AIMS | African Institute for Mathematical Sciences CAMEROON

Reinforcement Learning for Routing, Modulation And Spectrum Assignment Problem in Elastic Optical Networks

Junior Momo Ziazet(junior.ziazet@aims-cameroon.org)
African Institute for Mathematical Sciences (AIMS)
Cameroon

Supervised by: Prof. Brigitte Jaumard
Concordia University, Canada

December 20, 2019

Submitted in Partial Fulfillment of a Cooperative Masters Degree in Industrial Mathematics at AIMS-Cameroon

Abstract

Global Internet traffic will continue to grow at high speed in the forthcoming years. To be able to meet this continuous and rapid growth, network operators are moving towards a new technology named Elastic Optical Network (EON). With EONs, unfortunately, the flexible resource allocation mechanisms makes the corresponding service provisioning designs more complicated in terms of network control and management, hardware development and spectrum management. To fully exploit the benefits of such flexibilities and realize cost-effective EON, researchers have intensively investigated the Routing, Modulation and Spectrum Allocation (RMSA) problem for EONs using algorithms based on either heuristic algorithm, Integer Linear Programming or genetic algorithm. However, those works only apply fixed RMSA policies or rely on simple empirical policies based on manually extracted features and therefore are unable to achieve real adaptive provisioning in EONs. Motivated by the success of several recent proposed Reinforcement Learning (RL) based approaches solving optimization combinatorial problems, we proposed a Reinforcement Learning based RMSA framework that can learn successful policies from static network operations. We implemented Deep Q-Network, REINFORCE and Advantage Actor-Critic (A2C) methods by parameterizing the policies with a convolutional neural networks (CNNs) that can sense complex EON states. The optimization target of each of the models at each step of servicing a request was to maximize a cumulative reward (total throughput in our case) within the rest of the episode. We test the models on 4 large-sized EONs topologies and the experimental results show that the proposed algorithms provide good solutions, with great generalization capability on different traffic matrices and types. REINFORCE provided the best results and achieved granted average throughput of 96% in the worst situation and 100% on some topologies for different traffic matrices and rate distribution.

Keywords: Elastic Optical Networks, Optical Routing, Modulation and Spectrum management, Reinforcement Learning, Convolutional Neural Networks.

Declaration

I, the undersigned, hereby declare that the work contained in this essay is my original work and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

Junior Momo Ziazet, 20 Dec 2019.

Contents

List of Figures

List of Tables

Acronyms

A2C Advantage Actor-Critic.

AFA-CA Adaptive Frequency Assignment with Collision Avoidance.

AI Artificial Intelligence.

AIMS African Institute for Mathematical Sciences.

BLSA Balanced Load Spectrum Allocation.

CAGR Compound Annual Growth Rate.

CNN Convolutional Neural Network.

CPU Central Processing Unit.

DNN Deep Neural Network.

DQN Deep Q-Network.

EB ExaBytes.

EON Elastic Optical Network.

GB Giga Bytes.

Gbps Giga bit per sencond.

GERAD Group for Research in Decision Analysis.

GHz Giga Hertz.

ILP Integer Linear Programming.

IP Internet Protocol.

ITU International Telecommunication Union.

IVADO Institute for Data Valorisation.

MC Monte Carlo.

MDP Markov Decision Process.

OSNR Optical Signal to Noise Ratio.

QAC Q Actor-Critic.

QAM Quadrature Amplitude Modulation.

QoT Quality of Transmission.

RAM	Random Access Memory.
RL	Reinforcement Learning.
RMSA	Routing Modulation and Spectrum Assignment.
RSA	Routing and Spectrum Assignment.
SARSA	State Action Reward State Action.
SDG	Stochastic Gradient Descent.
SPSR	Shortest Path with maximum Spectrum Reuse.
TB	TeraBytes.
TD	Temporal Difference.
WDM	Wavelength Division Multiplexing.
ZB	Zeta Bytes.

1. Introduction

From the youngest to the oldest, from small metropolitan areas to large ones, from private to public companies, from the informal to the formal sector, today everyone is moving towards the new Information and Communication Technologies (ICT). This is mostly due to the fact that, Nowadays, communication networks play an undeniably crucial role in our daily lives. From the perspective of companies aiming to gain competitive advantage up to the efficiency of administrations and public services such as health, education, security, the effect of ICT become more tangible as our society grows. In such a demanding context, high-speed and flawless service to users is of vital importance. Therefore, one of the main concerns of network operators is the efficient use of network resources to be able to efficiently meet the continuous and rapid growth of the user needs. Fortunately, Elastic Optical Network (EON) [12, 6] has proven to be a promising candidate for future high-speed optical communication because of its ability to efficiently allocate spectrum by allowing finer grid spacing, resulting in sub-streams called frequency slots.

With such a technique, unfortunately, the flexible resource allocation mechanisms in EON, makes the corresponding service provisioning designs more complicated in terms of network control and management, hardware development, and spectrum management. The purpose of this piece of work is an attempt to optimize the usage of resources (in terms of optical fiber and spectrum occupation) so that we fully exploit the benefits of such flexibilities and realize cost-effective EON. We achieve this by developing new Artificial Intelligence (AI) based algorithms and specifically reinforcement learning that are designed to achieve real adaptive provisioning in EON. In the following sections, we will provide the context and the description of the problem as well as where our contribution will sit, and this will be followed by the plan of the thesis.

1.1 Background

At the end of July 2019, the number of internet users is estimated at 4,536,248,808 which represents 58.8% of the global population, a 7.7% growth as compared to the last year [7]. Estimates suggest that annual global IP traffic will reach 4.8 Zetabytes (1ZB= 2^{30}TB) per year by 2022, or 396 exabytes (EB) per month. In 2017, the annual run rate for global IP traffic was 1.5 ZB per year or 122 EB per month. The Overall, IP traffic will grow at a Compound Annual Growth Rate (CAGR) of 26% from 2017 to 2022 and globally, busy hour (or the busiest 60 minute period in a day) Internet traffic will increase by a factor of 4.8 between 2017 and 2022, and average Internet traffic will increase by a factor of 3.7[5]. This continuous growth of traffic, nourished by the rise of new applications, including multimedia streaming services, cloud computing, inter data-center networking, can only be met with greater efficiency, flexibility, and scalability provided by flexible or elastic optical networks.

EONs are widely considered as the next-generation optical networks. This because historical data suggests that we are at the edge of surpassing the capacity of the traditional Wavelength Division Multiplexing (WDM) networks that have been used so far. Different from WDM that has a 100Gb/s fixed channel grid and wavelengths (channels) that has a 50 GHz width as recommended by the International Telecommunication Union (ITU) [11], EON provides flexible grid channel that supports variable bandwidth channels where bandwidth can be allocated in 12.5 GHz increments per wave and each wave's center frequency can be assigned as needed. This mechanism allows for serving high bit-rate requests as big as 400Gb/s and 1Tb/s (this was impossible with WDM). (See Section 2.3).

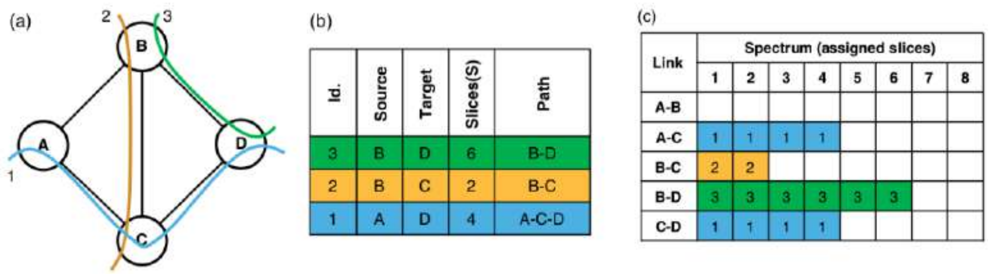
EONs further improve the spectrum utilization with consideration of adaptive modulation. Indeed, the modulation based EONs can reduce the allocated spectral bandwidth for shorter paths by increasing the number of modulated bits per symbol [40]. As a result, flexible network utilization efficiency is greatly improved compared to WDM based optical networks. More details on modulation are provided in Section 2.3.

With elastic optical networks, the challenge is on optimizing the spectrum usage through the so-called Routing Modulation and Spectrum Assignment (RMSA) problem.

1.2 Problem

In EONs, any demand or request is characterized by three factors namely the source, the destination, and the bit-rate. Given a network topology, The Routing and Spectrum Assignment (RSA) problem is defined as the problem of establishing connections for each request that has to be served (see Figure 1.1) by selecting:

- An appropriate routing path from the source node to the destination node,
- An available spectrum allocation on every link of the selected path.



Id.	Source	Target	Slices(S)	Path
3	B	D	6	B-D
2	B	C	2	B-C
1	A	D	4	A-C-D

Link	Spectrum (assigned slices)							
	1	2	3	4	5	6	7	8
A-B								
A-C	1	1	1	1				
B-C	2	2						
B-D	3	3	3	3	3	3		
C-D	1	1	1	1				

Figure 1.1: Example of RSA problem: (a) network topology and (b) the routed demands. (c) Spectrum allocation in each link. (extracted from [34]).

The most common objective are:

- Maximize the network throughput, which is defined as the summation of the rates of all granted requests,
- Minimize the total amount of spectrum usage.

The established connections have to satisfy the following constraints (see Figure 1.2):

- Non-overlapping: A frequency slot can be assigned to only one request on a given fiber link,
- Continuity: The same channels or frequency slots are assigned to all the path links of a request,
- Contiguity: The assigned slots have to be contiguous (adjacent to each other) in the frequency domain.

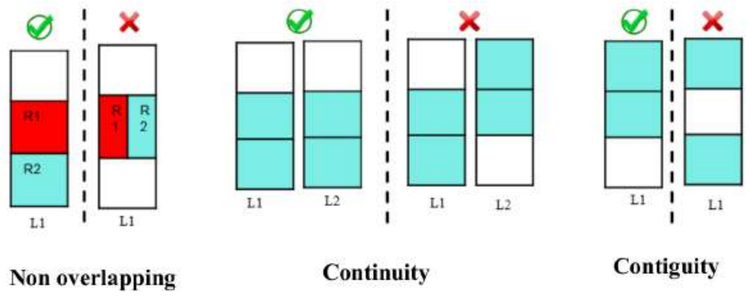


Figure 1.2: RSA Constraints.

The Routing, Modulation and Spectrum Assignment (RMSA) problem is an extension of RSA, where the additional requirement of selecting the required modulation format among the available ones is added.

1.3 Research Objective and Contributions

In the following sections, we first describe our objectives for the problem stated in Section 1.2, then present our contributions to this thesis.

1.3.1 Objective.

Our main objective in this thesis is to design an intelligent agent with a global vision on the network's activities, and great adaptability, that will perform the provisioning tasks. As an exploratory project, we want to see whether reinforcement learning can be used to improve the heuristic-based method used in the most popular and heavily optimization algorithms that only apply fixed RMSA policies or rely on simple empirical policies based on manually extracted features and therefore are unable to achieve real adaptive provisioning in EONs. This was motivated by the success of several recently proposed Reinforcement Learning (RL) based approaches to solve optimization combinatorial problems. One of the advantages of the RL methods is that they do not require prior knowledge of the underlying system dynamics and the system designer is free to choose reward metrics that best match the final objective.

1.3.2 Contributions.

The contribution of the thesis includes:

- Proposing a heuristic algorithm that will be used as a baseline. This heuristic achieves very good solutions with remarkably high granted average throughput of 100% on the testing scenarios,
- Proposing a new feature engineering representation that provides relevant information to the RL agent,
- Designing a simulator of the network provisioning scenario in EONs that was used as the environment in with the RL agent evolves,

- Designing an RL agent with great adaptability that finds a near-optimal policy to provision the request in a large-scale RMSA problem with static traffic. The REINFORCE agent achieved a granted average throughput of 96% in the worst situation and 100% on some topologies for different traffic matrices and rate distribution.

1.4 Thesis Organization

In this chapter, we started with an introduction to the subject area and scope of research. Thereafter, we introduced the specific problem; RMSA problem and motivated our choice of the proposed approach; reinforcement learning. We have also outlined in the previous section the objectives of this study as well as our contribution. Given this introduction, latter parts of the work are structured as follows:

Chapter 2 presents the preliminary materials and a literature review on the related subjects. Starting with a brief introduction to the concepts and fundamentals behind RL and Deep-RL, we will then proceed by a short review of EONs. The notion of spectrum slots and Modulations will be discussed, followed by an examination of the works in the Literature that will lead to the resolution of the RMSA Problem.

Chapter 3 will state the RMSA problem as well as the mathematical formulation of it. This will be followed by a general description of the entire proposed RL framework as well as the description of the heuristic baseline algorithm.

Chapter 4, finally, conducts a numerical analysis of the performance of the designed algorithms in the previous chapters, as well as the characteristics of the solutions.

This thesis will be ended with a conclusion as well as the statement of future lines of research.

2. Literature Review and Fundamentals

In this chapter, we introduce concepts and fundamentals behind RL and EONs. We first study the different elements of RL and present some RL algorithms. Then, we review the studies on provisioning strategies in EONs followed by the works done in the literature to solve the RMSA problem.

2.1 Reinforcement Learning

Machine learning is a subarea of artificial intelligence that study how computer algorithms can learn from data and improve their performance in making predictions and decisions. Reinforcement learning (RL) is one of the three basic machine learning paradigms, alongside supervised learning (task of learning a function that maps an input to an output based on example input-output pairs) and unsupervised learning (task that helps find previously unknown patterns in data without pre-existing labels) that is concerned with making sequences of decisions [13].

RL is the task of deciding from experience, the sequence of actions to perform in an uncertain environment in order to achieve some goals. In other words, RL is learning what to do and how to map situations to actions so as to maximize a numerical reward signal. The learner (that we call here agent) is not told which actions to take but instead must discover which actions yield the most reward by trying them [29]. As shown in Figure 2.1, It considers an agent situated in an environment: at each time step, the agent observes the state of the environment and takes an action, then the agent receives a reward which is a measure of the consequence of its action. The environment changes into a new state. Using the experience gathered, the goal of the artificial agent is to learn how to take actions in order to optimize some objectives given in the form of cumulative rewards.

