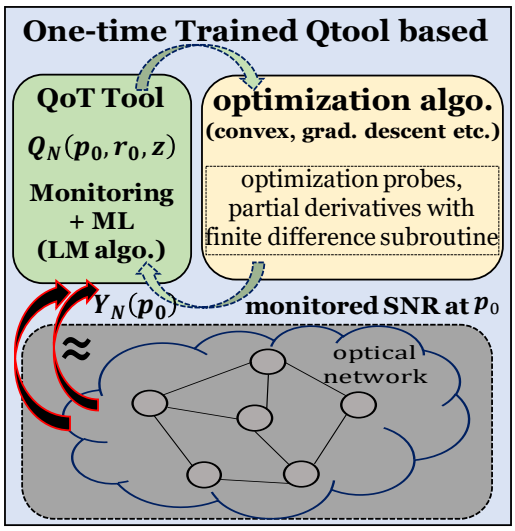
where, $\boldsymbol{v}$ is a vector that represents the monitoring error (or noise).

Stochastic subgradient methods [139] for zero mean errors provably find the optimum solution with specific step sizes but might require a very large number of iterations. However, in a real network where the use of small steps is not supported by the transponders and iterations are expensive since they involve several monitoring phases, such methods are hardly applicable.

In this monitoring probes optimization approach the algorithm optimizes the launch powers and checks at each step the actual conditions of the network. For zero error this approach identifies the optimum obj_{mon_prob} but requires a high optimization time T_{mon_prob}. So, we will use this as the reference for all other approaches. Also note that the monitoring probes, which are used in this method to identify the partial derivatives, is not a universal solution. A monitoring probe in the studied use case refers to the configuration of new launch power(s) to one (or more) transponders of the established connections and monitoring of all connections SNRs at their receivers. So, the definition is specific to the problem; different optimization problems require different monitoring probes definitions. For some tasks, monitoring probes might not be available, e.g., tasks involving the establishment/release of connections. For such tasks, we might need spare transponders to extract the information required for the optimization [11], which imply higher cost and complexity.

6.3.2 Optimization with One-time Trained Qtool

In this subsection, we consider the method where the QoT estimation tool is aligned only once, at the beginning of optimization. We perform the alignment of the Qtool using monitoring information $\boldsymbol{Y}_{\boldsymbol{N}}(\boldsymbol{p_0})$ from the actual network, assumed to take time t_{mon} as above. We then use ML to fit the parameters $\boldsymbol{r}$ of the PLM $\boldsymbol{Q}_{\boldsymbol{N}}(\boldsymbol{p_0}, \boldsymbol{r}, \boldsymbol{z})$ to the physical layer conditions, so as to identify $\boldsymbol{r_0}$. This is assumed to take time t_{train}. Then the optimization algorithm interacts with this one-time trained PLM, $\boldsymbol{Q}_{\boldsymbol{N}}(\boldsymbol{p_0}, \boldsymbol{r_0}, \boldsymbol{z})$, at each intermediate step, to estimate the QoT (SNR vector) of the connections. The detailed schematic of the scheme is shown in **Figure. 6.6(a)**. Since there are no closed form derivatives equations for the GN model, we use a similar derivatives identification subroutine (finite differences), as in the previous method/subsection, but this time we



(a).

Pseudo-code – 6.2

Start with initial launch power vector $\boldsymbol{p_0}$, iteration number i=-1
Align Qtool to initial / prior-to-optimization state (monitor $\boldsymbol{Y_N(p_0)}$ and train the Qtool $\boldsymbol{Q_N(p_0, r, z)}$ to find $\boldsymbol{r_0}$)
time: $t_{mon} + t_{train}$
While not converged
  Increase i
  Finite differences process, time : t_{Qtool} per SNR vector calculation by the one-time trained Qtool $\boldsymbol{Q_N}$
    Probe d_i times the Qtool: change the launch powers and calculate the SNR vector for all connections with the Qtool (e.g. for the first order partial derivative of connection n, set $\boldsymbol{p_i + \delta p_n}$ and calculate $\boldsymbol{Q_N(p_i + \delta p_n, r_0, z)}$),
  Calculate the derivates and next launch power vector $\boldsymbol{p_i}$, time: t_{calc}
  Evaluate convergence (e.g. compare objective improvement with a threshold), when converged: $L_{Qtool} = i$
Apply the last calculated power vector $\boldsymbol{p_i}$ $(= \boldsymbol{p_{L_{Qtool}}})$ to the network

(b).

Figure. 6.6. (a) Optimization with one-time trained Qtool at $Y_N(p_0)$, and (b) Pseudo-code for optimization with one-time trained Qtool.

interface that with the one-time trained Qtool instead of the actual network. We denote by t_{Qtool} the time that the Qtool takes to calculate the SNR values of all connections. As before, this subroutine is assumed to be called D times at each algorithm intermediate iteration. We assume that the time t_{calc} that the algorithm needs to calculate the gradients/Hessian and also the next launch powers is the same as the previous method. We also denote by L_{Qtool} the number of iterations that the algorithm performs. With the one-time trained Qtool, the total optimization time T_{Qtool} is given by

$$T_{Qtool} = t_{mon} + t_{train} + L_{Qtool}.\left(D.t_{Qtool} + t_{calc}\right) \tag{6.7}$$

It stands to reason that the Qtool training and estimation calculations and the algorithm calculations are substantially faster than monitoring ($t_{mon} \gg t_{train}, t_{Qtool}, t_{calc}$). We also expect a similar number of iterations ($L_{Qtool} \approx L_{mon\_prob}$), because the PLM (GN model) based Qtool satisfies the convexity properties [109]. So, we have $T_{Qtool} \approx t_{mon}$. By comparing this to Eq. (6.5) we can easily notice that the one-time trained Qtool based optimization approach requires substantially less optimization time than the previous approach as

$$T_{Qtool} \approx t_{mon} \ll T_{mon\_prob} \approx L_{mon\_prob}.D.t_{mon} \tag{6.8}$$

In particular, the speedup we obtain is in the order of $L_{mon\_prob} \cdot D$. This happens because the one-time trained Qtool approach provides all the necessary information very fast for the optimization algorithm at each intermediate step, eliminating frequent monitoring requirement. Optimizing with a one-time trained Qtool is described with pseudo-code 6.2 in **Figure. 6.6(b)**.

The optimization algorithm using the one-time trained Qtool approach identifies the launch powers that yield the optimum $\widetilde{obj}_{Qtool}$, but this is viewed through the one-time

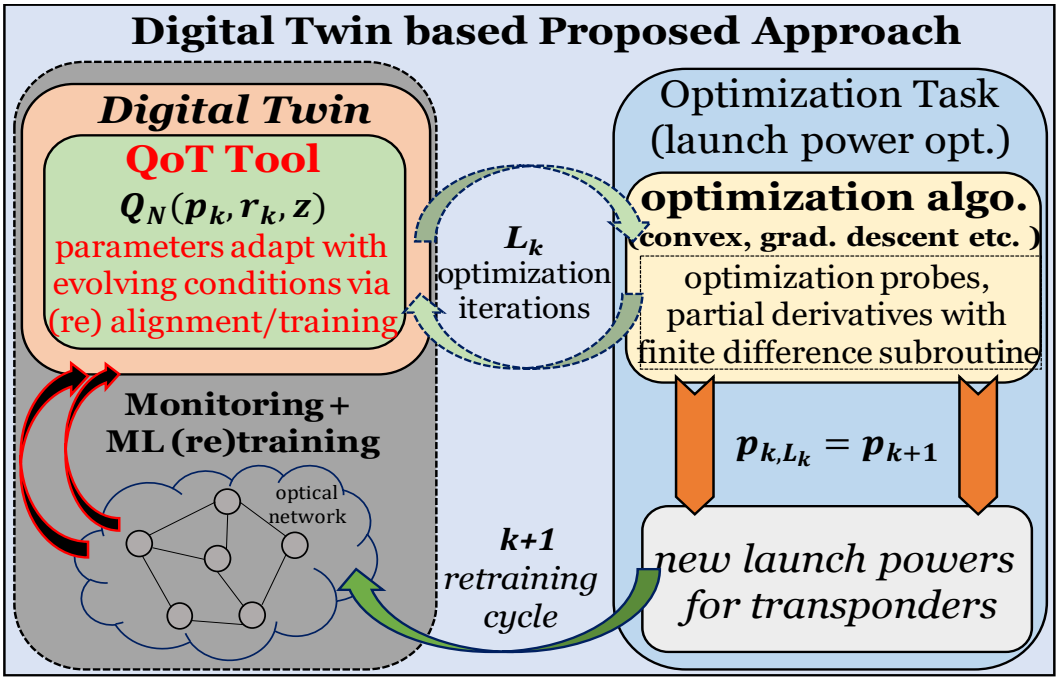


Figure. 6.7. Schematic of proposed QoT tool retraining or digital twin based approach for launch power optimization.

trained Qtool. However, the Qtool has certain accuracy (even after proper training), and was trained at initial conditions. So the identified launch powers yield the objective obj_{Qtool} in the real network, which is worse than the objective of the monitoring probe method being always evaluated in the real network, $obj_{Qtool} \leq obj_{mon\_prob}$. This problem was identified in Section 2 and shown in **Figure. 6.3** and **Figure. 6.4**.

6.3.3. Proposed Solution - Optimization with a Digital Twin

Following the discussions of the two above approaches in last two subsections, and the results presented in Section 6.2, we observe a clear tradeoff between optimization time and performance. The monitoring probes-based approach (in **Figure. 6.5(a)**) implements closed control loops which are not fast, due to the complex probing and slow monitoring subroutine. However, it achieves the real optimal as it tracks the network evolving conditions/states by configuring and monitoring. On the other hand, the one- time trained Qtool approach (in **Figure. 6.6(a)**) is substantially faster since it uses the one-time trained Qtool to quickly find the derivatives at intermediate states. However, the Qtool is trained only once, at the beginning of the optimization. Thereby, if the algorithm projects the network to substantially different physical conditions, then the optimization is suboptimal, since the Qtool differs from reality, as seen in **Figure. 6.3** and **Figure. 6.4**. The following proposed scheme keeps the benefits of both approaches: it finds a near to optimal solution, but with an overall low optimization time.

We propose to use an *iterative* closed control loop process to solve the dynamic optimization problem. At certain intermediate iterations of the algorithm, we configure the intermediate solution to the network, and monitor to realign the Qtool with ML to follow the projected network conditions. The detailed schematic of the scheme is

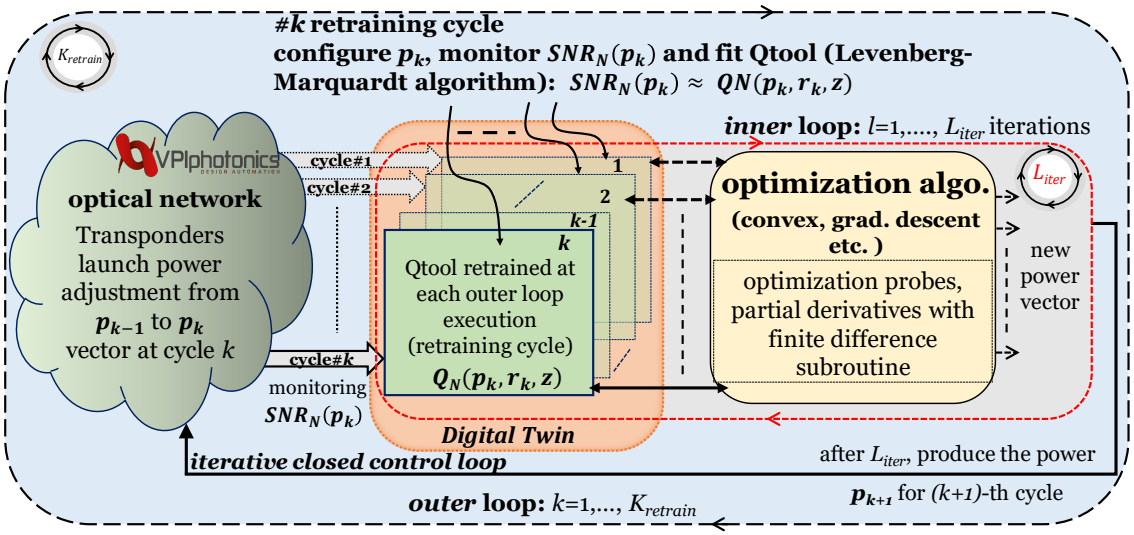


Figure. 6.8. Schematic showing the two nested for loops, outer for the Qtool retraining cycles and the inner for the optimization algorithm iterations.

depicted in **Figure. 6.7**. The idea is to make the Qtool a DT, to have a PLM (inside Qtool) model which is parametric and define the method to readjust/realign it to represents the physical system with enough accuracy to perform the dynamic optimization calculations at hand. For realigning the Qtool, many techniques can be used. We here rely on ML training. In this study we used a Qtool with PLM based on the GN model [50], which considers the launch powers and wavelength occupancy. Thus, it models quite accurately linear and NLI transmission impairments. We also extended it and added a wavelength dependent penalty on top of the GN SNR calculation to cover e.g., the EDFA ripple penalties [16]. The GN alignment process was described in Section 6.2., and extended here to be performed iteratively.

As above, we denote by $\boldsymbol{Q_N(p, r, z)}$ the calculation of the SNR values vector of all N connections by the Qtool with GN based PLM, where $\boldsymbol{p}$ is the launch power vector (optimization variables), $\boldsymbol{r}$ represents the Qtool (more specifically PLM) fitted parameters, the fiber coefficients and the wavelength dependent ripple penalty, and $\boldsymbol{z}$ represents the unchanged input parameters for our optimization such as routes, used wavelengths, etc. The dynamic optimization process starts with the configured launch power vector $\boldsymbol{p_0}$ of the established connections (e.g., all 0dBm). For this initial power vector $\boldsymbol{p_0}$ the Qtool is trained with the monitored SNR vector $\boldsymbol{Y_N(p_0)}$. To be more specific, we use ML and in particular the Levenberg-Marquardt (LM) algorithm to find $\boldsymbol{r_0} = \text{argmin}_r\left(\boldsymbol{Q_N(p_0, r, z)} - \boldsymbol{Y_N(p_0)}\right)^2$. The details about the algorithm are already stated in Chapter 2, Chapter 5 and in Section 6.3 of this Chapter also.

Now, let us assume that at the end of the k-th Qtool training cycle, the optimization algorithm has performed L_k intermediate iterations and identified the launch powers $\boldsymbol{p_k^{L_k}}$. We then start the next cycle $k+1$ by configuring the network with the outcome so $\boldsymbol{p_{k+1}} = \boldsymbol{p_k^{L_k}}$. To retrain the Qtool for the $k+1$ cycle we configure the network with $\boldsymbol{p_{k+1}}$ and monitor to obtain $\boldsymbol{Y_N(p_{k+1})}$. Then ML is used to fit the parameters $\boldsymbol{r_{k+1}}$, that is $\boldsymbol{r_{k+1}} = \text{argmin}_r(\boldsymbol{Q_N(p_{k+1}, r, z)} - \boldsymbol{Y_N(p_{k+1})})^2$. This retrained Qtool is then used in the

Pseudo-code – 6.3

Start with initial powers $\boldsymbol{p}_0$, outer loop iteration number k=0
While not converged
  Align the PLM to current network state (monitor $\boldsymbol{Y_N(p_k)}$ and train the Qtool $\boldsymbol{Q_N(p_k, r, z)}$ to identify $\boldsymbol{r_k}$, time: $t_{mon} + t_{train}$
  Increase k, $\boldsymbol{p_k^0 = p_{k-1}}$
  For l=0,..., L_{iter}-1(Inner loop iterations)
    Finite differences process, time : t_{PLM} per SNR vector calculation by the Qtool $\boldsymbol{Q_N}$
      Probe d_i times the PLM: change the launch powers and calculate the SNR vector of all connections with the trained Qtool
      (e.g. for the first order partial derivative of connection n, set $(\boldsymbol{p_k^l + \delta p_n})$ and calculate $\boldsymbol{Q_N(p_k^l + \delta p_n, r_k, z)}$
    Calculate the derivates and next power vector $\boldsymbol{p_k^{l+1}}$, time: t_{calc}
  Evaluate convergence (e.g. compare objective improvement with a threshold), when converged: $K_{retrain} = k$
  Apply the calculated power vector $\boldsymbol{p_k^{L_{iter}}}$ to the network

Figure. 6.9. Pseudo-code for optimization with the proposed QoT tool retraining or Digital Twin based scheme.

optimization algorithm iterations of cycle $k + 1$. Note that, at each retraining cycle of the Qtool, we can make use of the previously monitored SNR vectors, including thus the history, the network evolution conditions. This tends to improve the Qtool accuracy as the algorithm iterates, where the accuracy is more critical.

We assume that in total we retrain $K_{retrain}$ times the QoT tool. Although we can have different number of algorithm iterations per cycle, to simplify our analysis in the following we assume that the algorithm runs L_{iter} iterations after each re-training cycle of the QoT tool. So, $L_k = L_{iter}$ for all retraining cycles $k = 1,2,..,K_{retrain}$. It is easy to visualize and understand the overall concept as two nested for loops, as shown in **Figure. 6.8**. The outer one pertains to the retraining of the QoT tool, and the inner to the optimization algorithm intermediate iterations with the retrained Qtool or the DT. The time for each retraining cycle is equal to T_{Qtool} for L_{iter} iterations, that is $t_{mon} + t_{train} + L_{iter} \cdot (D \cdot t_{Qtool} + t_{calc})$. We denote, the overall optimization time with this DT based approach as T_{DT}, which is given by

$$T_{DT} = K_{retrain} \cdot \left(t_{mon} + t_{train} + L_{iter} \cdot \left(D \cdot t_{Qtool} + t_{calc} \right) \right) \qquad (6.9)$$

The proposed method of optimizing with a DT is described with pseudo-code 6.3 in **Figure. 6.9**.

Note that in total the optimization algorithm performs $K_{retrain} \cdot L_{iter}$ iterations, and retrains the Qtool $K_{retrain}$ times. Our target is to choose the retraining period L_{iter} appropriately so that the Qtool would follow with good accuracy the physical layer in the algorithm's intermediate calculations. If this is achieved, the Qtool estimated objective that is calculated at each iteration and the final one $\widetilde{obj}_{DT}$ would be very close to the real value in the real network obj_{DT}. Also, the achieved objective would be very close to the

optimum, as calculated by the monitoring probes method obj_{mon_prob}. So, we would have $\widetilde{obj}_{DT} \approx obj_{DT} \approx obj_{mon\_prob}$. Moreover, for an appropriate retraining period the iterations of the optimization algorithm would be close to those of the monitoring probes, that is $K_{retrain} \cdot L_{iter} \approx L_{mon\_prob}$. Looking at the total optimization times, and assuming that t_{mon} is the dominant factor, we have $T_{DT} \approx K_{retrain} \cdot t_{mon}$. Thus we obtain a speed-up of $L_{mon\_prob} \cdot D/K_{retrain} = L_{iter} \cdot D$ with respect to the monitoring probes optimization approach.

In the above presented proposed scheme, one of the most crucial factor to determine is the algorithm runs L_{iter} after each retraining cycle of the Qtool. In the previous paragraphs, we started with a fixed value of algorithm iterations after retraining Qtool for all k cycles, so

$$L_{k+1} = L_k = L_{fixed}, for\ all\ k = 1,2, \dots, K_{retrain} \tag{6.10}$$

However, we also devise an improved scheme in order to choose adaptive number of algorithm iterations. The main parameter to decide the number of iterations is based on the amount of Qtool alignment error at current training cycle $k+1$, which we denote as e_{k+1} and is given by

$$e_{k+1} = \|\boldsymbol{Y_N}(\boldsymbol{p_{k+1}}) - \boldsymbol{Q_N}(\boldsymbol{p_{k+1}}, \boldsymbol{r_k}, \boldsymbol{z})\|^2 \tag{6.11}$$

Note that in the above Eq. (6.11), the alignment error of the Qtool is calculated from the Qtool trained (with $\boldsymbol{r_k}$) at the start of the previous k cycle. We here decide the adaptive number of iterations $L_{k+1} \neq L_{fixed}$ at next cycle $k+1$ using an adaptive function A of the Qtool alignment errors of previous k and current $k+1$ cycle as

$$L_{k+1} = L_{iter} \neq L_{fixed} = A(e_{k+1}, e_k, L_k) \tag{6.12}$$

The idea is to track the evolution of the alignment of the Qtool and reduce/increase the realignment period accordingly. In particular, in the results section, we implemented a simple adaptive function A where we increased/decreased L_k (of previous cycle, k) by 5 (for the next cycle, $k+1$) if we observed a decrease/increase of the alignment error.

6.4 Results and Discussions

To quantify the benefits of the devised DT based power optimization approach, we carried out simulations using both VPI Transmission Maker [14] and MATLAB. The actual network was implemented in VPI, and the Qtool (relying on the GN model-based PLM) and the convex optimization algorithm were developed in MATLAB. This choice was made to capture the mismatch between the real network and the Qtool used in the optimization process. This is a considerable improvement in terms of realism compared to many prior studies (listed in Chapter 3), where authors used the same QoT tool for both the real network/ground truth and their proposed solution.

To be more specific, we implemented in MATLAB the GN model and the launch power optimization algorithm to maximize: (obj#1) the sum of SNR margins, or (obj#2) the lowest margin, as discussed in Section 6.2. This optimization problem is known to be convex and of polynomial complexity. Hence, we implemented an interior-point

algorithm to solve it. The algorithm was run until it found the optimum (optimality tolerance 10^{-6}). The GN model was interfaced with the optimization algorithm and both were integrated in an automated system in VPI. For each simulation, VPI implements the outer loop (Qtool retraining cycle). It takes as input the launch powers coming from the algorithm of the integrated MATLAB module, performs the detailed transmission simulations and calculates the SNR vector of the channels. These are passed as input to the integrated MATLAB module. With that input the integrated Qtool gets trained and this trained Qtool is then used by the convex algorithm for L_{iter} intermediate iterations. In those iterations the algorithm uses the Qtool to identify the partial derivatives, using the finite differences subroutine, and then the new launch powers. After the L_{iter} iterations (inner cycle), a new set of launch powers are automatically fed to VPI transponders as a closed control loop for the next retraining cycle. Note that, at each retraining cycle of the Qtool, we retrained with the current and previously monitored SNR vectors, including thus the history, the network evolution conditions.

6.4.1 Improvement with Fixed L_{iter}

In this subsection, we will present all the results with fixed number of algorithm iteration L_{iter} after each retraining cycle as indicated by Eq. (6.10). The monitoring probes approach (**Figure. 6.5(a)**) was used as reference in this work. To implement this, we implemented another (more frequent) closed control loop without using a Qtool: the monitoring probes from the finite differences subroutine (in MATLAB) were carried directly to VPI and the SNR values were then passed back to that subroutine. The one-time trained Qtool approach (**Figure. 6.6(a)**) represents the traditional optimization scheme via a Qtool. For that, we train the Qtool only once with the SNR data from VPI at the beginning of the simulation and used that Qtool for the power optimization. Note that in this case, all the intermediate iterations are calculated by the one-time trained Qtool but they are not configured in the network. Only the final launch power vector at the convergence is configured in the networks and corresponding objective function values are determined. Note that in all cases we start by assigning 0dBm uniform power to all transponders in VPI.

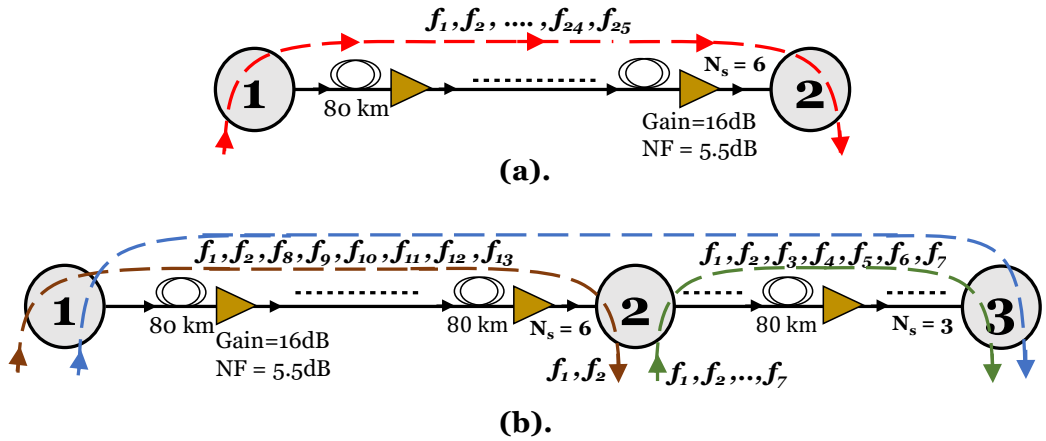


Figure. 6.10. VPI setup with (a) single link of 6 identical spans, and (b) 3 nodes and 15 connections with different paths/routes, added/dropped points to emulate a small network.

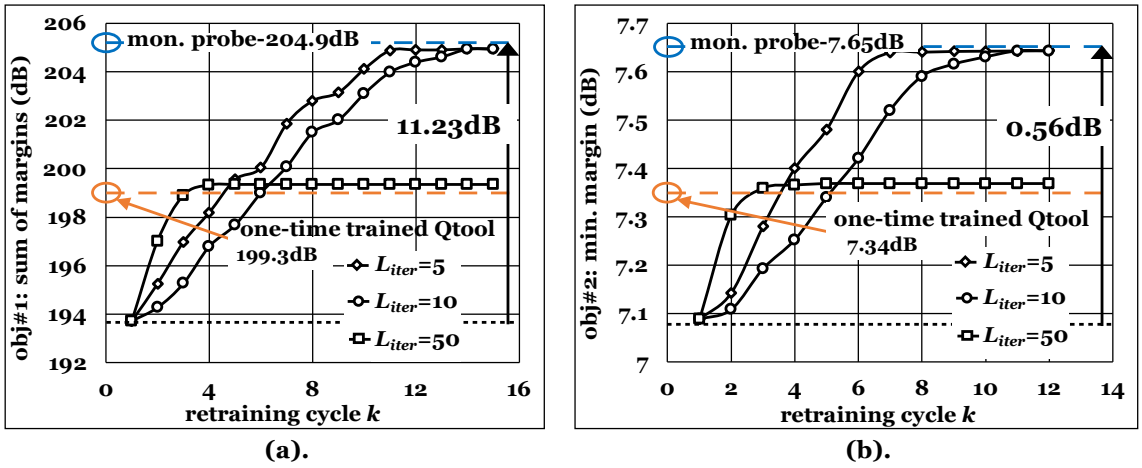


Figure. 6.11. (a) Obj#1, and (b) Obj#2 values as a function of the proposed DT approach retraining cycles (*k*), for flat EDFAs.