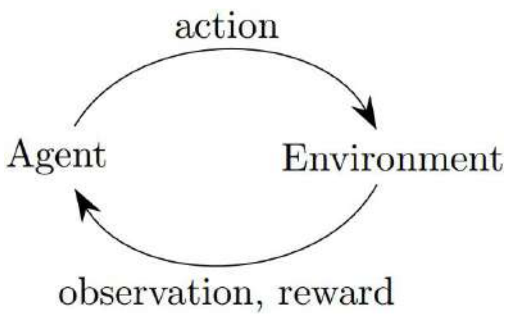


Figure 2.1: Agent, environment interaction (adopted from [29]).

2.1.1 Markov Decision Processes.

The interaction between the agent and its environment is described using ideas from dynamical systems theory, specifically, as a discrete-time stochastic control process (a Markov Decision Process (MDP) [2] in this case).

Definition 1. The MDP consists of a tuple of 5 elements $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ where:

- $\mathcal{S}$ is the state space. At each time step t, the agent receives some representation $S_t \in \mathcal{S}$ of the environment's state.

- $\mathcal{A}$ is the action space. At each time step t, based on its observation of the state, the agent selects an action $A_t \in \mathcal{A}(s)$, where $\mathcal{A}(s)$ is the set of allowed actions of state s.
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ is the transition function, defined by a set of conditional transition probabilities $p(S_{t+1} \mid S_t, A_t)$ between states. They describe how the environment state changes (when in a state S_t the agent performs an action A_t, it finds itself in a new state S_{t+1}).
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow \mathcal{R}$ is the reward function where $\mathcal{R} \subset \mathbb{R}$ is the set of possible rewards. After performing an action, one-time step later, as a consequence of its action, the agent receives a numerical reward $R_{t+1} \in \mathcal{R}$. This is represented by the reward model $p(R_{t+1} \mid S_t, A_t)$ that describes the real-valued reward value that the agent receives from the environment after performing an action. The reward signal is the primary basis for altering the policy; if an action selected by the policy is followed by a low reward, then the policy may be changed to select some other action in that situation in the future.
- $\gamma \in [0, 1)$ is the discount factor that controls the importance of future rewards. It has to be less than 1 so that the cumulative reward is bounded in the infinite horizon reward model. Making $\gamma = 0$ makes the agent being concerned only with maximizing the immediate rewards.

Definition 2. A discrete-time stochastic control process is Markovian (meaning that it has the Markov property) if:

$$p(S_{t+1} \mid S_t, A_t, \dots, S_0, A_0) = p(S_{t+1} \mid S_t, A_t), \quad \text{and} \tag{2.1.1}$$
$$p(R_{t+1} \mid S_t, A_t, \dots, S_0, A_0) = p(R_{t+1} \mid S_t, A_t). \tag{2.1.2}$$

The Markov property means that the future of the process only depends on the current state, and the agent has no interest in looking at the full history, meaning that the current state and action capture all relevant information from the history and provides sufficient information to describe the distribution of immediate reward and next state.

Figure 2.2 represents the interaction of the agent and the environment in a Markov decision process as described above.

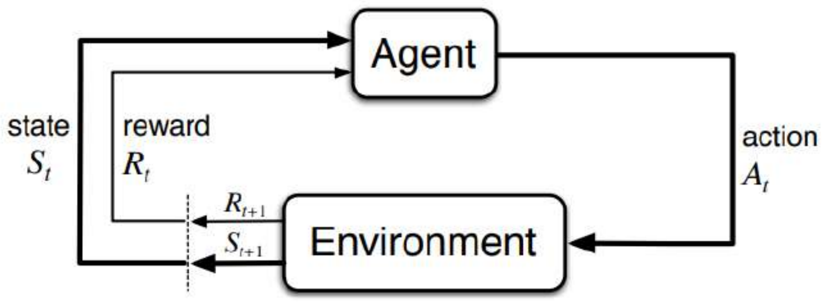


Figure 2.2: The agent-environment interaction in an MDP (extracted from [29]).

It is important to precise that we use R_{t+1} instead of R_t to denote the reward due to action A_t because it emphasizes that the next reward and the next state R_{t+1} and S_{t+1}, are jointly determined.

2.1.2 Discounted Expected Reward.

When training an effective agent, the goal is to find a policy π, that maps perceived states of the environment to actions to be taken when in those states in order to maximize the total amount of reward it receives in the long run term instead of just caring about the immediate return. This expected value of the accumulated discounted reward from time-step t is called **discounted return** and according to [29] is given by:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.1.3}$$

The discount factor γ determines the present value of future rewards.

2.1.3 Policy and Value Functions.

Definition 3. A policy π tells the agent how to behave. It models a probability distribution $\pi(a \mid s)$ over the number of available actions $a \in \mathcal{A}(s)$ for each state s, meaning that $\pi(A_t \mid S_t)$ is the probability of taking action A_t in state S_t.

Definition 4. A value function of a state s under a policy π denoted $V_\pi(s)$, is an estimate of how good it is for the agent to be in state s. $V_\pi(s)$ is the expected return when starting in s and following the policy π. For MDPs, [29] defines $V_\pi : \mathcal{S} \longrightarrow \mathbb{R}$ by:

$$V_\pi(s) = E_\pi[G_t \mid S_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \text{ for all } s \in \mathcal{S}, \tag{2.1.4}$$

where $E[.]$ denotes the expected value of the random variable given that the agent follows the policy π, and t the time step.

The equation (2.1.4) can be formulated recursively as shown in equation (2.1.8). This recursive form is called Bellman equation for V_π, in this equation (2.1.8), the value of the state s is only dependent on the next possible states s' while each state is weighted by the transition probability $p(s' \mid a, s)$.

$$\begin{align} V_\pi(s) &= E_\pi[G_t \mid S_t = s] \tag{2.1.5}\\ &= E_\pi\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \tag{2.1.6}\\ &= \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a)\left[R(s' \mid s, a) + \gamma E_\pi[G_{t+1} \mid S_{t+1} = s']\right] \tag{2.1.7}\\ &= \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a)[R(s' \mid s, a) + \gamma V_\pi(s')] \tag{2.1.8}\end{align}$$

In addition to the V-value function, the value of taking action a in the state s under policy π, denoted by $Q_\pi(s, a)$, is defined as the expected return starting from s, taking the action a, and thereafter and is called Q-value function. It estimates how good it is to take action a in state s. According to [29], the Q-value function $Q_\pi : \mathcal{S} \times \mathbb{A} \longrightarrow \mathbb{R}$ is defined as follows:

$$Q_\pi(s, a) = E_\pi[G_t \mid S_t = s, A_t = a] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}. \tag{2.1.9}$$

Reinforcement learning aims to find an optimal policy $\pi^\star$. According to [29], a policy π is better than another one π', (we note $\pi > \pi'$) if the value function of the policy π is better than the one of π' ($V_\pi(s) > V_{\pi'}(s)$) for all $s \in \mathcal{S}$. The optimal state-value function $V^\star$ can be defined as follows:

$$V^\star(s) = \max_\pi V_\pi(s), \text{ for all } s \in \mathcal{S}. \tag{2.1.10}$$

Similarly to V-value function, the optimal Q-value function $Q^\star(s,a)$ is defined as

$$\begin{aligned} Q^\star(s,a) &= \max_\pi Q_\pi(s,a), \quad \text{for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}, & (2.1.11)\\ &= E\left[R_{t+1} + \gamma V^\star(s') \mid S_t = s, A_t = a\right]. & (2.1.12)\end{aligned}$$

The optimal value- and action-value functions are connected by the following equations:

$$V^\star(s) = \max_{a \in \mathcal{A}} Q^*(s,a), \quad s \in \mathcal{S}. \tag{2.1.13}$$

The particularity of the Q-value function as compared to the V-value function is that the optimal policy can be obtained directly from $Q^\star(s,a)$ as:

$$\pi^\star(s) = \arg\max_{a \in \mathcal{A}} Q^\star(s,a). \tag{2.1.14}$$

2.1.4 Reinforcement Learning Algorithms.

a) Learning approaches

- **Monte Carlo Approach (MC):** This approach is defined only for episodic tasks meaning that we assume experience is divided into episodes, and that all episodes eventually terminate no matter what actions are selected. In such an approach, the value estimates $V_\pi(S_t)$ are computed only at the end of the episode. So the policies changed just after the completion of an episode, see equation (2.1.15). Hence, to estimate the value function from experience, Monte Carlo methods sample and average returns for each state-action pair. An important fact about Monte Carlo methods is that the estimates for each state are independent; the estimate for one state does not build upon the estimate of any other state. Monte Carlo methods are particularly attractive when one requires the value of only one state or a subset of states. The simplest Monte Carlo makes the update as follows:

$$V(S_t) \longleftarrow V(S_t) + \alpha[G_t - V(S_t)], \tag{2.1.15}$$

  where G_t is the actual return following time t, and α a constant step-size parameter.

- **Temporal Difference Learning Approach (TD):** The Temporal Difference approach, on the other hand, will not wait until the end of the episode to update the maximum expected future reward estimation: it will update its value estimation for the non-terminal states occurring at that experience. The simplest TD method makes the update immediately on the transition to S_{t+1} and receiving R_{t+1}. In effect, the target for the Monte Carlo update is G_t, whereas the target for the TD update is $R_{t+1} + \gamma V(S_{t+1})$:

$$V(S_t) \longleftarrow V(S_t) + \alpha[R_{t+1} + \gamma . V(S_{t+1}) - V(S_t)]. \tag{2.1.16}$$

b) Learning algorithms

- **Q-learning** is an off-policy, model-free value-based reinforcement algorithm. Off-policy means that the agent estimates the return for state-action pairs following a different policy from the current policy that was used to take the action. Model-free means that the algorithm estimates the optimal policy without using or estimating the dynamics (transition and reward functions) of the environment. In contrast, model-based algorithm uses the transition function (and the reward function) in order to estimate the optimal policy. As explained in [29], Q-learning algorithm finds the agent update its Q-function using rewards as follows:

$$Q_\pi(s_t, a_t) = Q_\pi(s_t, a_t) + \alpha[R_{t+1} + \gamma . \max_a Q(s_{t+1}, a) - Q_\pi(s_t, a_t)], \tag{2.1.17}$$

  where α is the learning rate. Low values for α increases the accuracy of learning but slows down the convergence.

- **SARSA (State Action Reward State Action)** is an on-policy, model-free algorithm value-based method. On-policy means that the agent estimates the return for state-action pairs assuming that the current policy continues to be followed. This process is commonly described by the equation (2.1.18):

$$Q_\pi(s_t, a_t) = Q_\pi(s_t, a_t) + \alpha[R_{t+1} + \gamma . Q_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t)], \tag{2.1.18}$$

  where a_{t+1} is the action that the agent should take at the next time step following the policy π.

2.2 Deep Reinforcement Learning

All the RL algorithms described in section 2.1.4 have a tabular setting, leading to essential disadvantages. The usage of a table limits the approaches to tasks with a low number of states and actions. In real-world problems, the state space can quickly get large. It is not feasible to visit all possible states to retrieve the value for all action-state pairs. Furthermore, the size of the table is limited due to memory constraints in hardware. To overcome the mentioned restrictions, a common approach is to replace the value table with a Deep Neural Network as a function approximator. Their ability to approximate nonlinear functions and to extract relevant features from raw inputs makes it possible to generalize over unseen states.

2.2.1 Artificial Neural Networks.

An artificial neuron (also called perceptron) models the biological neuron in a simplified way. Each artificial neuron has n input connections. The neuron processes the inputs by taking the weighted sum, adding a bias b and applying an activation function: $f(\sum_i^n \theta_i x_i + b)$. Figure 2.3 illustrates the parallels of a biological and an artificial neuron.

The commonly used activation functions are: sigmoid, tanh and ReLu and are respectively given by the following equations (2.2.1), (2.2.2), and (2.2.3):

$$sigm(x) = \frac{1}{1+e^{-x}}, \tag{2.2.1}$$

$$tanh(x) = 2.sigm(2x) - 1, \tag{2.2.2}$$

$$ReLu(x) = max(0, x). \tag{2.2.3}$$

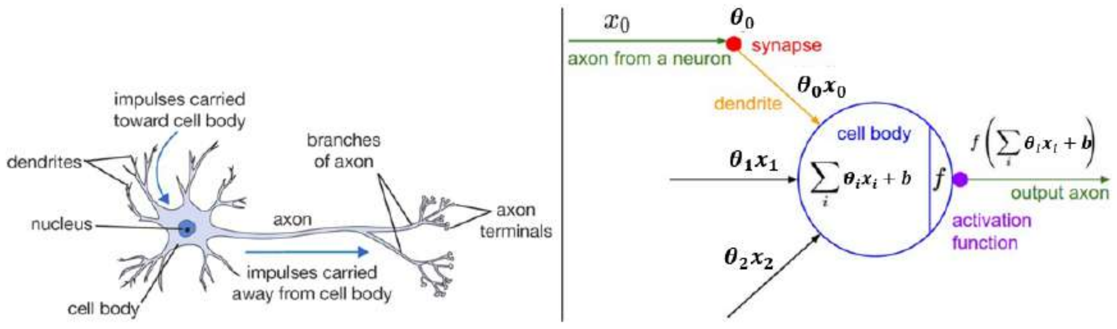


Figure 2.3: Biological neuron (left) vs. artificial neuron (right). The artificial neuron models the dendrites as weighted inputs and processes the sum through an activation function f (extracted from [14]).

The goal of the learning process is to find a parameter set θ and b that results in the best possible function approximation. In supervised learning, the true output Y of the input X is given and can be used to update the parameters θ and b. It is an iterative process, consisting of the following steps:

Step 1. **Forward Pass**: The input X is forwarded through the network and one gets the predicted output $Y_{pred} = f(X, Y, L)$.

Step 2. **Loss**: The predicted output Y_{pred} is compared to the true output Y by computing the loss $L(\theta)$. The choice of loss depends on the learning task. Relevant loss functions are :

- Mean-square-error [27] used for regression problems: It computes the $L2$-distance square between Y_{pred} and Y;

$$L(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (Y_i - Y_{pred,i})^2. \tag{2.2.4}$$

- Cross-entropy Loss function [27] used for classification problems.Given by

$$L(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{c} Y_i^{(j)} log Y_{pred,i}^{(j)}. \tag{2.2.5}$$

In equation (2.2.4) and (2.2.5), m represents the number of input and c the size of the output (number of classes).