For all the optimization schemes, the objective value was calculated in VPI, so in the real network. As discussed, there exists a difference between the view of the Qtool/optimization process that uses it and the real objective. Finally note that we evaluated the benefits of our proposed scheme for relatively small channel count (=25) and up to two links, due to the slow execution time of VPI (split-step Fourier simulations). Actually, this can be considered as an indication of the long time of interacting with/monitoring the real network.

To be specific, we made two fully automated setups in VPI with

(i) single link of 6 identical spans with 25 channels (**Figure. 6.10(a)**)

(ii) two links with 15 channels which were added/dropped at the intermediate node (**Figure. 6.10(b)**)

For the first setup, 25 WDM channels with polarization-multiplexed 16-QAM modulation format at 32 Gbaud, leading to 200 Gb/s datarate per channel were launched. We assumed SNR threshold of 13.9 dB [107]. The wavelength spacing between the channels was assumed to be 50 GHz. We started with uniform 0 dBm of launch powers for all channels whose SNR values for each channel were measured/monitored by VPI. As stated above, VPI acted as the real-world/ground truth. Approximately 1dB of maximum SNR difference between the untrained Qtool and VPI was found, after setting equal the fiber parameters (dispersion coefficient, slope, attenuation coefficient of fiber, non-linearity coefficient etc.). Then with ML training (using the Levenberg-Marquardt algorithm), the SNR mismatch was reduced to less than 0.1dB (**Figure. 6.2** of Section 6.2). This one-time trained Qtool was then used with the power optimization algorithm. The algorithm converged to 199.31 dB and 7.34 dB for obj#1 and obj#2, depicted with the orange circles and dotted lines in **Figure. 6.11(a)** and **Figure. 6.11(b)**, respectively.

We then optimized with the monitoring probes where the optimization algorithm was interfaced with the actual network (VPI). Again, the algorithm was run until it found the optimum for the objective function at hand. The algorithm converged to 204.96 dB and

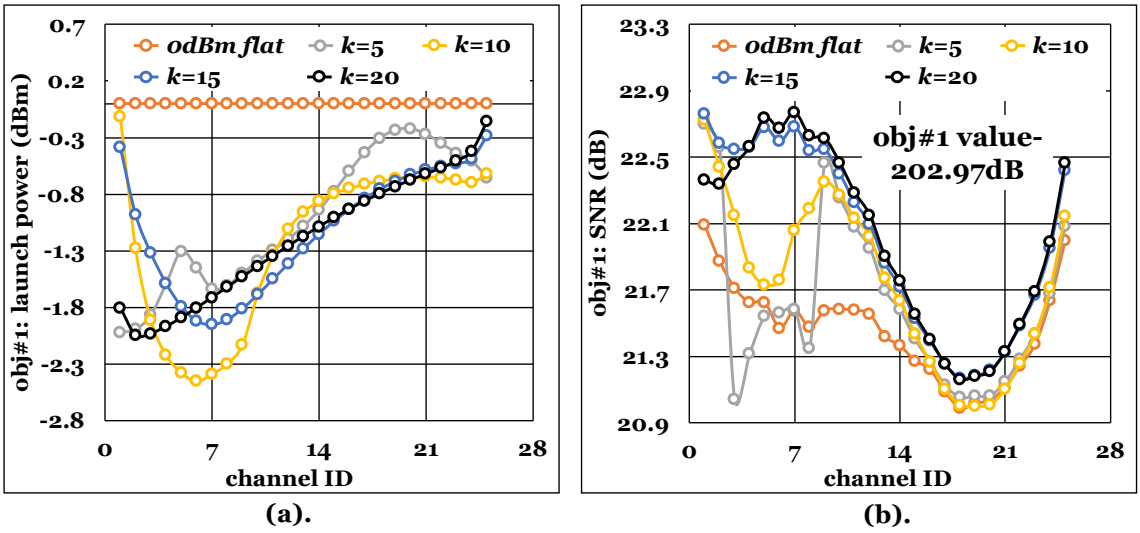


Figure. 6.12. (a) Launch power (dBm), and (b) SNR (dB) per channel as the number k of retraining cycles increase, for the proposed DT approach, $L_{iter} = 5$, obj#1 and EDFAs having peak-to-peak gain ripple of 0.4 dB.

7.64 dB for obj#1 and obj#2, depicted with the blue circles and dotted lines in **Figure. 6.11(a)** and **Figure. 6.11(b)**, respectively. A relatively mismatch of ~5.6 dB for obj#1 and ~0.3 dB for obj#2 between the monitoring probes (optimum) and the one-time trained QoT tool based approach was observed. With monitoring probes, the algorithm took around $L_{mon\_prob} \approx 120$ iterations to optimize the launch powers in both objectives. In case of the one-time trained Qtool, the algorithm converged a bit faster, after $L_{Qtool} \approx 90$ iterations.

We then optimized with our proposed Qtool retraining/DT approach. We examined different (but fixed) Qtool retraining periods of L_{iter}= 5, 10 and 50. **Figure. 6.11**. shows the optimization objective values evaluated in VPI as a function of the retraining cycles (index k) of our proposed scheme. We see that with each training cycle, the objective value moves towards the optimum/reference obtained with the monitoring probes approach. Note that L_{iter} are kept fixed after each retraining cycle as defined by Eq. (6.10) We also tried our approach with the adaptive number of L_{iter} given by Eq. (6.11) and Eq. (6.12), and noticed considerable improvements compared to $L_{iter} = L_{fixed}$. For the sake of conciseness, in this section, we only present the results from the approach that rely on Eq. (6.10) (independent of realignment error) in order to decide L_{iter}, that is $L_{iter} = L_{fixed}$. However, later in this section, we provide the additional benefits of adaptive number of L_{iter} as discussed in Eq. (6.11) and Eq. (6.12) on a 4 node topology. But from now onwards (till next two subsections), we only present and discuss about the results obtained with fixed L_{iter}.

For L_{iter}=5, after approximately $k = 11$ iterations for obj#1 and $k = 8$ iterations for obj#2, the objective becomes nearly constant indicating convergence as shown in **Figure. 6.11**. For small retraining periods, such as $L_{iter} = 5, 10$, the objective (both for obj#1 and obj#2) reached exactly that achieved with the monitoring probe-based approach, that is, $obj_{DT} = obj_{mon\_prob}$. However, for longer retraining periods, such as

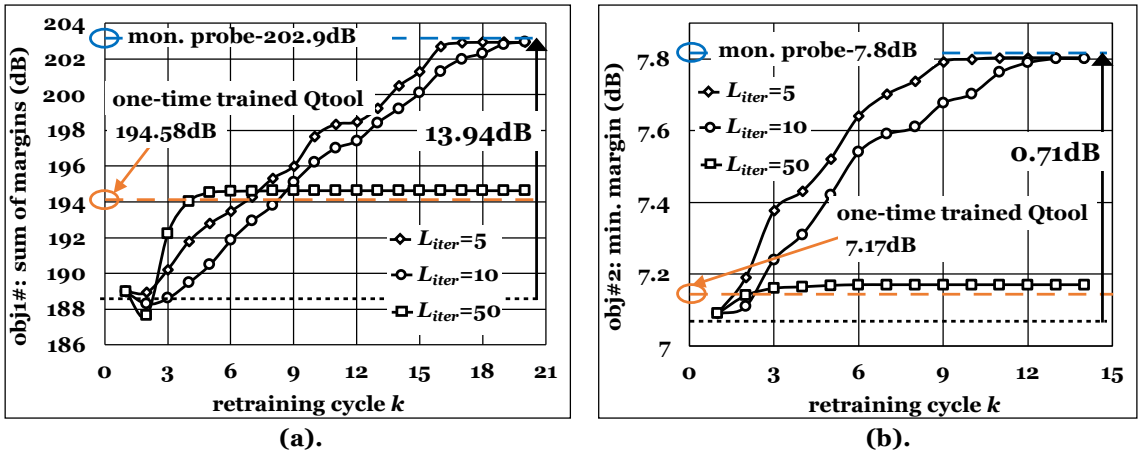


Figure. 6.13. (a) Obj#1, and (b) Obj#2 values as a function of the proposed DT approach retraining cycles (*k*), for EDFAs with peak to peak gain ripple of 0.4 dB.

$L_{iter} = 50$, the algorithm converged near to the one-time trained Qtool scheme. This happened because we allowed the optimization algorithm to perform long intermediate iterations without retraining the Qtool. Within these iterations the algorithm converged to an optimum, it could not improve its objective function ($\widetilde{obj}_{DT}$) with the used Qtool. However, after these long intermediate iterations, the Qtool was not representing reality with good accuracy, and thus the corresponding real network objective (obj_{DT}) was rather suboptimal.

In reality, EDFA gains are not flat and come with ripples. We assigned a gain ripple profile of peak-to-peak gain of 0.4 dB (±0.2 dB) to span EDFAs. The Qtool was then trained at 0 dBm of flat launch power and a maximum SNR difference of less than ±0.05 dB was observed as shown in **Figure. 6.2(b)** - Section 6.2. This led to a mismatch of ~8.5 dB (**Figure. 6.4(b)**) and ~0.64 dB in SNR margin between monitoring probes and one-time trained Qtool based optimization for obj#1 and obj#2, respectively. The mismatch is higher than previous, due to the EDFAs gain ripples that make the physical layer more dynamic. **Figure. 6.14(a)** shows the power per channel (in dBm) at different retraining cycles k for a retraining period $L_{iter} = 5$ and for obj#1. **Figure. 6.14(b)** shows the corresponding SNR (dB) values.

For obj#1, with around $k = 20$ Qtool retraining cycles, the transponders launch power and SNR values converged near the optimal values obtained with the monitoring probe-based approach (**Figure. 6.4**). The algorithm reached obj#1 = 202.96 dBs as shown by black curve in **Figure. 6.13(a)**, very close to the monitoring probe-based approach. Compared to the one-time trained Qtool it achieved an improvement of ~8.4 dB, which is ~0.34 dB of SNR improvement per channel.

Similar behaviour was also observed for obj#2. **Figure. 6.14(a)** and **Figure. 6.14(b)** show the input power (dBm) and the corresponding SNR (dB) for k retraining cycles with $L_{iter} = 5$ and for obj#2. From Figure. 6.13.(b) it can be observed that after $K_{retrain} = 12$ retraining cycles, the algorithm reaches obj#2 = 7.8 dB which is very close to that found with the monitoring probes approach. Compared to the one-time trained Qtool approach, we achieved an improvement of ~0.64 dB. **Figure. 6.13(a)** and **Figure. 6.13(b)**

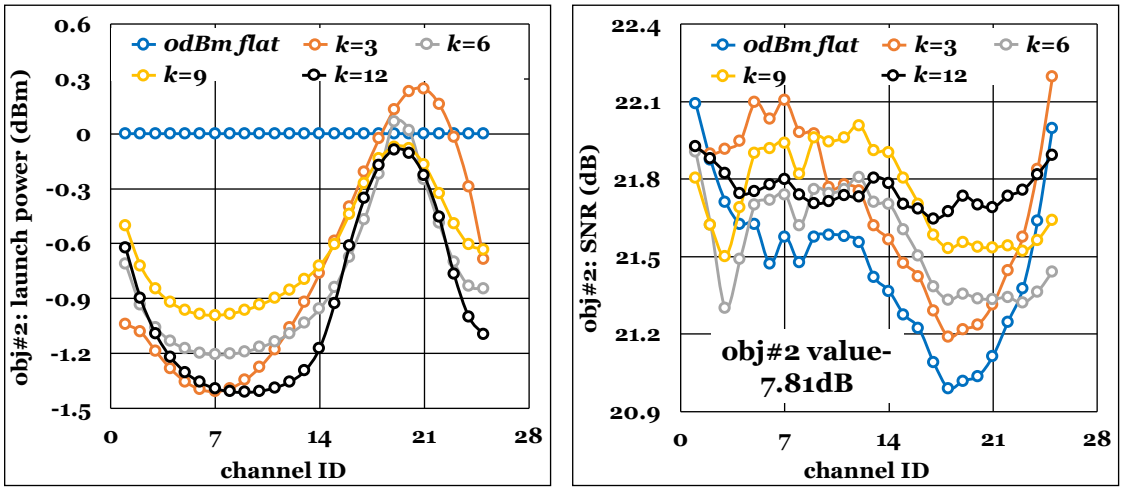


Figure. 6.14. (a) Launch power (dBm), and (b) SNR (dB) per channel as the number k of retraining cycles increase, for the proposed DT approach, $L_{iter} = 5$, obj#2 and EDFAs having peak-to-peak gain ripple of 0.4 dB.

show the evolution of obj#1 and obj#2 with the proposed approach for different retraining periods of $L_{iter} = 5, 10, 50$. Note that we have comparatively higher savings (~2.8dB) with respect to the one-time trained Qtool compared to flat EDFAs (**Figure. 6.11**), because the physical layer is more dynamic with the rippled EDFAs. The small drop in the second retraining cycle of **Figure. 6.13(a)** can be explained by the Qtool training process; in that cycle the Qtool did not match very well the real network and misled the optimization algorithm. This was then improved at the next retraining which involved more monitoring information from the new/projected network state.

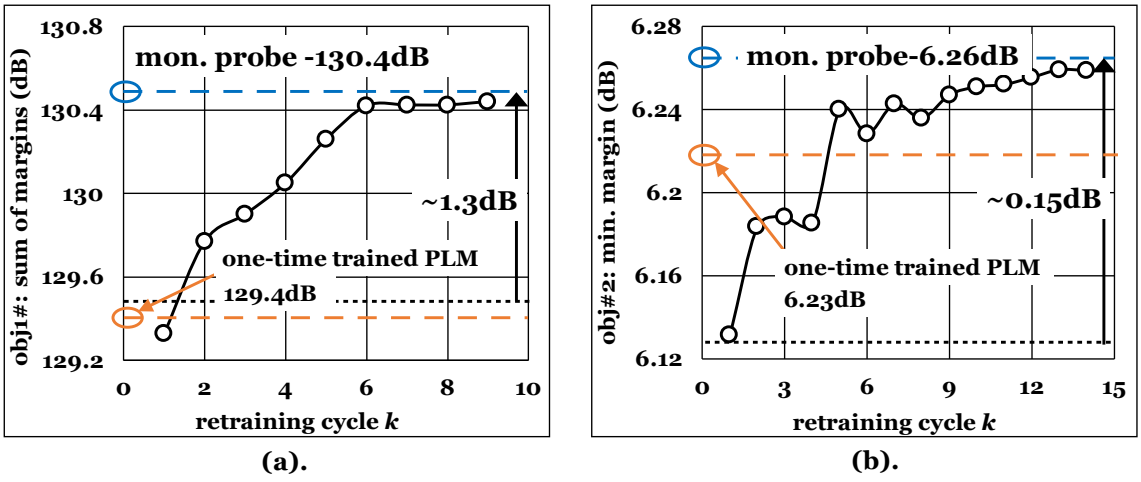


Figure. 6.15. (a) Obj#1, and (b) Obj#2 values as a function of the proposed DT approach retraining cycles (k), for flat EDFAs in the 2-links setup.

Finally, to study a scenario approaching a network, we extend the single link setup to a two-links setup as shown in **Figure. 6.10(b)**. We established 15 connections with different add/ drop locations and reused wavelengths at the intermediate node. We first focus on

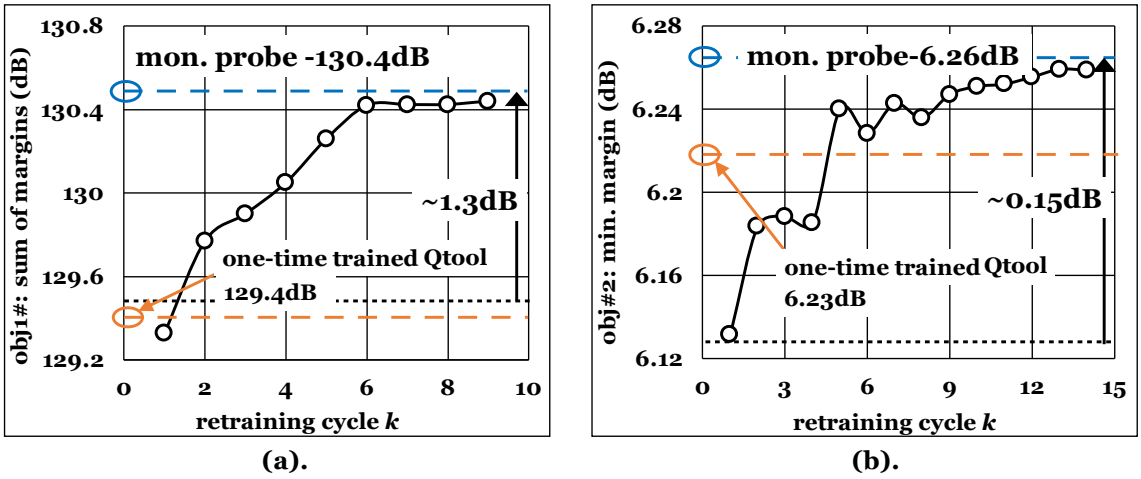


Figure. 6.16. (a) Obj#1, and (b) Obj#2 values as a function of the DT approach retraining cycles (*k*), for EDFAs with ripples in the 2-links setup.

Case 1, the span EDFA with flat gain profiles. **Figure. 6.15(a)** and **Figure. 6.15(b)** show the evolution of obj#1 and obj#2, respectively, as a function of the retraining cycles k for the DT scheme with $L_{iter} = 5$. The proposed DT approach improved by ~1.3 dB the obj#1 and by ~0.15dB the obj#2, with respect to the one-time trained Qtool based scheme.

For Case 2, we assigned a gain ripple profile of ±0.2dB and ±0.1dB to link 1-2 and link 2- 3, respectively. **Figure. 6.16(a)** and **Figure. 6.16(b)** shows the evolution of obj#1 and obj#2, respectively, as a function of the retraining cycles k for DT with $L_{iter} = 5$ scheme. We obtained a ~1.7dBs and ~0.6dBs of improvement for obj#1 and obj#2, respectively, with respect to training with a one-time trained Qtool based approach.

Compared to the single link setup, the improvements obtained for the two links setup were lower. The low channel load and the relatively wider channel separation due to add/drop is one of the main reasons for the low value of improvements. In the last subsection, we will show a comparatively bigger network topology with 3 links (4 nodes) with the adaptive number of L_{iter} as proposed in Eq. (5.11). The benefits of our proposed solution are higher for this network topology and on top of that we also noticed further reduction in monitoring calls compare to fixed L_{iter} scheme.

6.4.2 Effect of Monitoring Noise with Fixed L_{iter}

Monitors are not perfect and yield measurements that include errors, as discussed in Section 6.3. In our optimization we assumed the use of SNR values measured from the coherent receivers, which are typically assumed to have good accuracy. However, there are some fast-varying impairments that result in SNR fluctuations and contribute to monitoring errors. A typical method to suppress those is to average the SNR measurements over a period longer enough than the frequency of such effects. Hence, we modelled these monitoring errors by adding a random Gaussian noise v with mean = 0 and standard deviation (std) = 0.1, 0.2 and 0.4 dB, to the SNR values (provided by VPI). These noisy monitored SNR values were then fed to the interior point optimization algorithm, and were reflected in the algorithm's derivative calculations. The monitoring error results in a degraded optimization operation. In theory, the stochastic subgradient

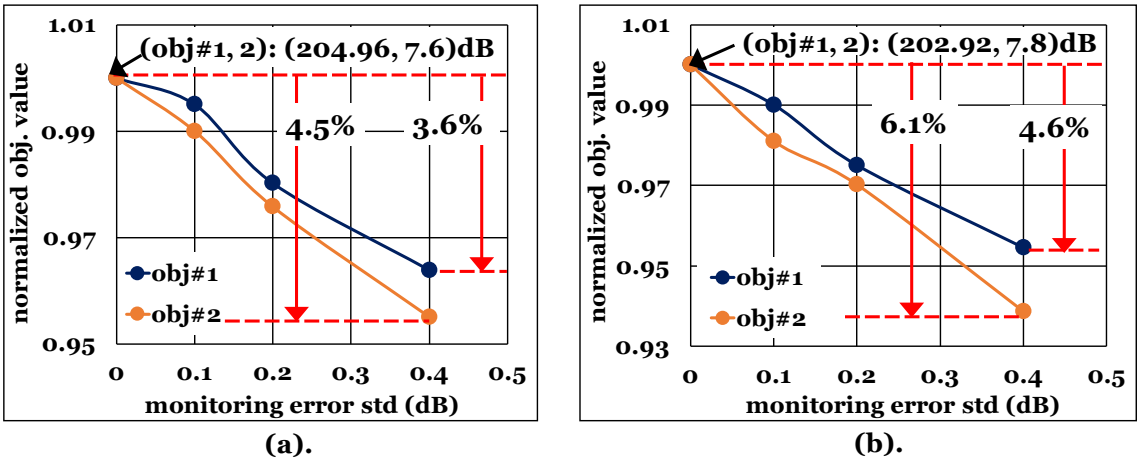


Figure. 6.17. Objective function variation as a function of the monitoring error std for (a) Flat EDFA gain, and (b) EDFA with gain ripple.

method [139] (which is a first order method) with specific small steps can find the optimum for the assumed zero mean errors after a high number of iterations. However, such method might not be applicable in real networks, where finite small steps are not feasible, and iterations must be constrained (to avoid effects of medium-term varying impairments). Also monitoring small SNR differences (due to the small steps) is rather hard (low slope of derivatives). So instead of seeking an ideal optimum we focused in a more realistic case and in the following results we used the interior point algorithm and a minimum power probe step $p_{step} = 0.1$ dBm. **Figure. 6.17(a)** shows the achieved objectives using the proposed DT approach with $L_{iter} = 5$ for varying monitoring error std (the mean was always equal to zero) for the flat EDFA case. Similar results were obtained for the monitoring probe optimization. A deterioration of the optimum with respect to ideal monitors (no noise, std=0) was observed. For a std = 0.4 dB with flat EDFAs, the objective decreased by 3.6% and 4.5% for obj#1 and obj#2, respectively. For EDFAs with gain ripples, the related decrease was even higher, ~5% and ~6% for obj#1 and obj#2, respectively, as shown in **Figure. 6.17(b)**.

6.4.3 Optimization Time

We now turn our attention on the optimization time. As discussed, the Qtool and the convex (interior-point) optimization algorithm were implemented in MATLAB. So, in MATLAB we measured the overall computation time, which included the time t_{calc} that the algorithm calculated the gradients/Hessian and also the next launch powers, the time t_{train} to train with ML the Qtool, and the time t_{Qtool} for the SNR calculations by the Qtool. Note that the Qtool-related times appear only in the schemes that use it (one-time trained Qtool, and proposed DT). Also, note that the computation time was measured until the algorithm obtained the optimum, so it included all retaining cycles, algorithm iterations and finite difference subroutine calls. Following the notation of Section 6.3, for the monitoring probes scheme we measured: $L_{mon\_prob} \cdot t_{calc}$, for the one-time trained PLM we measured: $t_{train} + L_{Qtool} \cdot (D \cdot t_{Qtool} + t_{calc})$, and for the DT we measured: $K_{retrain} \cdot (t_{train} + L_{iter} \cdot (D \cdot t_{Qtool} + t_{calc}))$. To obtain the DT results we used a retraining period $L_{iter} = 5$.

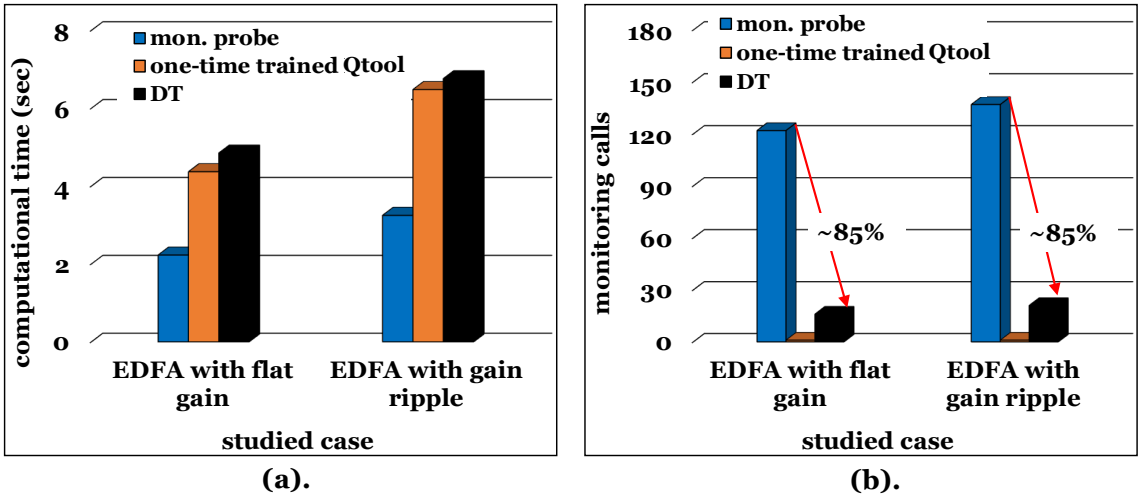


Figure. 6.18. (a) Computational time (sec), and (b) Number of monitoring calls required, for EDFA with flat and rippled gain cases respectively.

Figure. 6.18(a) shows the overall computational time for obj#1 for EDFA with flat and rippled gain (std=0). The computational time for the monitoring probe-based approach was the smallest around 2sec for flat gain EDFA and ~2.5 sec for EDFA with gain ripples, since it only includes t_{calc}. The one-time trained Qtool was slightly faster than the DT approach (~4 sec compared to ~4.5 sec for flat EDFA gain case and ~7 sec compared to ~7.5 sec for the case of EDFA with gain ripple), since the Qtool retraining time, which is their key difference, was quite fast. The computation time for the case of EDFAs with gain ripples (shown in **Figure. 6.18(a)**), was higher for all optimization schemes, since the algorithm took more iterations to reach the optimum. From the above measurements we excluded the monitoring time t_{mon}. It was excluded since we did not want to use some reference monitoring time. However, we expect it to be some orders of magnitude higher than the timescales reported in **Figure. 6.18(a)**.