Step 3. **Back-propagation and Update:** The global gradient of loss $\nabla_\theta L(\theta)$ is computed and back-propagated through the network. Hence, the weights of all neurons are updated. A common optimizer is stochastic gradient descent (SDG). Gradient descent changes the weights in the negative direction of the gradient of loss so that the function approximation approaches closer to the minimum in each iteration. Equation (2.2.6) shows the corresponding update rule of gradient descent. α is the learning rate parameter that defines how quickly the minimum should be approached.

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta_t} L(\theta). \tag{2.2.6}$$

Other most used optimizers are: Adam, RMSProb, Ada-grad, Adadelta (more details in [14]).

2.2.2 Convolutional Neural Networks.

Convolutional Neural Networks (CNN) can be seen as an artificial neural network particularly adapted to $2D$ and $3D$ signal or data processing. CNNs were inspired by Hubel and Wiesel's work [9] in the receptive field in the brain, that processes sensor input data and is sensitive to certain stimuli, e.g., edges in the visual system. They handle large input data efficiently and are consequently widely used in state-of-the-art approaches in the fields of Computer Vision like e.g. object detection [25, 26, 18] or image segmentation [21, 8].

Figure 2.4 shows the LeNet-5 [17] that recognizes digits in images. It provides a typical architecture of CNN, consisting of stacks of Convolutional Layers, followed by a subsampling Pooling Layer. The final hidden layers of the network are normally fully-connected to compute the final low-dimensional output of the network. It can be assumed that in the early stages of the network low-level features like edges and corners are learned, while in later layers those features are combined to high-level features.

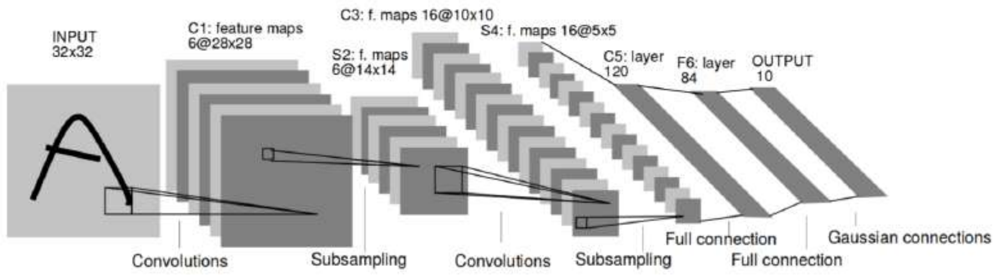


Figure 2.4: To illustrate a typical architecture for CNN, the LeNet-5 [17] is presented. It recognizes digits on images. The input image is processed by two stacks of Convolutional Layers, each followed by a subsampling Pooling Layer. The last three layers are fully-connected to map the high-level features to the final digit classification.

- **The Convolutional Layer** builds on the discrete convolution operation, that applies a filter f of the size $[w \times h]$ (its values represent the weights θ of the neural network) to an input matrix X of size $[W \times H]$ at position $[i, j]$ by computing the dot product. The output is of size $[W - w + 1 \times H - h + 1]$. Considering the notation $X^l[i, j]$, where $X[i, j]$ is the element of the input matrix X at the position $[i, j]$ at the output of the layer l and $\theta[i, j]$ the weight at position $[i, j]$ on the filter f. The convolution operation is given by the discrete cross-correlation shown in equation (2.2.7) and Figure 2.5.

$$X^l[i,j] = \sum_{u=0}^{w-1}\sum_{v=0}^{h-1} \theta[u,v].X^{l-1}[i+u, j+v] \tag{2.2.7}$$

  To produce an output that has the same size as the input, a technique called **zero padding** can be applied. Zero padding extends the input matrix by adding zero-values on each input's side. It is common to shift the filter with a constant **stride** S over the input so that every S^{th} position of the input will be convolved. It results in a reduction of the data size in the next layer.

- **Pooling Layer :** The Pooling Layer has a sub-sampling function in the spatial dimensions width and height by applying a down-sampling filter to the input. Common pooling filters are max- and average-pooling. At max-pooling, a filter of size $[w \times h]$ slides over the input and only the

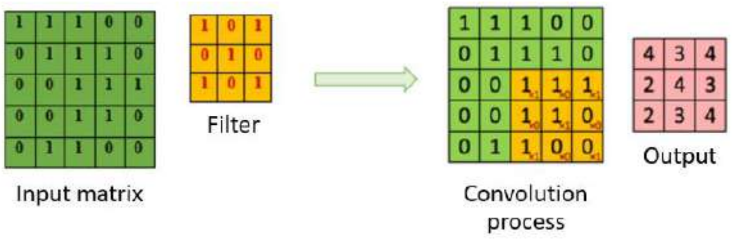


Figure 2.5: Convolution process.

maximum value remains in the output. Figure 2.6 provides an example: A $[2 \times 2]$- filter is shifted over the 2-dimensional input with a stride of 2. The resulting output size is a quarter of the input size. In average-pooling, the average of each position $[x, y]$ and its neighbors is computed by the filter.

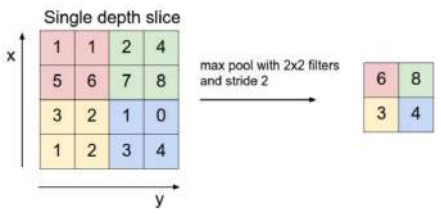


Figure 2.6: A max-pooling filter of size $[2 \times 2]$ with a stride of 2 is applied to a 2- dimensional input with the size $[4 \times 4]$. The maximum value remains in the output and the output size is reduced by 4 (extracted from [14]).

- **Fully Connected Layer** Finally, after several layers of convolution and pooling, the high-level reasoning in the neural network is done via fully connected layers. In CNN, each layer acts as a detection filter for the presence of specific characteristics or patterns present in the original data. The first layers of a convolution network detect characteristics that can be recognized and interpreted relatively easily. Subsequent layers increasingly detect more abstract characteristics. The last layer of the convolution network is able to do an ultra-specific classification by combining all the specific characteristics detected by the previous layers in the input data.

2.2.3 Value-Based Methods.

Value-based methods are more interested in "Value", which as explained in Section 2.1.3 estimates the expected reward for different actions given the initial states or simply the expected reward for different states. The idea is to replace the value table one-to-one and to approximate it with a Deep Neural Network (DNN). The DNN output provides the estimated value for each possible action.

Deep Q-Network (DQN)

deep-Q-Network (DQN) is an approach presented in 2015 by the DeepMind group [20] that showed great success. This approach combines Q-Learning with Deep Neural Networks. As input data, high-dimensional raw sensory input with no previously hand-crafted features are used. And the neural network end-to-end architecture extracts relevant features by itself. The output of the Q-network provides the estimated values of the possible discrete actions. This allows one to determine the best action for a given state with a single forward pass. This approach encountered a problem of unstable learning that has been solved with an additional mechanism called *experience replay*.

The idea of *experience replay* is to store the agent's experiences $S_t, A_t, R_{t+1}, S_{t+1}$ in a buffer that can hold N_{buffer} experiences in total. In each training step, a batch m of experiences is randomly sampled from the buffer and fed to the network. Hence, experience replay removes the correlations in the data sequences and feeds the network with independent data.

In DQN the network is updated using the equation (2.2.6) according to the loss function from equation (2.2.8).

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m} L_i(\theta_i), \tag{2.2.8}$$

$$\text{where} \quad L_i(\theta_i) = (R_{t+1} + \gamma . \max_a Q(s_{t+1}, a, \theta_i) - Q(s_t, a_t, \theta_i))^2. \tag{2.2.9}$$

In the equation (2.2.9), $R_{t+1} + \gamma . \max_a Q(s_{t+1}, a, \theta_i)$ represents the target value and $Q(s_t, a_t, \theta_i)$ the output of the DQN.

There exist many other variants of DQN like Double DQN [32] and Dueling DQN [36] that have not been studied in this thesis.

2.2.4 Policy Gradient Methods.

The goal of the policy gradient method is to directly optimize the policy function $\pi(a \mid s, \theta)$ instead of learning a value function and choosing the actions based on it. The quality of each policy can be measured by the policy's performance measure $J(\theta)$. The objective function of this method in equation (2.2.10) maximizes the scalar value $J(\theta)$.

$$\theta^{\star} = arg \max_{\theta} J(\theta). \tag{2.2.10}$$

The policy's parameter θ is updated via gradient ascent. Gradient ascent is the inverse of gradient descent and updates the parameters θ_t in the positive direction of the gradient of the policy's performance measure $\nabla_{\theta} J(\theta)$ (see equation (2.2.11)). Furthermore, the learning rate α defines, how strongly one step is in the gradient's direction.

$$\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} J(\theta_t). \tag{2.2.11}$$

Compared to the value-based method, the advantages of Policy Gradient Methods can be resumed as follows:

- Policy Gradient Methods have stable convergence property because the policy is updated directly and thus improves smoothly at each time step. With the value-based method, it is the value function that is updated at each time step and a small change in the value function can lead to a drastic change in the policy output. This makes the value-based methods often deal with big oscillations during training.

- Policy Gradient Methods can deal with infinite and continuous action spaces. Instead of determining a Q-value for each possible discrete action, the action can be estimated directly, e.g. the rotating angle of a mobile robot is estimated directly by the agent.
- Policy Gradient Methods have the ability to learn stochastic policies, i.e., actions are chosen with certain probabilities. It is especially necessary for uncertain, partially observable environments.

The big disadvantage of Policy Gradient Methods is that they rather converge to a local optimum than to the global optimum [29].

REINFORCE algorithm

REINFORCE is a Monte-Carlo variant of policy gradients introduced by [38] in 1992. The agent collects a trajectory τ (state-action sequence $s_0, a_0, \ldots, s_H, a_H$ where H is the length of the sequence) of one episode using its current policy, and uses it to update the policy parameter. The policy performance measure $J(\theta)$ in equation (2.2.12) is defined by the expected return of all trajectories τ. The contribution of each trajectory τ to the expected return is the product of the cumulative reward $R(\tau)$ and the probability of its occurrence $p(\tau \mid \theta)$ under policy π_θ.

$$J(\theta) = E[\sum_{t=0}^{H} R(s_t, a_t) \mid \pi_\theta] = \sum_{\tau} p(\tau \mid \theta) R(\tau), \tag{2.2.12}$$

where $R(\tau) = \sum_{t=0}^{H} R(s_t, a_t)$ is the total reward the agent obtains after performing the trajectory τ. And the goal is to find $\theta^\star$ (see equation (2.2.13)) that creates the trajectory τ that maximizes the expected rewards $J(\theta)$.

$$\theta^\star = arg \max_{\theta} \sum_{\tau} p(\tau \mid \theta) R(\tau). \tag{2.2.13}$$

To achieve this goal, the parameters θ are updated using the gradient ascent as shown in equation (2.2.14):

$$\theta = \theta + \alpha \nabla_\theta J(\theta), \tag{2.2.14}$$

where α is the learning rate and the gradient of $J(\theta)$, $\nabla_\theta J(\theta)$ of one trajectory can be determined by applying the Policy Gradient Theorem that has been derived in [29]. This results in equation (2.2.15)

$$\nabla_\theta J(\theta) = E[\nabla_\theta log \pi_\theta(a \mid s).R(\tau)], \tag{2.2.15}$$

and the global gradient for all the episodes is given by equation (2.2.16):

$$\nabla_\theta J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)}) R(\tau^{(i)}), \tag{2.2.16}$$

where N is the total number of episodes. The REINFORCE method can be represented as in Figure 2.7.

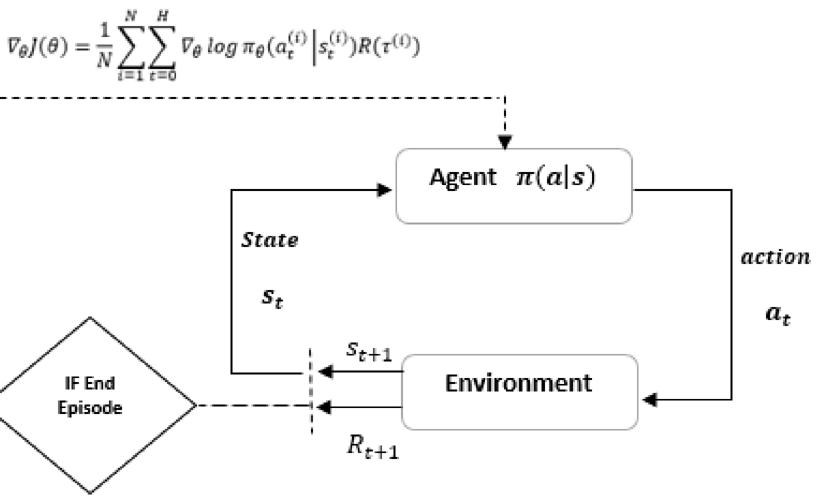

$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{H}\nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$
Agent $\pi(a|s)$
State
s_t
action
a_t
s_{t+1}
Environment
R_{t+1}
IF End Episode

Figure 2.7: Agent-environment interaction in REINFORCE (inspired from [10]).

Vanilla Policy Gradient Algorithm

REINFORCE suffers from high variance and low convergence. The variance provides conflicting descent direction for the model to learn. One sampled reward may want to increase the log-likelihood and another may want to decrease it. This hurts the convergence. One way to reduce variance and increase stability is subtracting the cumulative reward by a baseline. Vanilla Policy uses a baseline $b(s_t)$ to solve this issue and the equations become:

$$\theta = \theta + \alpha \nabla_\theta J(\theta), \tag{2.2.17}$$

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{H}\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})(R(\tau^{(i)}) - b(s_t)). \tag{2.2.18}$$

Intuitively, making the cumulative reward smaller by subtracting it with a baseline will make smaller gradients, and thus smaller and more stable updates. Common baseline functions are Q-value, Advantage, and TD-error (more details in the following section 2.2.5).