Today, transponders report the SNR/BER every 15 minutes [140]. The reporting time can be substantially reduced down to sec with NETCONF/YANG monitoring [141] or telemetry-based protocols [140], [142]. However, such reporting periods target failure recovery use cases, which are substantially different from the power optimization use case studied in this paper. Moreover, we need to consider that in the monitoring probes scheme, the monitoring happens after the probing, that is, after changing the launch power of one or more connections. In such a case, we would need to wait for EDFA transient effects to settle. Also, for optimization we need to have a low monitoring error. So, we would need to time average to suppress the fast-varying impairment effects. Moreover, depending on the optimization method, it is required to monitor different times.

In the one-time trained Qtool based approach, we monitor only once, at the beginning of the optimization. In the DT approach we monitor $K_{retrain}$ times, once every Qtool retraining cycle. In the monitoring probes approach we train $L_{mon\_prob} \cdot D$. However, note that D depends on the optimization algorithm and the type of partial derivatives it calculates (first or second order). Thus, we might have different number of probes/monitoring per algorithm iteration. To avoid any confusion with timescales, we

show in **Figure. 6.18(b)** the number of monitoring calls, which in our simulations were measured as the number of times that we set new launch powers in VPI, executed the VPI transmission simulation, obtained the SNR values and forwarded them in MATLAB. We can clearly see the substantial higher number of monitoring calls performed by the monitoring probes approach, which would result in substantially higher overall optimization time (the addition of computation **Figure. 6.18(a)** and monitoring **Figure. 6.18(b)** times).

6.4.4 Additional Improvements with Adaptive L_{iter}

In this subsection, we present the results obtained with the adaptive version of the proposed scheme described by Eq. (6.11) and Eq. (6.12). We discuss the additional improvements in terms of further reduction in monitoring calls (or overall optimization time) compared to the previously presented fixed L_{iter} scheme. We simulated a 4 nodes topology (assumed to be a part of a bigger network) with 15 reused wavelengths and 21 connections in VPI as shown in **Figure. 6.19**. To make the simulations more realistic, we assumed different attenuation (0.19-0.21 dB/km), dispersion (16.5-17 ps/(nm.km)) and fiber non-linearity coefficients (1.3-1.38 W-1.km-1) for the three links, introducing uncertainties which need to be modeled in ML using per link parameters $\boldsymbol{r}$. We again used EDFAs with peak-to-peak gain ripple 0.4 dB, and 32 GBaud 16-QAM modulation format per connection.

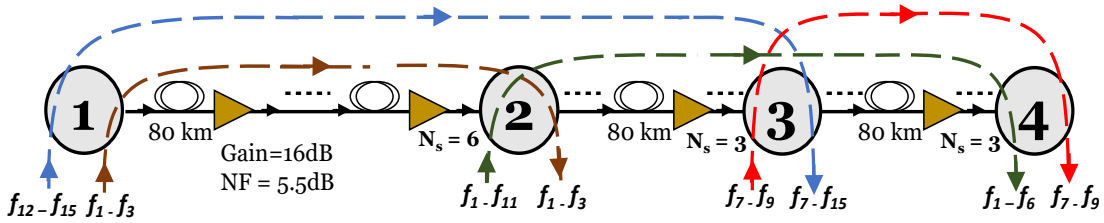


Figure. 6.19. VPI setup with 4 nodes and 21 connections with different paths/routes, added/dropped points to emulate a comparatively bigger network than Figure. 6.10.

When optimizing with the one-time trained Qtool approach, we monitor the initial SNR values in VPI, pass them to MATLAB, use them to train the PLM, which is then used for all algorithm intermediate iterations. The final powers are passed to VPI where we find the resulted real margin (obj#1 and obj#2). When optimizing with monitoring probes, the derivatives calculation subroutine probes, gives new launch powers, to VPI, which calculates the SNR values and passes them back. This process is performed several times until the algorithm calculates the partial derivatives and the next launch powers, and is repeated for each algorithm iteration.

In the proposed improved/extended approach, VPI takes the launch powers from the algorithm and calculates the SNR values. These are passed back to MATLAB, used to find the alignment error by Eq. 6.11 and the next period L_k by Eq. 6.12, and train the Qtool. This Qtool is then used for the next L_k algorithm intermediate iterations. Note that the L_k is not fixed in these simulations and depends upon the alignment error from the current and previous training cycle of the Qtool. Based on these alignment errors, the

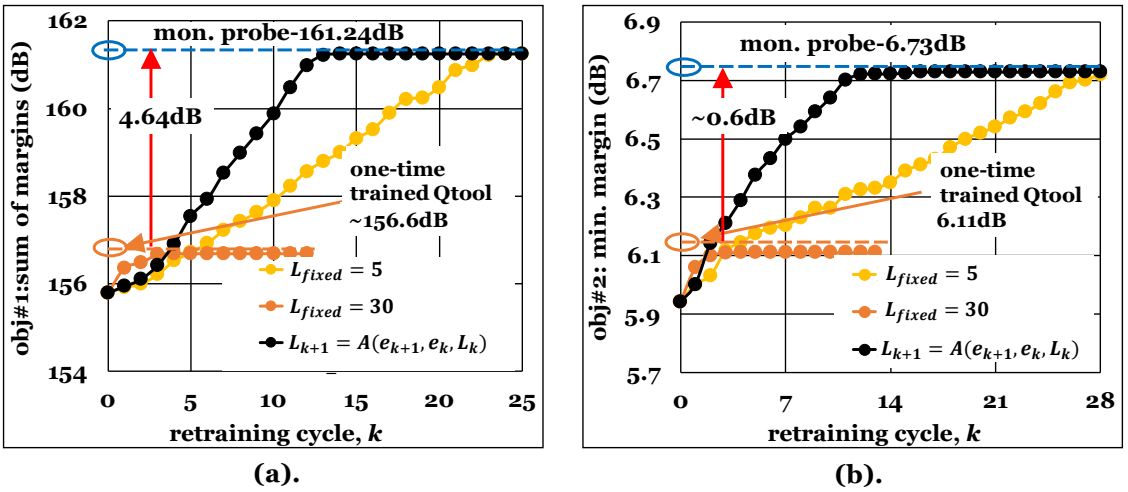


Figure. 6.20. (a) Obj#1, and (b) Obj#2 values as a function of proposed *adaptive*, *iterative* Qtool retraining based launch optimization approach for EDFAs with minimum-maximum peak-to- peak gain ripple values of ±0.1 to ±0.3 dB.

previous iterations are incremented/decremented by 5 using a function A as presented by Eq. 6.12.

Figure. 6.20 shows the optimization objective values (evaluated in VPI) as a function of the retraining cycle (k) of our proposed scheme. Compared to one-time trained Qtool approach, we noticed ~4.6 dB improvement in obj#1 (**Figure. 6.20(a)**) and ~0.6 dB in obj#2 (**Figure. 6.20(b)**). **Figure. 6.20** shows that with fixed retraining period $L_{iter} = L_{fixed} = 5$, the solution reaches the optimum, however it took more iterations. With the adaptive loop based scheme of Eq. 6.11 and Eq. 6.12 ($L_{iter} \neq L_{fixed} = A(e_{k+1}, e_k, L_k)$) we obtained ~54-60% reduction in retraining cycles without compromising the performance. In general, the monitoring probes approach achieves the optimum since it

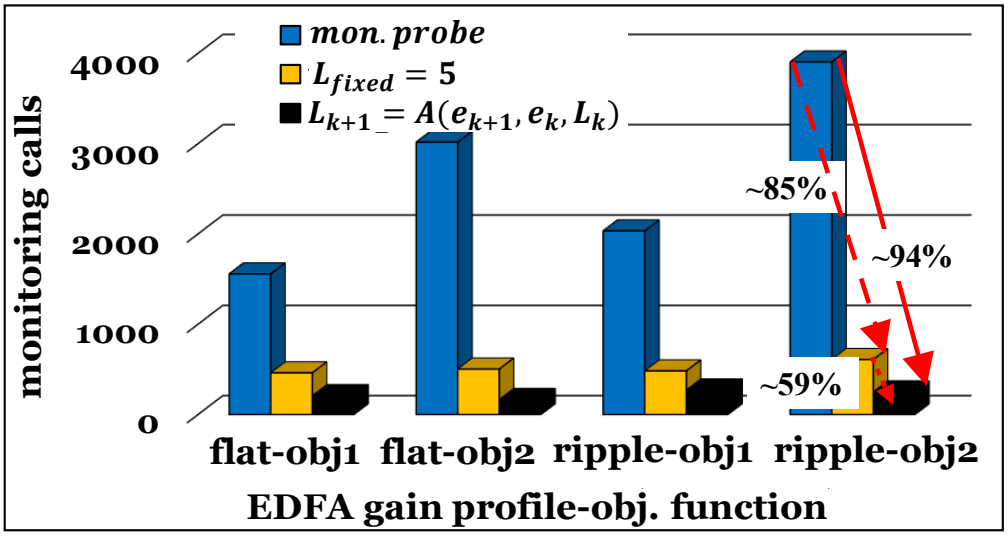


Figure. 6.21. Additional reduction (~59%) in the number of monitoring calls with *adaptive* iteration loop scheme for Obj#1 and Obj#2 with flat and rippled gain EDFA.

continually tracks the network state. However, the extensive interactions with the network are prohibitive.

In **Figure. 6.21**, we see that optimizing with monitoring probes required ~1500 to 3000 monitoring calls, for obj#1 and obj#2 with flat gain EDFAs. Also, ~1500 to ~4000 monitoring calls are required, for obj#1 and obj#2 with rippled EDFA gain profiles for the considered network topology (**Figure. 6.19**). This makes the overall approach very slow. Our approach substantially reduces the calls. With respect to the Section 6.4, where we used a fixed period (L_{fixed}) and links without uncertainties, the scheme already proved ~85% less monitoring calls requirement. However, with the devised adaptive and iterative loop scheme (Eq. (6.11) and Eq. (6.12)), we further reduced the calls by ~59%. Ultimately, we reduce by ~90 to 94% the monitoring calls with respect to optimizing with monitoring probes as shown in **Figure. 6.21**, which will reflect to a similar reduction in the total optimization time.

6.5 Conclusions

In control theory, closed control loops have been extensively studied as discussed in [128], [143]. However, control theory typically targets infinite time horizon problems, and considers fast loops with real-time feedback. Also, reinforcement learning has received attention on similar topics [144]. Reinforcement learning also targets infinite time horizon problems and a system described by a Markov decision process, which is different from the convex optimization problem that we have at hand. To the best of our knowledge, the identified issue of the lack of accuracy of the Qtool in dynamic optimization problems has not been studied in the past. Note that convex optimization algorithms and their interaction with a tool that represents reality (in optimization terms this is referred to as an “oracle”) have been studied [145], including exact and inexact oracles with varying inaccuracy models. The proposed solution presented in this Chapter shares certain ideas from this optimization field. We used convex algorithms to solve the launch power optimization problem, but we avoid heavy monitoring and apply the optimization to the network level. We also share ideas with [12], [70] on the use of monitoring and ML to train/align the Qtool with the physical layer conditions. However, we extend those and retrain/realign the Qtool in a closed control loop, targeting dynamic optimization problems.

Despite the extensive literature on ML-based and ML-improved Qtools, there has been limited discussion on the use of Qtools in dynamic optimization tasks. In this Chapter, we initially proposed to use an *iterative* closed control loop process to solve a complex/multivariable dynamic optimization problem. In particular, we propose to close the control loop after several algorithm intermediate calculations, apply the intermediate calculated changes and monitor the outcome so that we realign or retrain the Qtool with the real world/ optical network. The Qtool is used as a Digital Twin (DT), it is applied to a network with evolving conditions, it is parametric and is dynamically adjusted so that it replicates with enough accuracy the real-world for the optimization at hand. We applied our proposed method to solve the dynamic version of the launch power optimization problem. With the proposed iterative Qtool retraining/DT scheme, we showed an improvement of ~8.5dB and ~0.64dB in sum of margins and lowest margin, respectively on a single link with 25 WDM channels, over optimization with a one-time

trained Qtool scheme. Moreover, the proposed approach achieved near to optimum solutions as found by optimizing and continuously probing and monitoring the network but with ~85% fewer monitoring probe calls. This will indeed reflect to a similar reduction in the total optimization time.

We then extended the *iterative* closed control loop scheme with *adaptive* number of iterations after each retraining cycle based on the Qtool realignment error from the current and previous retraining cycle. In particular, we proposed to use an adaptive and iterative closed control loop process to solve the multivariable dynamic transponders launch power optimization problem. We showed an improvement of ~4.6dB and ~0.6dB in sum of margins and lowest margin, respectively, on a 4 node network with 21 established connections, over optimization with a one-time trained Qtool scheme. Compare to the initially proposed iterative closed control loop scheme, with this adaptive scheme, we showed additional improvements of ~8 to 9% resulting in overall ~94% fewer monitoring probe calls requirement, which will reflect to a similar reduction in the total optimization time.