2.2.5 Hybrid Methods: Actor-critic.

This approach combines value-based and policy-based methods. Two entities appear in the formulation of the Actor-Critic Methods, The **Critic** and the **Actor**. The critic estimates measures how good the action taken in a given state is (Q-value) or how good it is to be in a state (V-value). The actor controls how the agent behaves, it uses a policy gradient-based approach and updates the policy distribution in the direction suggested by the Critic. Both the critic and actor functions are parameterized with neural networks. For the one step approach, on each learning step, both the actor parameters θ and the critic parameters ω are updated. Because we do an update at each time step, the total reward $R(t)$ can't be used. Instead, we need to train the critic model that approximates the utility function. This utility function replaces the reward function in the REINFORCE approach. Hence, the actor parameters and critic parameters are respectively updated using following equations (2.2.19) and (2.2.21):

$$\theta \quad = \theta + \alpha_\theta \Delta\theta, \quad \text{where} \tag{2.2.19}$$

$$\Delta\theta = \hat{U}_\omega(s,a).\nabla_\theta(\log \pi_\theta(s,a)). \tag{2.2.20}$$

$$\omega \quad = \omega + \alpha_\omega \Delta\omega, \quad \text{where} \tag{2.2.21}$$

$$\Delta\omega = (R(s,a) + \gamma\hat{Q}_\omega(s_{t+1},a_{t+1}) - \hat{Q}_\omega(s_t,a_t)).\nabla_\omega\hat{Q}_\omega(s_t,a_t). \tag{2.2.22}$$

α_θ and α_ω are the learning rates of the actor and the critic, respectively. $\hat{Q}$ is the Q function approximation of the critic. The variable $\hat{U}$ depends on the type actor-critic method and is defined as follows:

$$\hat{U} = \begin{cases} \hat{Q}, & \text{method is called Q Actor-critic (QAC)} \\ \hat{A}, & \text{method is called Advantage Actor-critic (A2C)} \\ \hat{\sigma} = \text{TD error}, & \text{method is called TD Actor-critic} \end{cases}$$

with

$$\hat{\sigma} = \text{TD error} = R(s,a) + \gamma\hat{Q}_\omega(s_{t+1},a_{t+1}) - \hat{Q}_\omega(s_t,a_t), \tag{2.2.23}$$

and the advantage

$$\hat{A}(s,a) = \hat{Q}(s,a) - V(s), \tag{2.2.24}$$

where $V(s)$ in the average value of the state s.

The actor-critic learning process is represented in Figure 2.8.

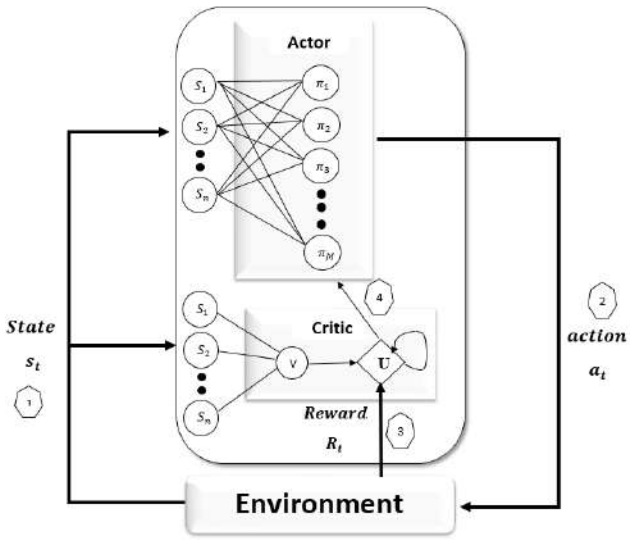


Figure 2.8: Actor-Critic learning process (inspired from [29]).

2.3 EONs Provisioning Strategies

Optical fibers are largely deployed in fiber-optic communications, as they enable transmission over longer distances and at higher bandwidths (data rates) than wire cables. These properties are due to lower losses and a higher number of channels that can be simultaneously transported over their large spectrum window. In an optical fiber, the region between 1.3 and 1.6 μm is exploited for transmission. Within this region, the C band presents the lowest losses of the whole fiber spectrum and is exploited for transmitting over very long distances (from tens to thousands of kilometers). The C band refers to the wavelengths around 1550 nm and includes wavelengths between approximately 1525 nm (or a frequency of 195.9 THz) and 1565 nm (191.5 THz). Wavelength Division Multiplexing (WDM) denotes the technology enabling the transmission of a number of optical signal carriers onto a single optical fiber by using different wavelengths (see Figure 2.9).

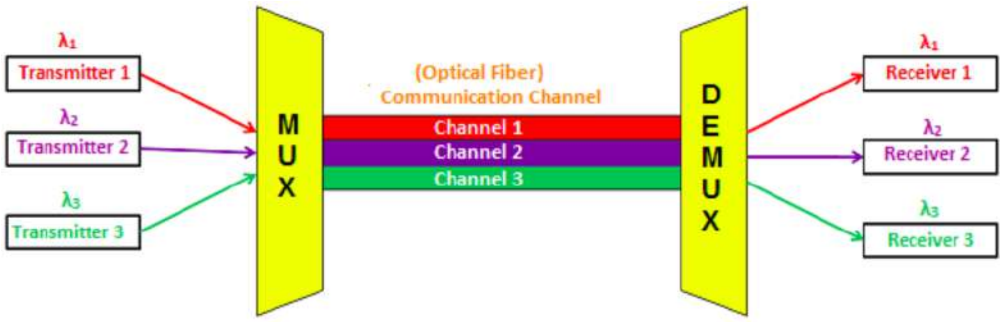


Figure 2.9: Wavelength Division Multiplexing (WDM (extracted from [24]).

The WDM spacing between two adjacent central frequencies is fixed and is 50 GHz, which is specified by the International Telecommunication Union (ITU)-T standards [11]. WDM usually has fibers that carry 50 GHz wavelengths with bandwidths up to 100 Gbps. As shown in Figure 2.10, if a request connection only requires a fraction of the available bandwidth of a channel, that channel would not be used efficiently since there would be a big wasted spectrum in it and no other requests would be able to use it for their transmission.

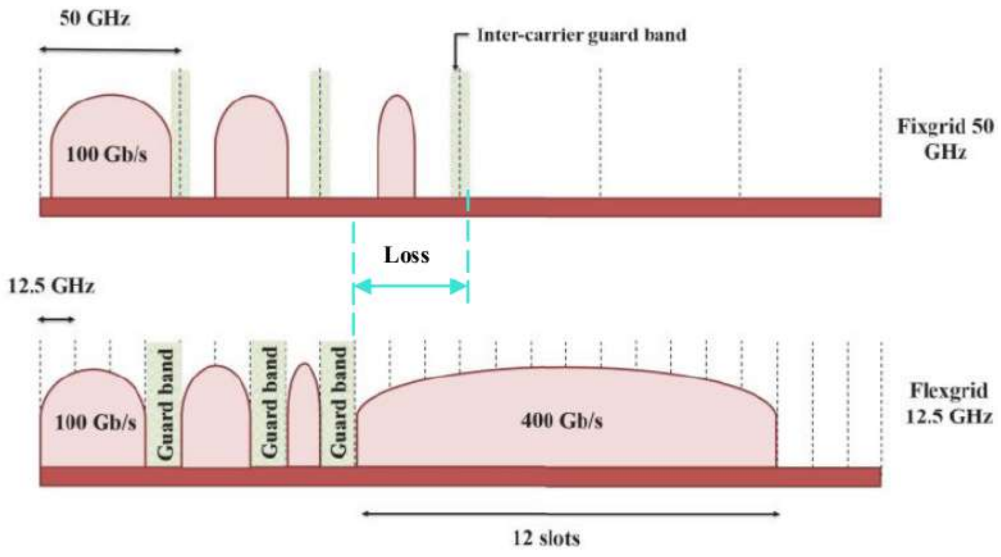


Figure 2.10: Fix and flexible frequency grids.

To face the ceaseless traffic growth in the last decade, mechanism that allows for serving high bit-rate requests as big as 400 Gbps and 1 Tbps is needed; It is provided by Elastic Optical Network that has the capability to slice the spectrum into frequency slot of width granularity of 12.5 GHz with 6.25 GHz central frequency granularity in contrast to the coarser 50 GHz width in the fixed grid. The flexible grid gives the ability to define an aggregate super-channel spectral width of N$\times$ 12.5 GHz to accommodate any combination of optical carriers, modulation formats, and data rates. This supports provisional modulation formats, which allow operators to balance their requirements for increased spectral efficiency versus extended reach of the optical signals. In addition, the flexible grid provides the capability to elastically allocate frequency slots on demand and/or modify the modulation format of optical channels according to traffic demands. This allows resources to be used efficiently in response to traffic variations. Figure 2.11, below, shows an example of a flexible grid spectrum allocated to 400G and 1Tb super-channels along with today's 50 GHz 100G channels.

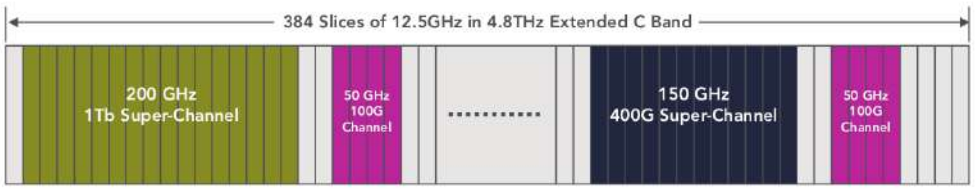
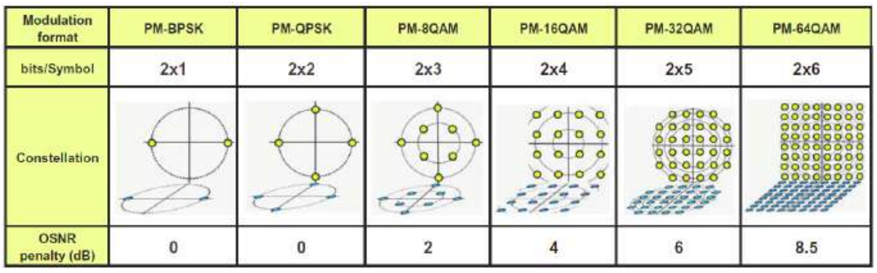


Figure 2.11: ITU-T G.694.1 Flexible Grid Architecture (extracted from [28]).

ITU-T REC G.694.1 [11] defines the following terms that will be used throughout this thesis.

- Frequency slot : The frequency range allocated to a slot and unavailable to other slots within a flexible grid. A frequency slot is defined by its nominal central frequency and its slot width.
- Slot width: The full width of a frequency slot in a flexible grid.
- The baud rate: It is the rate symbols which are generated at the source and, to a first approximation, equals to the electronic bandwidth of the transmission system.
- The modulation format: For a given baud rate the modulation format defines the equivalent number of bits each symbol is transporting. Figure 2.12 presents the encoded bits per symbol and data constellations for common modulation formats. We can see that high-level modulation (16-QAM for instance) has a higher number of bits per symbol. This makes high-level modulation more sensitive to attenuation, and definitely, a high-level modulation with a narrow spectrum and low OSNR (Optical Signal to Noise Ratio) tolerance may be selected for a short path, whereas a low-level modulation with a wider spectrum and high OSNR tolerance may be used for a longer path.



Modulation format	PM-BPSK	PM-QPSK	PM-8QAM	PM-16QAM	PM-32QAM	PM-64QAM
bits/Symbol	2x1	2x2	2x3	2x4	2x5	2x6
Constellation						
OSNR penalty (dB)	0	0	2	4	6	8.5

Figure 2.12: Encoded bits per symbol and data constellations for common modulation formats (extracted from [22]).

2.4 Related Works

In this section, we present the methods generally used to solve the RMSA problem. In the first part, we present some used heuristics, exact methods and ILP methods and in the second part, we focus on methods related to reinforcement learning.

2.4.1 RMSA with OR tools.

Several variants of the RSA problem have been studied in the literature, and take into account various design aspects. Accordingly, a variety of integer linear program(ILP) formulations have been proposed, each tailored to a specific problem variant. Since the problem is intractable, these ILP formulations cannot be solved within a reasonable amount of time for problem instances corresponding to network topologies encountered in practice. Therefore, an array of heuristic algorithms have been put forward to obtain reasonably good solutions efficiently. In [35], Link-based ILP formulations of RSA as a multi-commodity flow problem have been studied and two heuristic algorithms were developed to solve the RSA problem. The first algorithm, referred to as the shortest path with maximum spectrum reuse (SPSR), uses shortest path routing and the first-fit spectrum allocation strategy to assign frequency slots to demands in decreasing order of their size. The second algorithm, balanced load spectrum allocation (BLSA), considers the k shortest paths as candidates for each demand and selects the one that minimizes the maximum link load, so as to balance the use of spectrum across the network links.

A different, path-based ILP formulation of the RSA problem was presented in [15]. In a path-based formulation, k paths are precomputed for each demand, such that the demand may be routed only along one of these paths. The ILP is used to assign one of the predetermined paths to each demand, while also satisfying the spectrum contiguity and non-overlapping spectrum constraints; the spectrum continuity constraint is implicitly satisfied since the assignment of subcarriers is along a whole path. The objective is to minimize the number of sub-carriers that are used in any link in the network. Although this path-based formulation is more compact than the link-based ILP of [35], the number of decision variables and constraints is substantially large that it cannot be solved directly. Accordingly, a heuristic algorithm called adaptive frequency assignment with collision avoidance (AFA-CA) was presented to select the path for each demand. Another path-based ILP formulation for the RSA problem was presented in [4] where the objective is to minimize the maximum subcarrier index assigned on any link in the network. A greedy heuristic algorithm is also presented that processes demands in decreasing order of either their size or their shortest path length; this order is fixed, unlike the heuristic in [15] that adapts the order as the algorithm progresses. A simulated annealing meta-heuristic that builds upon the greedy algorithm was also presented.