Chapter 7

Conclusions and Closing Discussions

In this thesis we presented ML assisted QoT estimation tool accuracy enhancement schemes to improve optical network planning, upgrade, and optimization operations. We also proposed novel aspects of Qtool accuracy improvement, as well as its integration with a dynamic multivariable optimization problem.

7.1 Summary

This thesis began with an overview of optical communications and networks, as well as a look at the current state of the field. We discussed how the Telecom industry's increased data traffic and revenue compression are putting a lot of pressure on the optical community to develop novel solutions to increase the transport capacity while remaining cost effective. This requirement is pushing operators towards multi-vendor network ecosystem. Every piece of equipment and transmission technique at the physical layer have an impact on the overall network behavior in such rapidly evolving optical networks. As a result, physical layer awareness in network design and operation is critical for fairly assessing the potentialities of different physical layer technologies and exploiting their capabilities in modern flexible, transparent, and reconfigurable optical networks. Monitoring provides information about the network's underlying physical layer condition, while ML can be used to better understand these conditions and optimize the network.

For reliable and efficient network planning, upgrade, operation and optimization, accurate estimation of QoT before establishing or reconfiguring the connection is necessary. In optical networks, a PLM is typically used as a QoT estimation tool (Qtool) which includes a design margin to account for modeling and parameter inaccuracies, ensuring the acceptable performance. Starting from these requirements to implement a Qtool, we provided a review of propagation impairments in transparent optical networks, mainly focusing on nonlinear modeling or more specifically the well-known GN model, which is then used as a PLM. We extended the GN model to capture the EDFA gain ripple penalties. We proposed to use available monitored information from established connections and appropriate supervised ML to model the EDFA gain ripple penalties. We then proposed ML model to estimate end-to-end penalty generated at ROADM nodes due to filter spectral uncertainties & their cascaded effects. We developed independent

models for the two effects and then a joint model. The ML model would be then used for estimating the penalties of new connection requests, improving the Qtool estimation accuracy and thus reducing the required design margin.

Later in Chapter 5, we proposed Qtool extensions to capture the performance variations of multi-vendor transponders (TPs) in partial disaggregated optical networks. We also devised a monitoring and ML based scheme to identify the performance factors of multi- vendor TPs for more accurate Qtool estimations in offline and online network planning scenarios. We trained standard Qtool (based on the GN model as PLM) on these performance factors via ML based nonlinear fitting technique. The trained Qtool would be then used for estimating the QoT of new connection requests, improving the Qtool estimation accuracy and thus reducing the required design margin. We also showed the benefit of the proposed multi-vendor Qtool for static optimization of transponders launch power in planning phase with potential SNR improvements.

Finally, in Chapter 6, we proposed to use an adaptive and iterative closed control loop process to solve a complex/multivariable dynamic optimization problem. In particular, we proposed to close the control loop after several algorithm intermediate calculations, applying the intermediate calculated changes, and monitoring the outcome so that we realign (retrain) the Qtool with the real-world/optical network. In a sense, the Qtool is used as a digital twin (DT); it is applied to a network with evolving conditions, it is parametric and is dynamically adjusted so that it replicates the real world for the optimization at hand with enough accuracy. We showed the benefits of the proposed scheme with dynamic launch power optimization problem as a use case, where we have a set of established connections, and we optimize their launch powers while the network operates.

Most of the work conducted in this thesis is based on simulations. By incorporating experimentally measured EDFA gain ripple profiles and simulating the real world with VPI Transmission Maker, we attempted to make the simulation environment as realistic as possible. However, we do believe that much more experimental work on the methods presented in this thesis is required to verify the full benefits of ML in the design and optimization of optical networks.

7.2 Topics for Further Research – Future Steps

There are several possible future evolutions for the topics tackled in this thesis.

- First and foremost is that most of the work on the Qtool accuracy improvement and dynamic optimization presented in this thesis is developed based on simulation measurements in VPI Transmission Maker due to the lack of real measurements. So, it will be very important to further extend the work considering experimental or real- field measurements.

- In Chapter 4, we mainly focused on improving the Qtool estimation accuracy and reducing the design margin for incremental planning scenarios. However, the network-wide savings in terms of used transponders, spectrums, throughput, and other similar factors may be an important research area that should be investigated further in the future.

- The thesis mostly focuses on the use of monitoring information to improve the Qtool estimation accuracy and to reduce the design margin. However, we relied on offline capturing/monitoring, pre-processing the data and training of the ML models. Another possible extension of the work is to create an online data monitoring and pre-processing platform and to derive various types of machine learning models (weights, biases etc.) that can capture real-time changes in networks and adapt models accordingly. It is also worth looking into ways/techniques to reduce the computational cost and processing time of complex ML models and platforms so that the Qtool learning process is quick and adaptable to more advanced networks like multivendor disaggregated networks, 5G backhaul applications, and so on.

- All the schemes presented in Chapter 4 to Chapter 6 are based on exploiting the monitoring information to train Qtool. In all these schemes, one of the biggest factor that needs to be consider is the trade-off between the accuracy of the monitored information over the monitoring time. Some fast varying effects such as polarization needs proper averaging by the monitors, to provide an accurate value of the monitored QoT parameter. So, one of the extension of the presented work is to use real (SNR) monitors, characterize them and obtain measurements with the real error or fluctuations. By doing so, it is possible to relate the real monitoring-time (with error) and convergence of the implemented algorithms in this Ph.D. thesis.

- In Chapter 6, the Qtool retraining work was focused on the interaction between the monitored data, Qtool and the real world (VPI Transmission Maker). However, in order to configure intermediate iterations to retrain the Qtool, we didn't explicitly focus on the control plane part, which is related to the interaction between the physical layer monitoring and control plane. It could be important to experimentally verify the proposed approach considering also the control plane functionalities and features into account that can affect the performance of the schemes/results presented in this thesis such as optimization time, deviation from optimization objectives etc. We could also verify the proposed Qtool retraining concept based on monitoring and ML predictions with variety of use cases and their corresponding appropriate optimization algorithms.

- Moreover, it could also be important to work on the extension of the proposed Qtool with more accurate modeling options such as stimulated Raman Scattering (SRS) effect, considering hybrid (EDFA + Raman) amplification etc., and the integration of it in network orchestrator. A real demonstration of such integration can actually show how a highly accurate Qtool can bring advantage in terms of network's flexibility, performance and automation.

Bibliography

[1] J. Hecht, "City of Light: The Story of Fiber Optics," Oxford University Press, 2004.

[2] S. Matsuoka, "Ultrahigh-speed ultrahigh-capacity transport network technology for cost-effective core and metro networks," NTT Technical Review, vol. 9, no. 8, 2011.

[3] V.Lopez and L.Velasco (Editors), "Elastic Optical Networks. Architectures, Technologies, and Control," Springer, 2016.

[4] P. Lu, L. Zhang, X. Liu, J.Yao, and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," in IEEE Network, vol. 29, no. 5, pp. 36-42, Sep. – Oct. 2015.

[5] O.Gerstel, M. Jinno, A. Lord, and S. J. B.Yoo, "Elastic optical networking: A new dawn for the optical layer?," in IEEE Communications Magazine, vol. 50, no. 2, pp. s12-s20, Feb. 2012.

[6] K. Roberts, Q. Zhuge, I. Monga, S. Gareau, and C. Laperle, "Beyond 100 Gb/s: Capacity, flexibility, and network optimization [Invited]," in IEEE/OSA Journal of Optical Communications and Networking, vol. 9, no. 4, pp. C12-C23, Apr. 2017.

[7] A. Mitra, A. Lord, S. Kar, and P. Wright, "Effect of link margin and frequency granularity on the performance of a flexgrid optical network," in Optics Express, vol. 22, no. 1, pp. 41–46, Jan. 2014.

[8] Y. Pointurier, "Design of low-margin optical networks," in IEEE/OSA Journal of Optical Communications and Networking, vol. 9, no. 1, pp. A9-A17, Jan. 2017.

[9] J. L.Auge, "Can we use flexible transponders to reduce margins?," in proceedings of Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013, pp. 1-3.

[10] M. Filer, M. Cantono, A. Ferrari, G. Grammel, G. Galimberti, and V. Curri, "Multi-vendor Experimental Validation of an Open Source QoT Estimator for Optical Networks," in Journal of Lightwave Technology, vol. 36, no. 15, pp. 3073-3082, Aug. 2018.

[11] K. Christodoulopoulos, C. Delezoide, N. Sambo, A. Kretsis, I. Sartzetakis, A. Sgambelluri, N. Argyris, G. Kanakis, P. Giardina, G. Bernini, D. Roccato, A. Percelsi, R. Morro, H. Avramopoulos, P. Castoldi, P. Layec, and S. Bigo, "Toward efficient, reliable, and autonomous optical networks: the ORCHESTRA solution [Invited]," in IEEE/OSA Journal of Optical Communications and Networking, vol. 11, no. 9, pp. C10-C24, Sep. 2019.

[12] E. Seve, J. Pesic, C. Delezoide, S. Bigo, and Y. Pointurier, "Learning Process for Reducing Uncertainties on Network Parameters and Design Margins," in IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 2, pp. A298-A306, Feb. 2018.

[13] D. Rafique and L. Velasco, "Machine learning for network automation: overview, architecture, and applications [Invited Tutorial]," in IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 10, pp. D126-D143, Oct. 2018.

[14] [Online] https://www.vpiphotonics.com/index.php

[15] A. Mahajan, K. Christodoulopoulos, R. Martinez, S. Spadaro and R. Munoz, "Machine learning assisted EDFA gain ripple modelling for accurate QoT estimation," in proceedings of European Conference on Optical Communication (ECOC), 2019, pp. 1-4.

[16] A. Mahajan, K. Christodoulopoulos, R. Martínez, S. Spadaro, and R. Muñoz, "Modeling EDFA Gain Ripple and Filter Penalties with Machine Learning for Accurate QoT Estimation," in Journal of Lightwave Technology, vol. 38, no. 9, pp. 2616-2629, May 2020.

[17] A. Mahajan, K. Christodoulopoulos, R. Martinez, S. Spadaro, and R. Munoz, "Improving QoT Estimation Accuracy with DGE Monitoring using Machine Learning," in proceedings of International Conference on Optical Network Design and Modeling (ONDM), 2020, pp. 1-6.

[18] A. Mahajan, K. Christodoulopoulos, R. Martinez, S. Spadaro and R. Munoz, "Modeling Filtering Penalties in ROADM-Based Networks with Machine Learning for QoT Estimation," in proceedings of Optical Fiber Communications Conference and Exhibition (OFC), 2020, pp. 1-3.

[19] A. Mahajan, K. (Kostas) Christodoulopoulos, R. Martínez, S. Spadaro, and R. Muñoz, "Modelling Multi-Vendor Transponders Performance and Optimizing Launch Power", in proceedings of European Conference on Optical Communication (ECOC), Dec. 2020, pp. 1-4.
[20] A. Mahajan, K. Christodoulopoulos, R. Martínez, R. Muñoz, S. Spadaro, "Quality of Transmission Estimator Retraining for Dynamic Optimization in Optical Networks", in IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 13, no. 4, pp. B45-B59, Apr. 2021.
[21]A. Mahajan, K. Christodoulopoulos, R. Martínez, R. Muñoz, and S. Spadaro, "Adaptive and Iterative QoT Estimator Retraining for Launch Power Optimization", in proceedings of Optical Fiber Communication Conference (OFC), 2021, pp. 1-3.
[22] https://h2020-onfire.eu/
[23] J. M. Simmons, "Optical Network Design and Planning," Springer, 2014.
[24] I. Tomkos, "The Evolution of Optical Networking," in Proceedings of the IEEE, vol. 100, no. 5, pp. 1017-1022, May 2012.
[25] X, Liu, "Evolution of Fiber-Optic Transmission and Networking toward the 5G Era," in Elsevier iScience, vol. 22, pp. 489-506, Dec. 2019.
[26] A. A. M. Saleh and J. M. Simmons, "All-Optical Networking—Evolution, Benefits, Challenges, and Future Vision," in Proceedings of the IEEE, vol. 100, no. 5, pp. 1105–1117, May 2012.
[27] T. Economist, "The great telecoms crash," The Economist, Jul. 2002.
[28] R. Mears, L. Reekie, I. Jauncey, and D. Payne, "Low-noise erbium doped fibre amplifier operating at 1.54μm," in Electronics Letters, vol. 23, no. 19, pp. 1026, Sep. 1987.
[29] R. W. Tkach, "Scaling optical communications for the next decade and beyond," in Bell Labs Technical Journal, vol. 14, no. 4, pp. 3-9, Dec. 2010.
[30] B. Collings, "New devices enabling software-defined optical networks," in IEEE Communications Magazine, vol. 51, no. 3, pp. 66-71, Mar. 2013.
[31]J. Theodoras, "A Primer on ROADM Architectures," Whitepaper, ADVA Optical Networking, Dec. 2012.
[32] F. Cugini, "Designing Disaggregated Optical Networks," in proceedings of International Conference on Optical Network Design and Modeling (ONDM), 2020, pp. 1-3.
[33] E. Riccardi, P. Gunning, Ó. G. de Dios, M. Quagliotti, V. López, and A. Lord, "An Operator view on the Introduction of White Boxes into Optical Networks," in Journal of Lightwave Technology, vol. 36, no. 15, pp. 3062-3072, Aug. 2018.
[34] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2016–2021," [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf
[35] S. Hardy, "Open optical line system pros and cons at the Open Optical Conference," in Lightwave Online, Nov. 2017. Available: http://www.lightwaveonline.com/articles/2017/11/open-optical-linesystem-pros-and-cons-at-the-open-optical-conference.html
[36] OpenROADM web site: http://www.openroadm.org/home.html
[37] OpenConfig web site: http://www.openconfig.net/
[38] G. P. Agrawal, "Nonlinear Fiber Optics," Amsterdam: Academic Press, 2013, OCLC: 820850077.
[39] C. R. Menyuk and A. Galtaross, "Polarization Mode Dispersion," New York, N.Y.: Springer, 2005, OCLC: 64583024.
[40] T. Rahman, A. Napoli, D. Rafique, B. Spinnler, M. Kuschnerov, I. Lobato, B. Clouet, M. Bohn, C. Okonkwo, and H. de Waardt, "On the Mitigation of Optical Filtering Penalties Originating from ROADM Cascade," in IEEE Photonics Technology Letters, vol. 26, no. 2, pp. 154-157, Jan. 2014.