2.4.2 RMSA with RL tools.

In our knowledge, a few works addressed the RMSA problem using reinforcement learning. In [3], the author proposes a Deep-RMSA, a deep reinforcement learning-based self-learning RMSA agent, to realize dynamic autonomous and cognitive RMSA for EONs. Using a deep Q-network consisting of multiple convolutions and fully connected layers, the k-shortest paths for each request are computed and the deep Q-network has to choose for each of the request a path between the k. They used the impairment-aware-modulation format adaption and the first-fit spectrum assignment to simplify the model.

3. Proposed Reinforcement Learning Framework for RMSA Problem

The chapter presents the used methods and the concrete training setup. Section 3.1 presents the mathematical formulation of the Routing, Modulation and Spectrum Assignment Problem followed by Section 3.2 that covers in detail the different steps that have been followed to solve the problem. Section 3.3 presents the RL-agent specific setup that includes different observation spaces, action spaces, reward functions, and neural network architectures.

3.1 Mathematical Formulation of the Offline RMSA Problem

3.1.1 Offline RMSA Problem Statement.

An Elastic Optical Network (EON) can be represented by a directed graph $G = (N, L)$, where N is the set of nodes (locations) and L is the set of optical fiber links connecting pairs of locations. The frequency spectrum that is available is defined by the standard ITU-T G.694.1 [11]. According to the standard, the spectrum capacity of each fiber link is defined by a set S of 384 frequency slots with a width of 12.5 GHz ($S = s_1, s_2, \ldots, s_{|S|}$). A channel can be considered as a specific number of contiguous slots (adjacent to each other) on a specific link, see Figure 3.1. The formulation is the following:

Input: Given

i) An EON, represented by a directed connected graph $G(N, L)$,

ii) A set S of frequency slots,

iii) A set D of demands or requests. Each demand $d \in D$ is represented by the tuple $< s_d, t_d, b_b >$ characterized by:

- A source node $s_d \in N$ and a target node $t_d \in N$, such that $(s_d, t_d) \in \mathcal{SD}$, where $\mathcal{SD}$ is the set of source-destination node pairs with traffic demand, $s \neq t$.
- A bit-rate $b \in B$, where B is the set of rates.

Output: The route over the flex-grid optical network and the spectrum allocation of every transported demand. Provisioning a request consists of determining:

i) A path from the source node to the destination node,

ii) An adapted modulation format among the available ones,

ii) A spectrum allocation on every link of the selected path, which must satisfy the continuity and contiguity constraints, (see below for their definitions).

Objective: The offline problem is defined as determining an optimal provisioning of the set of requests D. The two most common objectives are to:

i) Maximize the network throughput, which is defined as the summation of the rates of all granted requests,

ii) Minimize the total amount of used slots.

Note that from the problem definition, demands can be served or alternatively rejected. We choose that objective instead of serving all demands to avoid in-feasibility that may appear when trying to serve large sets of demands over a capacitated network.

Constraints: The provisioning task is subject to the following constraints:

i) **Non-Overlapping Constraint**: A slot can be assigned to only one request on a given fiber link.

ii) **Continuity Constraint**: The same slots are assigned to the path links of a request.

iii) **Contiguity Constraint**: The assigned slots on a link have to be contiguous (adjacent to each other) in the frequency domain.

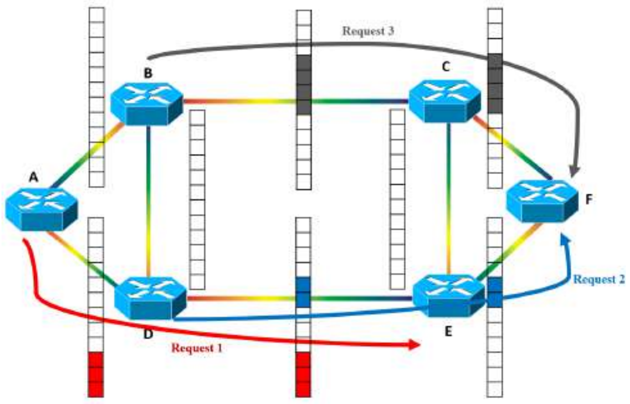


Figure 3.1: Provisioning example.

3.1.2 Mathematical Formulation.

In the following, we present an ILP model for the above problem, based on the formulations in [33]. This formulation based on channel assignment explicitly removes the spectrum contiguity and continuity constraint, however, it takes into account these constraints; Table 3.1 presents the sets and parameters that have been defined for the formulation.

In this formulation, a precomputed set of candidate channels defining a set of contiguous frequency slots is given as an input parameter to the formulation. The definition of the channel can be mathematically formulated as follows. Let γ_{cs} be a coincidence coefficient that is equal to 1 whenever channel $c \in C$ uses slot $s \in S$, and 0 otherwise. Let us assume that a set of channels $C(d)$ is predefined for each demand d, which requests n_d^m slots (m refers to the corresponding modulation). Then, $\forall c \in C(d)$ the spectrum contiguity constraint is implicitly imposed by the proper definition of γ_{cs}, such that

$$\forall s_i, s_j : \gamma_{cs_i} = \gamma_{cs_j} = 1 \;,\; s_i < s_j \Rightarrow \gamma_{cs_k} = 1, \; \forall s_k \in \{s_i, \dots, s_j\}, \; \sum_{s \in S} \gamma_{cs} = n_d^m. \tag{3.1.1}$$

Table 3.1: Terminology and notation of the mathematical formulation.

Notation	**Description**
	Decisions Variables
x_d	Binary. Equal to 1 if demand d is rejected, 0 otherwise.
y_{pc}	Binary. Equal to 1 if channel c is assigned to path p, 0 otherwise.
	Precomputed parameters
$P(d)$	Subset with the predefined candidate paths for demand d. Each path p consists of a set of links $l \in L$ so that nodes s_d and t_d are connected.
C	Set of Channels, index c. Each channel contains a subset of contiguous frequency slots.
$C(d)$	Subset of channels for demand d.
δ_{pl}	Equal to 1 if path p uses link l, 0 otherwise.
γ_{cs}	Equal to 1 if channel c includes frequency slot s, 0 otherwise.
n_d^m	Number of slots to transport the requested bitrate of demand d using modulation m.
$len(l)$	Length in kilometers of link $l \in L$
$len(p)$	Length in kilometers of path $p \in P$, $len(p) = \sum_{l \in L} \delta_{pl} len(l)$
M	Set of modulation formats, index m.
$len(m)$	Reach in kilometers of modulation format m.
q_{cm}	Equal to 1 if channel $c \in C$ is computed for modulation format m, 0 otherwise.

Additionally, the decision variable y_{pc} implicitly imposed the continuity constraint since this variable equals 1 if channel c is assigned to path p. Knowing that a path is a set of links, saying that a channel is assigned to a path means that each of the links of that path uses that channel and therefore, the continuity constraint is satisfied. The number of channels that uses n slots on a link given by $|C(n)| = |S| - (n-1)$ ensures that the capacity of each link is not exceeded.

To account for guard bands, without loss of generality, we consider that they are included as a part of the requested spectrum (n_d^m). Therefore, we can define the problem like one that finds a route and assigns a proper channel to a set of input demands, so that the number of active slots in the channel guarantees that the bitrate requested by each demand can be transported.

The formulation is an optimization problem where we want:

$$\text{maximize} \quad \sum_{d \in D} (1 - x_d).b_d. \tag{3.1.2a}$$

$$\text{subject to} \quad \sum_{p \in P(d)} \sum_{c \in C(d)} y_{pc} + x_d = 1, \qquad \forall d \in D, \tag{3.1.2b}$$

$$\sum_{d \in D} \sum_{p \in P(d)} \sum_{c \in C(d)} \gamma_{cs}.\delta_{pl}.y_{pc} \leq 1, \qquad \forall l \in L,\ s \in S, \tag{3.1.2c}$$

$$\sum_{p \in P(d)} \sum_{c \in C(d)} q_{cm}.len(p).y_{pc} \leq len(m), \qquad \forall d \in D,\ m \in M. \tag{3.1.2d}$$

The objective function (3.1.2a) maximizes the amount of bitrate that is served (accepted). Firstly, Constraint (3.1.2b) is required to ensure that a lightpath (feasible path and channel) is selected for each demand provided that the demand is served; otherwise, the demand cannot be served and is therefore rejected. Secondly, Constraint (3.1.2c) is used to guarantee that every slot in every link is assigned to one demand at the most. Finally, Constraint (3.1.2d) is used to guarantee the selection of a modulation format with enough reachability for the selected path.

3.2 Method Description

Let's recall that we modeled a demand d from the source node s to the target node t with bit-rate b Gbps by a tuple $< s_d, t_d, b_d >$. To provision the demand d, we need to compute an end to end routing path P_{st}, determine a proper modulation format m to be used with a QoT (Quality of Transmission) assurance, and allocate a number of spectrally contiguous slots on each link along P_{st}. Figure 3.2 represents the basic operation of the RL-agent in a network routing scenario.

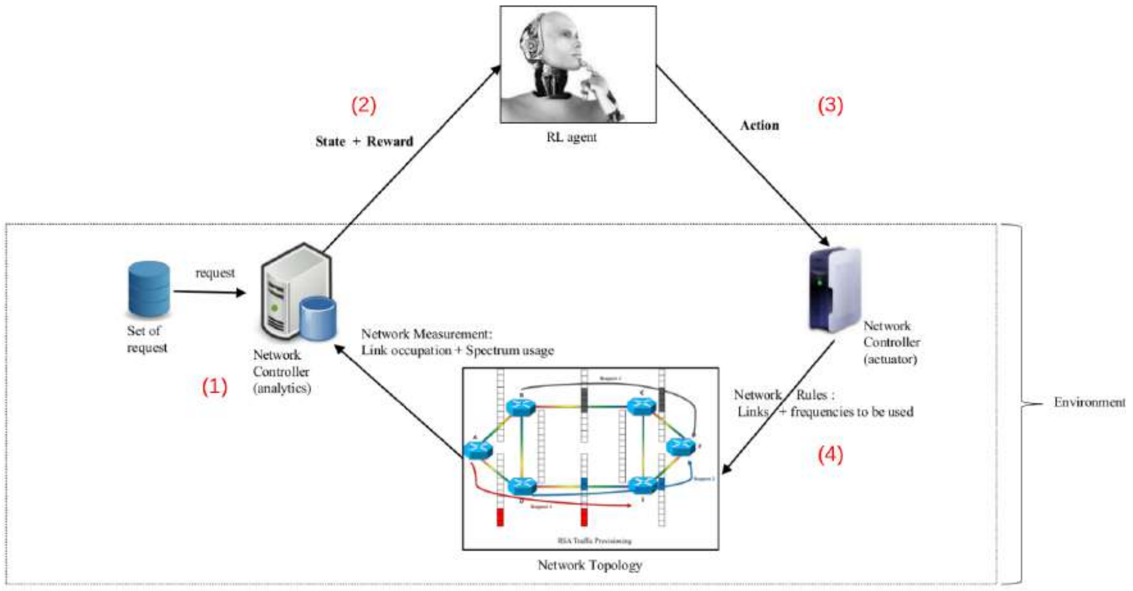

(2)
State + Reward
RL agent
Action
(3)
request
Set of request
Network Controller (analytics)
(1)
Network Measurement: Link occupation + Spectrum usage
Network Controller (actuator)
Network Links
Rules : + frequencies to be used
(4)
Environment
Network Topology

Figure 3.2: Schematic representation of the RMSA-RL operation.

When there is a new set of requests to be routed, this is communicated to a network controller (1). Then, the controller generates a new state representation that will be the input of the RL agent (2). This state representation includes information about the state of the network (optical link occupation and spectrum utilization) and the new traffic request (source, target, and rate). With this input, the RL agent selects an action that involves making a routing decision (3)(choice of a path, a modulation, and frequency slots). Lastly, the controller translates the resulting action to a particular set of network rules that are installed into some network devices (4). This way, during the training phase, the RL agent learns how to properly act over the network by iteratively exploring different routing strategies and considering the reward obtained. To reach that goal, Figure 3.3 describes the process as follows:

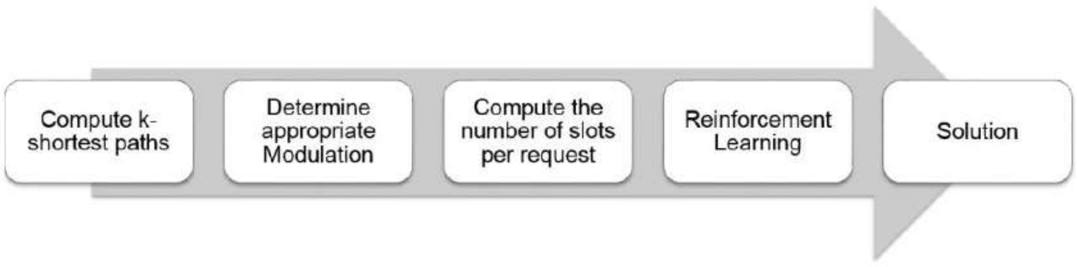

Compute k-shortest paths
Determine appropriate Modulation
Compute the number of slots per request
Reinforcement Learning
Solution

Figure 3.3: Steps to Solve the Problem.

3.2.1 Computation of the K-Shortest paths.

Using the same notation of Section 3.1.1, $G(N, L)$ being our network with $|N|$ nodes and $|L|$ links, where any $(s, t) \in \mathcal{SD}$ is assigned with the value $c_{st} \in \mathbb{R}^+$, that denotes the cost, or distance or length, of (s, t) .

- A path p from $s \in N$ to $t \in N$ in G ($s \neq t$), is a sequence of the form $p =< s = n_1, n_2, \ldots, t = n_{|p|} >$, where $(n_i, n_{i+1}) \in \mathcal{SD}$, for any $i \in \{1, \ldots, |p| - 1\}$. Here $|p|$ represents the hope of p, that is its number of nodes.
- The total cost, or distance, or length of p is defined by:

$$c(p) = \sum_{(s,t)\in p} c_{st} \tag{3.2.1}$$

For each demand or request $d \in D$, a set of K_{st} paths $P(d) = \{p_1, p_2, \ldots, p_{K_{st}}\}$ is calculated (K_{st}-shortest paths) using the Yen's KSP algorithm [39] given in Algorithm 1.

Algorithm 1 Yen's Algorithm.

```
Input:
- A directed graph where each edge has a real-valued positive weight
- A source node $s$ and target $t$ in a directed graph.
- The value of $k$
Output: List of the $k$-shortest path
Begin
 1: Find the shortest path $p^1$
 2: for $k = 2, 3, \ldots$ do, find $p^k$ as follows:
 3:     Let $B^k = B^{k-1}$ , the set of candidate paths from iteration $k - 1$
 4:     for $1 \leq i < |p^{k-1}|$ do
 5:         Let $x = p_i^{k-1}$
 6:         Hide incoming edges to $x$ for the remainder of iteration $k$
 7:         for each $j$ such that the first $i$ nodes in $p^j$ match $p^{k-1}$ do
 8:             Hide edge $(x, p^j_{i+1})$for the remainder of iteration $k$
 9:         end for
10:         $R_i^k$ is the first $i$ nodes of $p^{k-1}$
11:         $S_i^k$ is the shortest path from $x$ to $t$
12:         Join $R_i^k$ and $S_i^k$ to form $D_i^k$
13:         Add candidate path $D_i^k$ to $B^k$
14:     end for
15:     Remove the shortest path from $B^k$ and return it
16: end for
17: return the $k$ Shortest paths
End.
```

The above Yen's algorithm employs any efficient shortest path algorithm (Dijkstra [37] in our case, see appendix A.0.1) to find the best path, then proceeds to find $k - 1$ deviations of the best path. The Yen's algorithm is written using the following terminologies and notations:

- $p^k = \{s, p_2^k, p_3^k, \ldots, p_{|p^k|-1}^k, t\}$ is the k^{th} shortest path from s to t.
- D_i^k is the deviation from p^{k-1} at node p_i^{k-1}; More specifically, the shortest s to t path that:
  - coincides with p^{k-1} from s to p_i^{k-1},
  - deviates to a node n where n is not used as this deviation in any of the $k-1$ shortest paths,
  - reaches t by a shortest path from n without using any node in the first part of the path.
- $R_i^k = \{s, p_2^k, p_3^k, \ldots, p_i^k\}$ is the root of D_i^k.
- $S_i^k = \{p_i^k, \ldots, t\}$ is the spur of D_i^k.

An illustration of R_i^k and S_i^k is given in Figure 3.4.

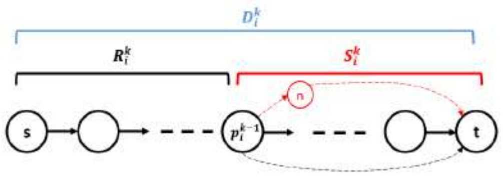


Figure 3.4: Root and Spur Representation.

3.2.2 Determine the appropriate Modulation.

Once the K_{st} shortest paths of each request are determined, the geographical lengths of those paths are computed. Then, the length of each of the paths and the bit-rate of the corresponding request are used to determine the modulation format and the number of necessary slots per channel. Flexible spectral resource allocation can be done as follows: As explained in Section 2.3, since the shortest path experiences the smallest optical SNR degradation and filter narrowing effect, the most spectrally efficient modulation is selected (e.g., 16-QAM). For path having a long distance, a more robust modulation is utilized (e.g., QPSK). Table 3.2 gives reachable distances for each of the modulation.

Table 3.2: Distance-Modulation principle.

Modulation	**Distance (Km)**
BPSK	>4000
QPSK	4000
8-QAM	1200
16-QAM	600

3.2.3 Computation of the number of slots per request.

At this point, knowing the modulation and the bit-rate b_d of the corresponding request d, the number of frequency slot n for the corresponding channel is computed using the formula in equation (3.2.2):

$$n = \frac{b_d}{B.W}, \tag{3.2.2}$$

where $B \in \{1, 2, 3, 4\}$ is the number of bits per symbol of the constellation on each polarization respectively for BPSK, QPSK, 8-QAM and 16-QAM (see Section 2.3) and W the slot bandwidth.

3.3 Reinforcement learning framework

To solve the RMSA problem, three reinforcement learning algorithms (Deep-Q-learning, REINFORCE and Advantage Actor-critic (see Section 2.2)) have been implemented with an agent that acts in the environment behaving like a real network provisioning environment. At each time t, the agent receives the state S_t from the environment, it takes an action A_t and receives a reward R_{t+1}. The agent continues this process $(\ldots, S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, \ldots)$ until the end of the provisioning. In reinforcement learning, the design of the environment, the state space and action space representations as well as the design of reward function are important for the algorithms overall performance. We describe the four key elements as follows.

3.3.1 Design of the RMSA Environment.

The proposed RMSA environment architecture is presented in Figure 3.5. The DRL-agent interacts with the environment by calling the functions defined in the environment interface (env.reset() and env.step()). The dashed and the solid black arrows in the figure show the interactions between the components when the DRL-agent calls the reset and step functions respectively.

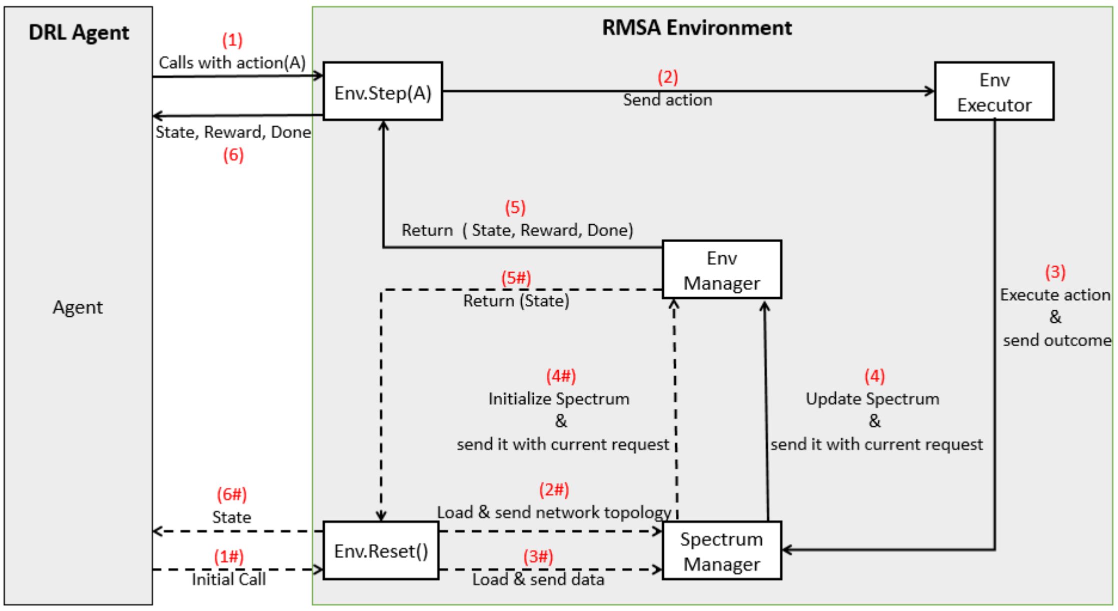


Figure 3.5: Proposed RMSA Environment Architecture.

In Figure 3.5, the dashed arrows are followed during the initialization of the environment. The Process starts when the DRL-agent calls the reset function (1#). The network topology (2#) and the data (3#) are loaded and sent to the Spectrum-Manager. The Spectrum-Manager uses those information to generate and initialize the spectrum utilization that is then sent with information on the current request to the Env-Manager(4#). The Env-Manager will then build the state representation and send it to the DRL-agent(5#) as output of the reset function (6#).

The solid arrows in the other hand are followed when the DRL-agent calls the environment with an action (1) using the step function. The action is sent (2) to the Env-Executor that will check if the action led to a well routing of the corresponding request. It will then send the outcome (3) to the Spectrum-Manager that will update or refresh the spectrum utilization and send it with a new request to the Env-Manager (4). The latter will generate the new state representation, compute the obtained reward and send them to the DRL-agent with an additional information called "Done" which informs the DRL-agent if it is the end of the process or not (5) (6).

The end of the process is the end of an episode. During the training phase, we can repeat as many episodes as we want.

3.3.2 State Space.

The DRL-agent needs to get all relevant information about the current state of the environment to be able to fulfill the task successfully. The following listing provides the raw data that has been identified as relevant.

- The total spectrum usage of all the links: It is a $(|L| \times |S|)$ array containing the information of the spectrum utilization.
- The information about the current request : That is the links belonging to each of the k paths (Obtained through Step 3.2.1) and the corresponding number of frequency slots assigned (Obtained through Step 3.2.3).

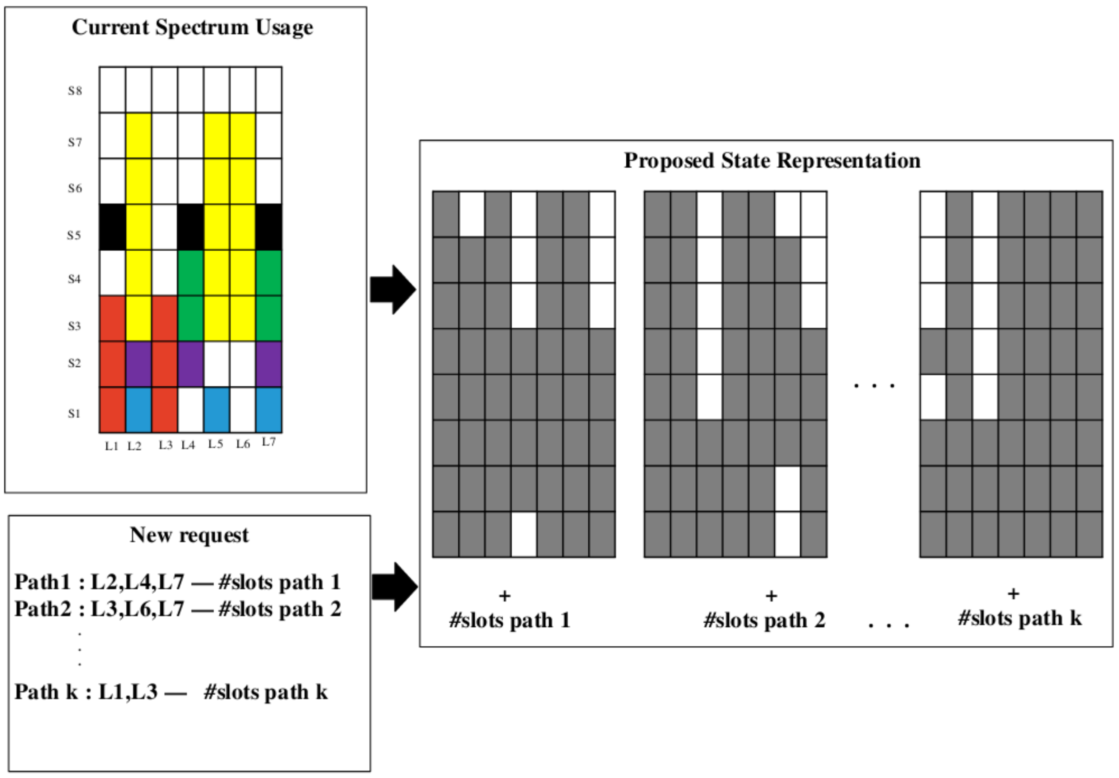

Current Spectrum Usage
S8
S7
S6
S5
S4
S3
S2
S1
L1 L2 L3 L4 L5 L6 L7
New request
Path1 : L2,L4,L7 — #slots path 1
Path2 : L3,L6,L7 — #slots path 2
Path k : L1,L3 — #slots path k
Proposed State Representation
+ #slots path 1
+ #slots path 2
+ #slots path k

Figure 3.6: Proposed State Representation.

Figure 3.6 shows a scheme of our representation. Here, given the current total spectrum usage and the information about the new request (Figure 3.6 left), our approach is to provide directly the agent with information about the spectrum utilization of all the links associated with each of the k candidate paths and the corresponding number of slots (Figure. 3.6 right). Intuitively, this simplifies the problem given that the representation proposed shows the agent a comprehensive picture that describes the link occupation of each path and with the given number of slots, the agent can easily make a comprehensive action.

In the spectrum usage representation in Figure 3.6 left, The white color cells (slots) correspond to available slots and colored cells correspond to occupied slots (different colors are assigned to different requests). In Figure 3.6 right, we just have two colors (white for available and black otherwise). As a path will just use its associated links, the representation considered all the other links as completely unavailable and just the associated links of that path have the real link occupation.

3.3.3 Action Space.

Once the state space is defined, the action space $\mathcal{A}$ of the RMSA problem is straightforward. The action space is defined as:

$$\mathcal{A} = \{A_t \ / \ \ A_t = \{p^k \mid p^1, \ldots, p^{|K|}, p^{|K|+1}\}, \ t \leq H\}, \tag{3.3.1}$$

where A_t is the provisioning action at time t, and H the maximum number of step per episode. p^k denotes that the agent selects the k^{th} shortest path to be routed at time step t given the set of K paths of request d. The action $p^{|K|+1}$ represents the action "do nothing", meaning that the agent is not sure about the path to be selected or it prefers to handle the request later. In the case of action $p^{|K|+1}$, the request is sent at the end of the requests in the queue. Hence, given a request, the possible action at each time step is either to choose the path or to postpone the request.

The spectrum assignment is done using the First Fit algorithm (The first available slots going from the bottom to the top that satisfy all the constraints will be selected).

3.3.4 Reward function.

The design of the reward function could impact provisioning policies, which is critical for policy training. The reward at each time step should help guide the actual provisioning actions, and the accumulative long term reward should also reflect the final provisioning objective. Knowing that when the agent selects a request to be routed on a particular channel at a particular time t we have two possible cases, $i)$ the request is well routed, $ii)$ the request is not routed because of the non-availability of continuous, contiguous and non-overlapping slots at time t or because of capacity overflow. Based on the above understanding, we defined the reward function as:

$$R_{t+1} = \begin{cases} b_{d_t}, & \text{If request } d_t \text{ is well routed,} \\ 0 & \text{Otherwise ,} \end{cases}$$

where R_{t+1} is the immediate reward at time t. b_{d_t} is the bit rate of the current request d at time t.