[41] J. M. Fabrega, M. Svaluto Moreolo, L. Martin, A. C. Piat, E. Riccardi, D. Roccato, N. Sambo, F. Cugini, L. Potì, S. Yan, E. Hugues-Salas, D. Simeonidou, M. Gunkel, R. Palmer, S. Fedderwitz, D. Rafique, T. Rahman, H.D. Waardt, and A. Napoli, "On the filter narrowing issues in elastic optical networks," in IEEE/OSA Journal of Optical Communications and Networking, vol. 8, no. 7, pp. A23-A33, Jul. 2016.

[42] M.G. Taylor, "Coherent Detection Method Using DSP for Demodulation of Signal and Subsequent Equalization of Propagation Impairments," in IEEE Photonics Technology Letters, vol. 16, no. 2, pp. 674-676, Feb. 2004.

[43] B. Pedersen, A. Bjarklev, J. H. Povlsen, K. Dybdal, and C. C. Larsen, "The design of erbium-doped fiber amplifiers," in Journal of Lightwave Technology, vol. 9, no. 9, pp. 1105-1112, Sep. 1991.

[44] E. Desurvire, "Analysis of noise figure spectral distribution in erbium doped fiber amplifiers pumped near 980 and 1480 nm," in Applied Optics, vol. 29, no. 21, pp. 3118-3125, Jul. 1990.

[45] J. Zyskind, and A. Srivastava, "Optically Amplified WDM Networks: Principles and Practices. Amsterdam, The Netherlands: Elsevier, 2011.

[46] A. Carena, V. Curri, G. Bosco, P. Poggiolini, and F. Forghieri, "Modeling of the Impact of Nonlinear Propagation Effects in Uncompensated Optical Coherent Transmission Links," in Journal of Lightwave Technology, vol. 30, no. 10, pp. 1524–1539, May 2012.

[47] P. Johannisson and M. Karlsson, "Perturbation Analysis of Nonlinear Propagation in a Strongly Dispersive Optical Communication System," in Journal of Lightwave Technology, vol. 31, no. 8, pp. 1273–1282, Apr. 2013.

[48] R. Dar, M. Feder, A. Mecozzi, and M. Shtaif, "Accumulation of nonlinear interference noise in fiber-optic systems," in Optics Express, vol. 22, no. 12, p. 14 199, Jun. 16, 2014.

[49] A. Bononi, P. Serena, N. Rossi, E. Grellier, and F. Vacondio, "Modeling nonlinearity in coherent transmissions with dominant intrachannel fourwave- mixing," in Optics Express, vol. 20, no. 7, pp. 7777–7791, Mar. 2012.

[50] P. Poggiolini, G. Bosco, A. Carena, V. Curri, Y. Jiang, and F. Forghieri, "The GN-model of fiber non-linear propagation and its applications," in Journal of Lightwave Technology, vol. 32, no. 4, pp. 694–721, Feb. 2014.

[51] A. Splett, C. Kurtzke, and K. Petermann, "Ultimate Transmission Capacity of Amplified Optical Fiber Communication Systems taking into Account Fiber Nonlinearities," in proceedings of European Conference and Exhibition on Optical Communications (ECOC), 1993, pp. 41–44.

[52] H. Louchet, A. Hodzic, and K. Petermann, "Analytical model for the performance evaluation of DWDM transmission systems," in IEEE Photonics Technology Letters, vol. 15, no. 9, pp. 1219–1221, Sep. 2003.

[53] J. Pan, C. Pulikkaseril, L. Stewart, and S. Tibuleac, "Comparison of ROADM filter shape models for accurate transmission penalty assessment," in proceedings of IEEE Photonics Conference (IPC), 2016, pp. 550–551.

[54] B. Ramamurthy, D. Datta, H. Feng, J. Heritage, and B. Mukherjee, "Impact of transmission impairments on the teletraffic performance of wavelength-routed optical networks," in Journal of Lightwave Technology, vol. 17, no. 10, pp. 1713–1723, Oct. 1999.

[55] P. Poggiolini, G. Bosco, A. Carena, R. Cigliutti, V. Curri, F. Forghieri, R. Pastorelli, and S. Piciaccia, "The LOGON Strategy for Low-Complexity Control Plane Implementation in New-Generation Flexible Networks," in proceedings of Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013, pp. 1-3.

[56] I. Sartzetakis, K. Christodoulopoulos, C. P. Tsekrekos, D. Syvridis, and E. Varvarigos, "Quality of transmission estimation in WDM and elastic optical networks accounting for space–spectrum dependencies," in IEEE/OSA Journal of Optical Communications and Networking, vol. 8, no. 9, pp. 676–688, Sep. 2016.

[57] Y. Pointurier, “Machine learning techniques for quality of transmission estimation in optical networks” in IEEE/OSA Journal of Optical Communications and Networking, vol. 13, no. 4, pp. B60-B71, Apr. 2021.
[58] Introductory White Paper, “Software-Defined Networking: The New Norm for Networks (White Paper),” 2014. https://www.opennetworking.org/sdn-resources/sdn-library/whitepapers
[59] J. Mata, I.D. Miguel, R.J. Durán, N. Merayo, S. K. Singh, A. Jukan, and M. Chamania, “Artificial Intelligence (AI) Methods in Optical Networks: A Comprehensive Survey', in Elsevier, Optical Switching and Networking, vol. 28, pp. 43-57, Apr. 2018.
[60] [Online] - http://www.orchestraproject.eu/
[61] “Draft revised G.694.1 version 1.3,” Unpublished ITU-T Study Group 15, Question 6.
[62] J. Yu and N. Chi, “Digital Signal Processing in High-Speed Optical Fiber Communication Principle and Application,” Springer, 2020.
[63] [MIT Lecture] - https://www.mit.edu/~9.520/spring09/Classes/multiclass.pdf
[64] H. P. Gavin, “The Levenberg-Marquardt Algorithm for Non-Linear Least Squares Curve-Fitting Problems,” Notes CE281, Duke university, North Carolina, 2020 (http://people.duke.edu/~hpgavin/ce281/lm.pdf)
[65] A. Björck, “Numerical methods for least squares problems, Society for industrial and Applied Mathematics (SIAM), 1996.
[66] M.I.A. Lourakis, “A brief description of the Levenberg-Marquardt algorithm implemented by levmar,” Technical Report, Institute of Computer Science, Foundation for Research and Technology - Hellas, 2005.
[67] K. Madsen, N.B. Nielsen, and O. Tingleff, “Methods for nonlinear least squares problems. Technical Report,” Informatics and Mathematical Modeling, Technical University of Denmark, 2004.
[68] D.W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters,” in Journal of the Society for Industrial and Applied Mathematics, vol. 11, no. 2, pp. 431-441, Jun. 1963.
[69] L. Barletta, A. Giusti, C. Rottondi and M. Tornatore, “QoT estimation for unestablished lighpaths using machine learning,” in proceedings of Optical Fiber Communications Conference and Exhibition (OFC), 2017, pp. 1-3.
[70] I. Sartzetakis, K.Christodoulopoulos, and E.Varvarigos, “Accurate quality of transmission estimation with machine learning,” in IEEE/OSA Journal of Optical Communications and Networking, vol. 11, no. 3, pp. 140-150, Mar. 2019.
[71] K. Ishii, J. Kurumida, and S. Namiki, “Wavelength assignment dependency of AGC EDFA gain offset under dynamic optical circuit switching learning,” in proceedings of Optical Fiber Communications Conference and Exhibition (OFC), 2017.
[72] [Online] Available - https://www.cisco.com/c/en/us/td/docs/optical/15000r7_0/dwdm/planning/guide/70epg/d7ina.html
[73] R. Peretti, B. Jacquier, D. Boivin, E. Burov, and A. M. Jurdyc, “Inhomogeneous gain saturation in EDF: Experiment and modeling,” in Journal of Lightwave Technology, vol. 29, no. 10, pp. 1445–1452, May 2011.
[74] L. Qiao, A. Solheim, Q. Bu, Y. Luo, C. Fu, W. Zhang, and M. Le, “Erbium doped fiber amplifier with passive temperature compensation,” in proceedings of Optical Fiber Communications Conference and Exhibition (OFC), 2017, pp. 1-3.
[75] W. J. Miniscalco and R. S. Quimby, “General procedure for the analysis of Er3 +cross sections,” in Optics Letters, vol. 16, no. 4, pp. 258–260, Feb. 1991.
[76] S. J. Savory, R. J. Vincent, and D. J. Ives, “Design considerations for low-margin elastic optical networks in the nonlinear regime [Invited],” in IEEE/OSA Journal of Optical Communications and Networking, vol. 11, no. 10, pp. C76-C85, Oct. 2019.

[77] Y. Huang, C. L. Gutterman, P. Samadi, P. B. Cho, W. Samoud, C. Ware, M. Lourdiane, G. Zussman, and K. Bergman, "Dynamic mitigation of EDFA power excursions with machine learning," in Optics Express, vol. 25, no. 3, pp. 2245–2258, Feb. 2017.

[78] S. Zhu, C. L. Gutterman, W. Mo, Y. Li, G. Zussman, and D. C. Kilper, "Machine learning based prediction of erbium-doped fiber WDM line amplifier gain spectra," in proceedings of European Conference on Optical Communication (ECOC), 2018, pp. 1-3.

[79] S. Zhu, C. L. Gutterman, A. D. Montiel, J. Yu, M. Ruffini, G. Zussman, and D. C. Kilper, "Hybrid machine learning EDFA model," in proceedings of Optical Fiber Communications Conference and Exhibition (OFC), 2020, pp. 1-3.

[80] Y. You, Z. Jiang and C. Janz, "Machine Learning-Based EDFA Gain Model," in proceedings of European Conference on Optical Communication (ECOC), 2018, pp. 1-3.

[81] W. Mo, C. L. Gutterman, Y. Li, S. Zhu, G. Zussman, and D. C. Kilper, "Deep-neural-network-based wavelength selection and switching in ROADM systems," in IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 10, pp. D1-D11, Oct. 2018.

[82] C. L. Gutterman, W. Mo, S. Zhu, Y. Li, D. C. Kilper, and G. Zussman, "Neural network based wavelength assignment in optical switching," in proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, 2017, pp. 37-42.

[83] M. Ionescu, "Machine learning for ultrawide bandwidth amplifier configuration," in proceedings of International Conference on Transparent Optical Networks (ICTON), 2019, pp. 1-4.

[84] D. Zibar, A. M. R. Brusin, U. C. de Moura, F. Da Ros, V. Curri, and A. Carena, "Inverse System Design Using Machine Learning: The Raman Amplifier Case," in Journal of Lightwave Technology, vol. 38, no. 4, pp. 736-753, Feb. 2020.

[85] A. M. R. Brusin, U. C. de Moura, V. Curri, D. Zibar, and A. Carena, "Introducing load aware neural networks for accurate predictions of Raman amplifiers," in Journal of Lightwave Technology, vol. 38, no. 23, pp. 6481-6491, Dec. 2020.

[86] J. Wass, J. Thrane, M. Piels, R. Jones, and D. Zibar, "Gaussian process regression for WDM system performance prediction," in proceedings of Optical Fiber Communications Conference and Exhibition (OFC), 2017, pp. 1-3.

[87] J. Cho, S. Chandrasekhar, E. Sula, S. Olsson, E. Burrows, G. Raybon, R. Ryf, N. Fontaine, J.-C. Antona, S. Grubb, P. Winzer, and A. Chraplyvy, "Supply-power-constrained cable capacity maximization using multi-layer neural networks," in Journal of Lightwave Technology, vol. 38, no. 14, pp. 3652-3662, Jul. 2020.

[88] A. D'Amico, S. Straullu, A. Nespola, I. Khan, E. London, E. Virgillito, S. Piciaccia, A. Tanzi, G. Galimberti, and V. Curri, "Using machine learning in an open optical line system controller," in IEEE/OSA Journal of Optical Communications and Networking, vol. 12, no. 6, pp. C1-C11, Jun. 2020.

[89] L. Xia, J. Zhang, S. Hu, M. Zhu, Y. Song, and K. Qiu, "Transfer learning assisted deep neural network for OSNR estimation," in Optics Express, vol. 27, no. 14, pp. 19398–19406, Jul. 2019.