3.3.5 Neural Network Architecture.

Google recently reported a human-level control paradigm leveraging deep reinforcement learning [20]. Specifically, they parameterized a convolution neural network that can learn successful policies from high-dimensional sensory data (e.g., images). Inspired by this work, and as our state representation is a set of images, for the three reinforcement algorithms that we implemented, we structured a deep neural network consisting of multiple convolutions and fully connected layers to learn the best RMSA policies regarding the proposed EON states. The training of the CNN takes advantage of two key ideas from [20], i.e., deployment of target action-value Q-network and experience replay, for avoiding the divergence of parameters.

Tables 3.3, 3.4 and 3.5 respectively present the network architecture used for the DQN, REINFORCE and Actor-Critic models. There are three convolutional layers and two fully connected layers. The first hidden layer is a 2D-Convolution with 32 filters with a filter size of $[8 \times 8]$ and a stride of 4. The second and third layers are as well 2D-Convolution with 64 filters, while the second layer has a filter size of $[4 \times 4]$ and a stride of 2 and the third layer has a filter size of $[3 \times 3]$ and a stride of 1. The fourth and last hidden layer is a fully-connected layer with 512 neurons. All hidden layers apply the ReLu activation function. The output size depends on the used action space size mentioned in Section 3.3.3 and the input on the state representation (Section 3.3.2). It is a stack of *#paths* 84×84 images that are processed by the first layer. The vector of size *#paths* containing the number of slots associated with those paths is additionally provided by concatenating it with the flattened output of layer three.

Table 3.3: Convolutional Neural Network Architecture (DQN).

Layer	Type	Activation	Number of Filter	Filter Size	Stride
1	Convolution	ReLU	32	$[8 \times 8]$	4
2	Convolution	ReLU	64	$[4 \times 4]$	2
3	Convolution	ReLU	64	$[3 \times 3]$	1
4	Fully-Connected	ReLU	512	-	-
5	Fully-Connected	None	#paths +1	-	-

Table 3.4: Convolutional Neural Network Architecture (REINFORCE).

Layer	Type	Activation	Number of Filter	Filter Size	Stride
1	Convolution	ReLU	32	$[8 \times 8]$	4
2	Convolution	ReLU	64	$[4 \times 4]$	2
3	Convolution	ReLU	64	$[3 \times 3]$	1
4	Fully-Connected	ReLU	512	-	-
5	Fully-Connected	Softmax	#paths+1	-	-

Table 3.5: Convolutional Neural Network Architecture (Actor-Critic).

Layer	Type	Activation	Number of Filter	Filter Size	Stride
1	Convolution	ReLU	32	$[8 \times 8]$	4
2	Convolution	ReLU	64	$[4 \times 4]$	2
3	Convolution	ReLU	64	$[3 \times 3]$	1
4	Fully-Connected	ReLU	512	-	-
5 (Actor)	Fully-Connected	Softmax	#paths+1	-	-
5 (Critic)	Fully-Connected	None	1	-	-

3.4 Baseline Algorithm: a Heuristic Approach

Research conducted by Talebi *et al.* [30] and Alaskar *et al.* [1] mapped the Spectrum Assignment (SA) problem to the task scheduling problem in multi-processor systems where multiple processors are simultaneously assigned to execute the same task. Indeed, given a multiprocessor system that includes a set of m identical processors, a set of n tasks, and processing time p_i for task τ_i, schedule the tasks with the objective to minimize the total processing time consumed by all the tasks, with three different constraints: (1) Preemption is not allowed, (2) Each task should be simultaneously assigned to the corresponding processors, (3) Each processor can be assigned to only one task at a time. As a result of the mapping between the spectrum allocation problem in EONs and scheduling problem in multiprocessor systems, a network is identical to a multi-processing system, requests are identical to tasks, links are identical to processors, and the amount of spectrum is identical to the processing time. As the aim in the multi-processing system is to minimize the processing time consumed by all the tasks, it is for the SA to minimize the total amount of spectrum needed to serve the traffic demand. The problem in that form was solved using a well-known list scheduling heuristic called compact scheduling algorithm.

As the baseline algorithm, we used the compact scheduling algorithm [30] and we adapted the algorithm to allow it to consider the choice through multiple paths. The original compact scheduling algorithm is constituted by the following steps:

1. Select the first request in the list and assign it to a set of consecutive links,
2. Delete the executed request from the list, and update the status (idle or busy) of the corresponding links,
3. Scan the list at the same provisioning instant to select requests that can be executed simultaneously with the currently executed requests,
4. Continue scanning the list until there are no other requests that can be executed at that provisioning instant or no available links,
5. Advance the provisioning time based on the earliest finishing request, and add the available links to the set of free links,
6. Repeat the aforementioned steps until all the requests have been satisfied.

To be able to manage the choice of a path between the available paths, we adapted the compact scheduling algorithm as follows: When a request j having a set of path $path_j$ have to be served, the different paths sorted in ascending order of geographical distance are checked and the first path with the available links is selected. The following algorithm describes the process.

Algorithm 2 Adapted Compact Scheduling Algorithm for multiple paths.

Input: A list D of $|D|$ requests on $|L|$ links each request j having a set of path $path_j$ (ranked in ascending order of geographical distance), each path p having a bandwidth n_{pj} and a set $line_{p_j} \subseteq \{1, 2, \ldots, |L|\}$ of required links.
Output: A schedule of requests, i.e the starting slots S_j which serves the corresponding request when each request j starts provisioning on the network and the path $Pselect_j$ selected for the request j.
Begin

```
1:  $slot \leftarrow 0$                                              ▷ Current slot
2:  $F \leftarrow \{1, 2, 3, \ldots, |L|\}$                          ▷ Set of currently idle links
3:  while $list\ D \neq \emptyset$ do
4:      $j \leftarrow$ first request in the list $D$
5:      $p_j \leftarrow$ first path in the set of $path_j$
6:      while $line_{p_j} \nsubseteq F$ and not at the end of set $path_j$ do
7:          $p_j \leftarrow$ next path in $path_j$                   ▷ Take the next path
8:      end while
9:      if $Line_{p_j} \subseteq F$ then
10:         Remove the request $j$ from the list $D$
11:         $S_j \leftarrow slot$                    ▷ Request $j$ starts at frequency slice $slot$
12:         $Pselect_j \leftarrow p_j$
13:         $F \leftarrow F \setminus line_{p_j}$                ▷ Links of $Line_{p_j}$ become busy
14:     end if
15:     while not at the end of the list D or $F \neq \emptyset$ do
16:         $k \leftarrow$ first request in the list $D$
17:         $p_k \leftarrow$ first path in the set of $path_k$
18:         while $line_{p_k} \nsubseteq F$ and not at the end of set $path_k$ do
19:             $p_k \leftarrow$ next path in $path_k$               ▷ Take the next path
20:         end while
21:         if $Line_{p_k} \subseteq F$ then
22:             Remove the request $k$ from the list $D$
23:             $S_k \leftarrow slot$                ▷ Request $k$ starts at frequency slice $slot$
24:             $Pselect_k \leftarrow p_k$
25:             $F \leftarrow F \setminus line_{p_k}$            ▷ Links in $Linep_k$ become busy
26:         end if
27:     end while                    ▷ No more request may start at frequency slice $slot$
28:     $j \leftarrow$ the first request provisioning at slot $slot$ to complete
29:     $t \leftarrow S_j + n_{pj}$                      ▷ Advance the starting slice $slot$
30:     $F \leftarrow F \cup line_{p_j}$                       ▷ Links in $line_{p_j}$ become idle
31: end while
32: return the starting frequency slices $S_j$ and the selected path $Pselect_j$
```

End.

Following the mathematical formulation of the RMSA problem, as well as the description of the Reinforcement Learning framework and the baseline Algorithm, the next chapter, will present the stage for the experiment and the results obtained will be stated.

4. Experiments and Results

In this chapter, we present and analyze the numerical results obtained with the proposed algorithms for the static RMSA problem, using data sets whose sizes match those of today's real optical networks.

The chapter is organized as follows: In Section 4.1, we describe the details of the network topologies and the demand traffic. Section 4.2 examines and discusses the results of the preprocessing step. We assess the performances of the proposed algorithms through their quantitative and qualitative evaluations that are respectively given in Section 4.3 and 4.4.

4.1 Experimental Setup

This section describes the network topologies and the way the traffic demand has been generated.

4.1.1 Network Topologies.

Four different topologies have been used, i.e., CONUS (a large USA topology reproduced in Figure 4.1), USA (another USA topology, but of medium size, replicated in Figure 4.2), NTT (a topology of Japan shown in Figure 4.1.1) and GERMANY (a topology of Germany shown in Figure 4.4). References and more details for those topologies can be found in SNDlib [23], Monarch Network Architects [23] and The Internet Topology Zoo [16]. The key characteristics of each topology (number of nodes and links; diameter (Δ); average, maximum and minimum node degrees; average, maximum and minimum link length (in Km)) are described in Table 4.1. Columns entitled 'Avg.deg.', 'Max.deg.' and 'Min.deg.' are respectively the average nodal degree, the maximum nodal degree, and the minimum nodal degree. These information are useful to measure the network connectivities, a key characteristic number of shortest (or near shortest) paths. The last 3 columns entitled 'Avg.Link.', 'Max.Link.' and 'Min.Link.' that respectively represent the average link length, the maximum link length and the minimum link length are evaluated, are particularly of interest in order to select the required modulation for the requests routed over short or medium or long paths.

Table 4.1: Key topology characteristics.

Datasets	$\|N\|$	$\|L\|$	Δ	Avg.deg.	Max.deg.	Min.deg.	Avg.Link.	Max.Link.	Min.Link.
CONUS	60	158	15	5.26	8	4	447.93	1468	24.2
USA	24	86	6	7.16	10	4	995.34	2600	250
NTT	55	144	15	5.23	10	2	16.43	50	10
GERMANY	50	176	9	7.04	10	4	41.68	146	1

The four network topologies are represented in Figures 4.1-4.4, where undirected connections represent bidirectional links. Both CONUS and USA topologies have links with very long lengths while NTT and Germany ones have shorter links. NTT network is particular as it contains nodes of degree 2, i.e., nodes $(0), (1), (3), (4)$ and (13). It entails that, for some node pair, there is either a unique one-link path, or a one-link shortest path and then a quite long path for the second shortest path.

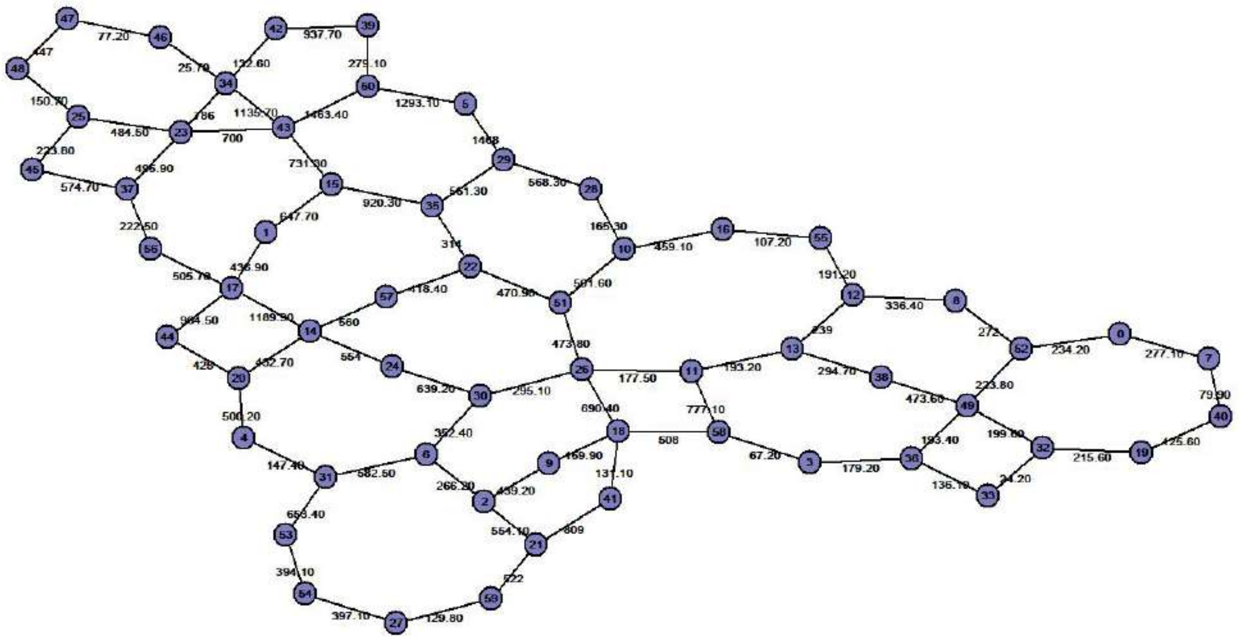


Figure 4.1: CONUS Network Topology.

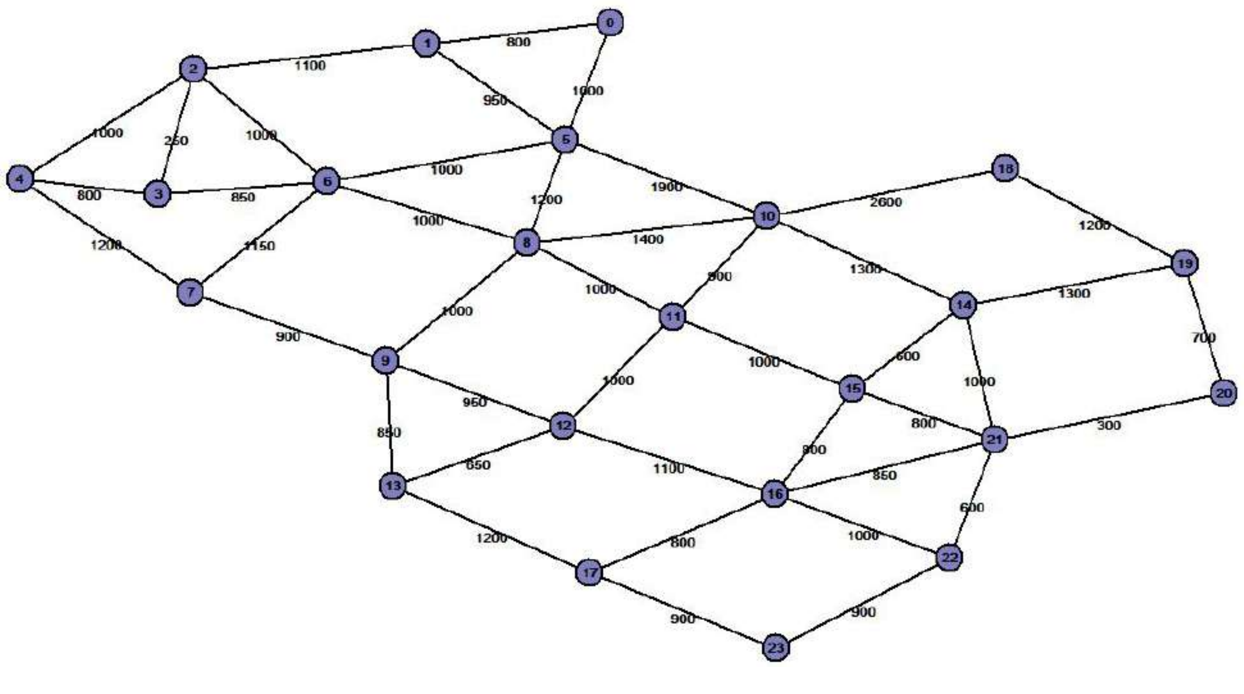


Figure 4.2: USA Network Topology.