[90] J. Yu, W. Mo, Y.-K. Huang, E. Ip, and D. C. Kilper, "Model transfer of QoT prediction in optical networks based on artificial neural networks," in IEEE/OSA Journal of Optical Communications and Networking, vol. 11, no. 10, pp. C48-C57, Oct. 2019.

[91] M. Bouda, S. Oda, O. Vassilieva, M. Miyabe, S. Yoshida, T. Katagiri, Y. Aoki, T. Hoshida, and T. Ikeuchi, "Accurate prediction of quality of transmission based on a dynamically configurable optical impairment model," in IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 1, pp. A102-A109, Jan. 2018.

[92] E. Seve, J. Pesic, and Y. Pointurier, "Associating machine-learning and analytical models for quality of transmission estimation: combining the best of both worlds," in IEEE/OSA Journal of Optical Communications and Networking, vol. 13, no. 6, pp. C21-C30, Jun. 2021.

[93] I. Khan, M. Bilal, M. Siddiqui, M. Khan, A. Ahmad, M. Shahzad, and V. Curri, "QoT estimation for light-path provisioning in un-seen optical networks using machine learning,"

in proceedings of International Conference on Transparent Optical Networks (ICTON), 2020, pp. 1-4.

[94] T. A. Strasser, and J. L. Wagener, "Wavelength-selective switches for ROADM applications," in IEEE Journal of Selected Topics in Quantum Electronics, vol. 16, no. 5, pp. 1150-1157, Sept.-Oct. 2010.

[95] F. Paolucci, F. Cugini, F. Fresi, G. Meloni, A. Giorgetti, N. Sambo, L. Potí, A. Castro, L. Velasco, P. Castoldi, "Superfilter technique in SDN-controlled elastic optical networks [Invited]," in IEEE/OSA Journal of Optical Communications and Networking, vol. 7, no. 2, pp. A285-A292, Feb. 2015.

[96] C. Delezoide, P. Layec, and S. Bigo, "Automated alignment between channel and filter cascade," in proceedings of Optical Fiber Communications Conference and Exhibition (OFC), 2019, pp. 1-3.

[97] C. Delezoide, P. Ramantanis, and P. Layec, "Weighted filter penalty prediction for QoT estimation," in proceedings of Optical Fiber Communications Conference and Exposition (OFC), 2018, pp. 1-3.

[98] J. Pan and S. Tibuleac, "Real-Time ROADM Filtering Penalty Characterization and Generalized Precompensation for Flexible Grid Networks," in IEEE Photonics Journal, vol. 9, no. 3, pp. 1-10, Jun. 2017.

[99] M. P. Belanger, M. O'Sullivan and P. Littlewood, "Margin requirement of disaggregating the DWDM transport system and its consequence on application economics," in proceedings of Optical Fiber Communications Conference and Exposition (OFC), 2018, pp. 1-3.

[100] M. Filer, H. Chaouch, and X. Wu, "Toward transport ecosystem interoperability enabled by vendor-diverse coherent optical sources over an open line system," in IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 2, pp. A216-A224, Feb. 2018.

[101] J. Santos, N. Costa and J. Pedro, "On the Impact of Deploying Optical Transport Networks Using Disaggregated Line Systems," in IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 1, pp. A60-A68, Jan. 2018.

[102] "Evolving the Awareness of Optical Networks," whitepaper, Infinera.

[103] Y. R. Zhou, K. Smith, J. Weatherhead, P. Weir, A. Lord, J. Chen, W. Pan, D. Tanasoiu, and S. Wu, "Demonstration of a Novel Technique for Non-Intrusive In-Band OSNR Derivation Using Flexible Rate Optical Transponders Over a Live 727 km Flexible Grid Optical Link," in IEEE/OSA Journal of Lightwave Technology, vol. 35, no. 20, pp. 4399-4405, Oct. 2017.

[104] D. J. Ives and S. J. Savory, "Transmitter optimized optical networks," in proceedings of Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013, pp. 1-3.

[105] D. J. Ives, P. Bayvel and S. J. Savory, "Adapting Transmitter Power and Modulation Format to Improve Optical Network Performance Utilizing the Gaussian Noise Model of Nonlinear Impairments," in Journal of Lightwave Technology, vol. 32, no. 21, pp. 4087-4096, Nov. 2014.

[106] L. Yan, J. Zhao, E. Agrell and H. Wymeersch, "Power optimization in nonlinear flexible-grid optical networks," in proceedings of European Conference on Optical Communication (ECOC), 2015, pp. 1-3.

[107] P. Soumplis, K. Christodoulopoulos, M. Quagliotti, A. Pagano and E. Varvarigos, "Network Planning with Actual Margins," in IEEE/OSA Journal of Lightwave Technology, vol. 35, no. 23, pp. 5105-5120, Dec. 2017.

[108] H. Nakashima, Y. Akiyama, T. Hoshida, T. Oyama, T. Tanimura and J. C. Rasmussen, "Launch power optimization co-operated with digital nonlinear compensator for elastic optical network," in proceedings of Opto-Electronics and Communications Conference (OECC), 2015, pp. 1-3.

[109] I. Roberts, J. M. Kahn and D. Boertjes, "Convex Channel Power Optimization in Nonlinear WDM Systems Using Gaussian Noise Model," in IEEE/OSA Journal of Lightwave Technology, vol. 34, no. 13, pp. 3212-3222, July 2016.

[110] I. Roberts and J. M. Kahn, “Measurement-Based Optimization of Channel Powers with Non-Gaussian Nonlinear Interference Noise,” in IEEE/OSA Journal of Lightwave Technology, vol. 36, no. 13, pp. 2746-2756, Jul. 2018.

[111] I. Roberts, J. M. Kahn, J. Harley and D. W. Boertjes, “Channel Power Optimization of WDM Systems Following Gaussian Noise Nonlinearity Model in Presence of Stimulated Raman Scattering,” in IEEE/OSA Journal of Lightwave Technology, vol. 35, no. 23, pp. 5237-5249, Dec. 2017.

[112] L. Velasco, A. C. Piat, O. Gonzlez, A. Lord, A. Napoli, P. Layec, D. Rafique, A. D'Errico, D. King, M. Ruiz, F. Cugini, and R. Casellas, “Monitoring and Data Analytics for Optical Networking: Benefits, Architectures, and Use Cases,” in IEEE Network, vol. 33, no. 6, pp. 100-108, Nov.-Dec. 2019.

[113] L. Velasco, A. Sgambelluri, R. Casellas, L. Gifre, J. L. I. Zaragoza, F. Fresi, F. Paolucci, R. Martínez, E. Riccardi, “Building autonomic optical white box-based networks,” in IEEE/OSA Journal of Lightwave Technology, vol. 36, no. 15, pp. 3097-3104, Aug. 2018.

[114] K. D. R. Assis, S. Peng, R. C. Almeida, H. Waldman, A. Hammad, A. F. Santos, and D. Simeonidou, “Network virtualization over elastic optical networks with different protection schemes,” in IEEE/OSA Journal of Optical Communications and Networking, vol. 8, no. 4, pp. 272-281, Apr. 2016.

[115] J. Cho, S. Chandrasekhar, E. Sula, S. Olsson, E. Burrows, G. Raybon, R. Ryf, N. Fontaine, J. Antona, S. Grubb, P. Winzer, and A. Chraplyvy, “Maximizing fiber cable capacity under a supply power constraint using deep neural networks,” in proceedings of Optical Fiber Communications Conference and Exhibition (OFC), 2020, pp. 1-3.

[116] P. Soumplis, K. Christodoulopoulos and E. Varvarigos, “Dynamic connection establishment and network re-optimization in flexible optical networks,” in Springer, Photonic Network Communications, vol. 29, pp. 307-321, Apr. 2015.

[117] A. Castro, L. Velasco, M. Ruiz, M. Klinkowski, J. P. Fernández-Palacios, and D. Careglio, “routing and spectrum (re)allocation in future flexgrid optical networks,” in Elsevier, Computer Networks, vol. 56, no. 12, pp. 2869-2883, Aug. 2012.

[118] P. Papanikolaou, K. Christodoulopoulos and E. Varvarigos, “Incremental planning of multi-layer elastic optical networks,” in proceedings of International Conference on Optical Network Design and Modeling (ONDM), 2017, pp. 1-6.

[119] [Online]https://www.cisco.com/c/en/us/td/docs/optical/15000r7_0/dwdm/planning/guide/70epg/d7ina.html

[120] [Online]https://www.finisar.com/roadms-wavelength-management/foa-m7100da-evg2c-aa0xx

[121] [Online] https://www.nistica.com/fullfledge.html

[122] [Online] https://www.finisar.com/roadms-wavelength-management/focm01fxc1mn

[123] F. Vacondio, O. Rival, C. Simonneau, E. Grellier, A. Bononi, L. Lorcy, J.C. Antona, and S. Bigo, “On Nonlinear Distortions of Highly Dispersive Optical Coherent Systems”, in OSA Optics Express, vol. 20, no. 2, 1022–1032, Jan. 2012.

[124] S. P. Boyd and L. Vandenberghe, Convex Optimization. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[125] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, “A tutorial on geometric programming,” Optimization Eng., vol. 8, no. 1, pp. 67–127, 2007.

[126] A. Sadollah, H. Sayyaadi, and A. Yadav, “A dynamic metaheuristic optimization model inspired by biological nervous systems: Neural network algorithm,” in Elsevier Applied Soft Computing, vol. 71, pp. 747-782, Oct. 2018.

[127] S. Shahkarami, F. Musumeci, F. Cugini and M. Tornatore, “Machine-Learning-Based Soft-Failure Detection and Identification in Optical Networks,” in proceedings of Optical Fiber Communications Conference and Exposition (OFC), 2018, pp. 1-3.

[128] M. Hadi and E. Agrell, "Iterative Configuration in Elastic Optical Networks: (Invited Paper)," in proceedings of International Conference on Optical Network Design and Modeling (ONDM), 2020, pp. 1-3.
[129] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," in Synth. Lect. Commun. Netw. 3, pp. 1–211, 2010.
[130] M. Channegowda, R. Nejabati and D. Simeonidou, "Software-defined optical networks technology and infrastructure: Enabling software-defined optical network operations [invited]," in IEEE/OSA Journal of Optical Communications and Networking, vol. 5, no. 10, pp. A274-A282, Oct. 2013.
[131] S. Yan, F. N. Khan, A. Mavromatis, D. Gkounis, Q. Fan, F. Ntavou, K. Nikolovgenis, F. Meng, E. H. Salas, C. Guo, C. Lu, A. P. T. Lau, R. Nejabati, and D. Simeonidou, "Field trial of Machine-Learning-assisted and SDN-based Optical Network Planning with Network-Scale Monitoring Database," in proceedings of European Conference on Optical Communication (ECOC), 2017, pp. 1-3.
[132] N. Stojanovic, and D. Milenovic, "Data-driven Digital Twin approach for process optimization: an industry use case," in proceedings of IEEE International Conference on Big Data (Big Data), 2018, pp. 4202-4211.
[133] L. Wright, and S. Davidson, "How to tell the difference between a model and a digital twin," in Springer, Advanced Modeling and Simulation in Engineering Sciences, vol. 7, no. 13, Mar. 2020.
[134] E. Glaessgen, and D. Stargel, "The digital twin paradigm for future NASA and US air force vehicles," in proceedings of AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA, 2012.
[135] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," White paper, Mar. 2015.
[136] S. Boyd, and J. Park, "Subgradient Methods," Notes EE364b, Stanford, CA, USA: Stanford Univ., 2014 (https://stanford.edu/class/ee364b/lectures/subgrad_method_notes.pdf)
[137] M. S. Gockenbach, "Computing Derivatives by Finite Differences," Notes MA5630, MTU, Houghton, Michigan, 2013 (https://pages.mtu.edu/~msgocken/ma5630spring2003/lectures/diff/diff/diff.html)
[138] [Online] - https://metro-haul.eu/deliverables/
[139] S. Boyd, A. Mutapcic and J. Duchi, "Stochastic Subgradient Methods," Notes EE364b, Stanford, CA, USA: Stanford Univ., 2014 (https://web.stanford.edu/class/ee364b/lectures/stoch_subgrad_notes.pdf)
[140] R. Vilalta, N. Yoshikane, R. Casellas, R. Martínez, S. Beppu, D. Soma, S. Sumita, T. Tsuritani, I. Morita, and R. Muñoz, "GRPC-based SDN control and telemetry for soft-failure detection of spectral/spacial superchannels," in proceedings of European Conference on Optical Communication (ECOC), 2019, pp. 1-4.
[141] I. Sartzetakis, K. Christodoulopoulos, and E. Varvarigos, "Cross-layer adaptive elastic optical networks," in IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 2, pp. A154-A164, Feb. 2018.
[142] F. Paolucci, A. Sgambelluri, F. Cugini and P. Castoldi, "Network Telemetry Streaming Services in SDN-Based Disaggregated Optical Networks," in Journal of Lightwave Technology, vol. 36, no. 15, pp. 3142-3149, Aug. 2018.
[143] D. Bertsekas, Dynamic Programming and Stochastic Control, 1st ed. (Elsevier/Academic, 1976).
[144] R. S. Sutton and A. G. Barto, Introduction to Reinforcement Learning, 1st ed. (MIT Press, 1998).
[145] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods of smooth convex optimization with inexact oracle," Springer, Math. Program., vol. 146, pp. 37–75, Jun. 2013.