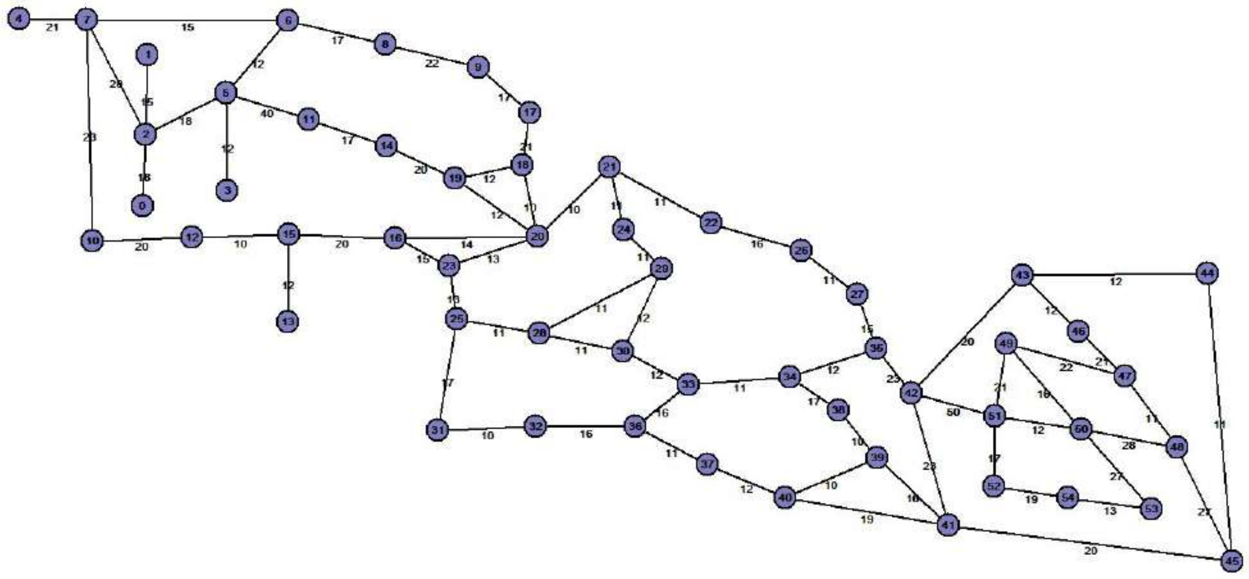


Figure 4.3: NTT Network Topology.

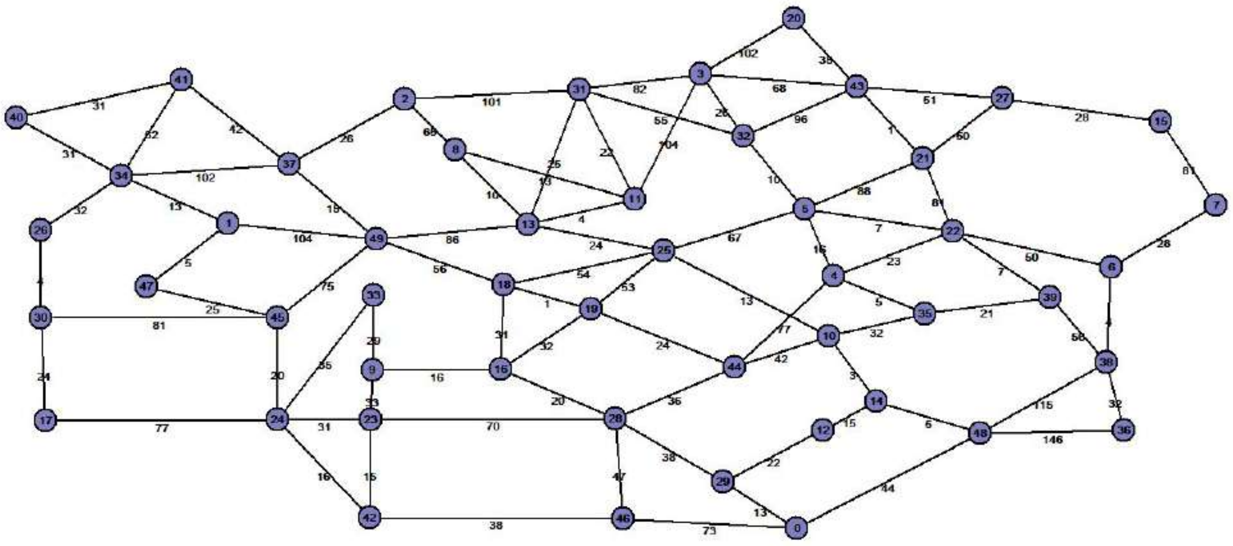


Figure 4.4: GERMANY Network Topology.

4.1.2 Traffic Generation.

For each network topology and for the training, validation and test experiments, a number of request sets has been generated. Source and destination node pairs have been randomly selected. Three types of request bit-rates are considered: 100 Gbps, 200 Gbps, and 400 Gbps. In the first experiment setup all the invoked bit-rates are uniformly distributed (we called it uniform rate traffic in this work, see Table 4.2) and in the second, the bit-rates are distributed as follows: 70% of 100 Gbps, 20% of 200 Gbps and 10% of 400 Gbps (we called it non uniform rate traffic, see Table 4.3).

Table 4.2: Bit-rate distribution: Uniform rate traffic.

Uniform rate traffic					
CONUS Dataset			USA Dataset		
Rate	# requests	Percentage	Rate	# requests	Percentage
100	183	37.57	100	287	38.9
200	160	32.85	200	237	32.1
400	144	29.56	400	213	28.9
Total	487	100	Total	737	100
Offered load (Gbps)	107,900		Offered load (Gbps)	161,300	
NTT Dataset			GERMANY Dataset		
Rate	# requests	Percentage	Rate	# requests	Percentage
100	360	33.24	100	879	36.9
200	358	33.05	200	774	32.5
400	365	33.70	400	729	30.6
Total	1,083	100	Total	2,382	100
Offered load (Gbps)	253,600		Offered load (Gbps)	534,300	

Table 4.3: Bit-rate distribution: Non-uniform rate traffic.

Non-uniform rate traffic					
CONUS Dataset			USA Dataset		
Rate	# requests	Percentage	Rate	# requests	Percentage
100	504	70	100	753	69.98
200	144	20	200	215	19.98
400	72	10	400	108	10.04
Total	720	100	Total	1,076	100
Offered load (Gbps)	108,000		Offered load (Gbps)	161,500	
NTT Dataset			GERMANY Dataset		
Rate	# requests	Percentage	Rate	# requests	Percentage
100	1,183	70	100	2,493	70.00
200	338	20	200	712	19.99
400	169	10	400	356	10.01
Total	1,690	100	Total	3,561	100
Offered load (Gbps)	253,500		Offered load (Gbps)	534,100	

To build models that can generalize, and go beyond the training on a unique traffic demand matrix, we generated, for each network topology, 20 traffic matrices for the training, and 8 for the validation. To test a model, a set of demand for each of the topologies is generated with some selected traffic parameters and the corresponding model is then tested.

With the rates 100 Gbps, 200 Gbps, and 400 Gbps, the number of required frequency slots is computed for different modulation formats using Equation (3.2.2). Corresponding numbers of required frequency slots are given in Figure 4.5.

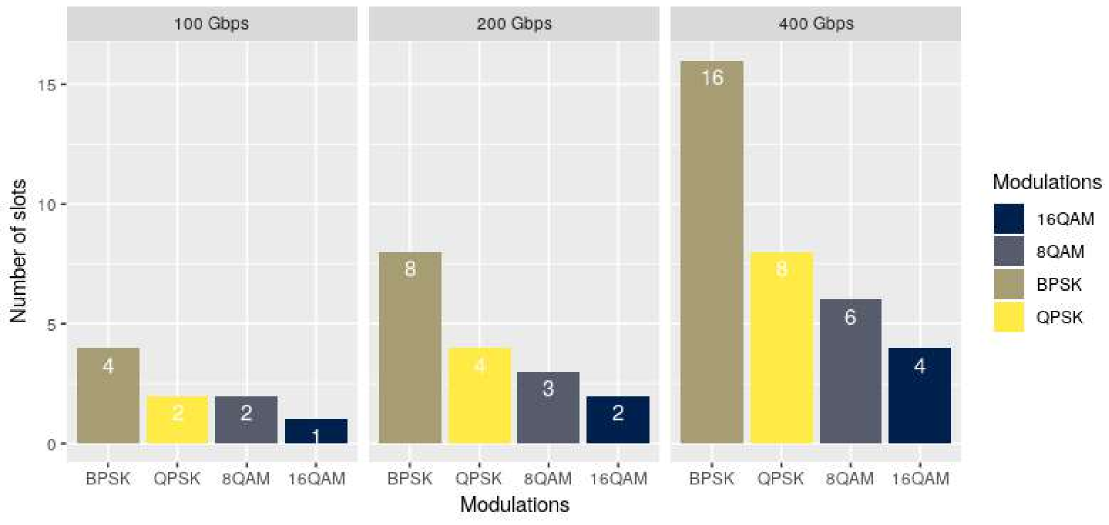


Figure 4.5: Number of slots per modulation and data rate.

From Figure 4.5, we can see that for a given bitrate, the number of slots needed decreases when we move from lower-level modulation (e.g., BPSK) to higher-level modulation (e.g., 16QAM). For a given request, the longer its routing or the higher its rate, the more frequency slots it will require as we will need to use a more slot consuming modulation. Indeed, the RL-agent has to identify the combination of the best route and modulation in order to minimize the number of required frequency slots, while taking care of the congestion on its fiber link. This means the RL-agent may need to compromise between granting a request with a higher rate (so uses more slots) on a short route or granting a request with a lower rate (so less slots occupation) on a longer route in order to maximize the throughput.

4.2 Data Preparation

For each of the training, validation and test traffic matrices that contain requests characterised by source, destination, and rate, a preparation step is processed to find information that will be given to the RL-agent to perform its action. This preparation step is the implementation of the three first steps of the process represented in Figure 3.3 and described in Section 3.2.1, Section 3.2.2 and Section 3.2.3. This will help to generate the state representation of the model. Table 4.4 shows an extracted result of the data preparation on the CONUS traffic demand matrix, where we can see precomputed possible paths, modulations and number of slots for three demands.

For a good Quality of Transmission (QoT), the number of paths per request is determined such as the geographical length of the path does not increase by more than 10% from the shortest path. For the purpose of training time, we set a stopping condition on the maximum number of paths. This condition is that we cannot have more than 5 paths per request. Five because after several experiments, we found out 5 paths to be quite effective, and using a higher number of paths did not result in policy improvement despite the increase in the training time. As the number of admissible paths is different per request, we implemented a masking scheme that will automatically mask all the non-admissible paths at each time step.

Table 4.4: Three first requests on CONUS training demand Preparation.

$Id.$	s	t	b		Paths	Length	Modulation	Slots
1	6	39	400	1	6-30-26-51-22-35-15-43-50-39	5,300.3	BPSK	16
				2	6-30-26-51-10-28-29-5-50-39	5,396.7	BPSK	16
				3	6-30-26-51-22-35-29-5-50-39	5,507.7	BPSK	16
				4	6-31-4-20-44-17-56-37-23-34-42-39	5,700.0	BPSK	16
				5	6-30-26-51-22-35-15-43-34-42-39	5,763.8	BPSK	16
2	45	37	200	1	45-37	574.7	16QAM	2
				2	-	-	-	-
				3	-	-	-	-
				4	-	-	-	-
				5	-	-	-	-
3	41	5	100	1	41-18-26-51-10-28-29-5	3,998.5	QPSK	2
				2	41-18-26-51-22-35-29-5	4,109.5	BPSK	4
				3	41-18-26-11-13-12-15-55-16-10-28-29-5	4,390.3	BPSK	4
				4	-	-	-	-
				5	-	-	-	-

4.3 Quantitative Evaluation

4.3.1 Training Process.

For each network topology, three methods (DQN, REINFORCE, and A2C) are implemented and the average reward is taken. After a Grid-Search on several parameters, the discount factor γ was set to 0.99, the learning rate α to 10^{-6} and the Adam optimizer was used. Instead of the ε-greedy method normally used for action selection (selecting an action corresponding to the maximum Q-value with probability 1-ϵ), we found by experimentation that softmax policy works better for our problem so, we parameterized our policy as a softmax function (that uses Boltzmann distribution) for multiple actions where the probability of selecting an action a at t is computed as:

$$p(a = a^{(i)} \mid s) = \pi_\theta(a = a^{(i)} \mid s) = \frac{\exp(a^{(i)}).f(a^{(i)} \mid s)}{\sum_{j=1}^{N_a} \exp(a^{(j)}).f(a^{(j)} \mid s)}, \tag{4.3.1}$$

where N_a is the number of actions and f a function representing the masking scheme such as:

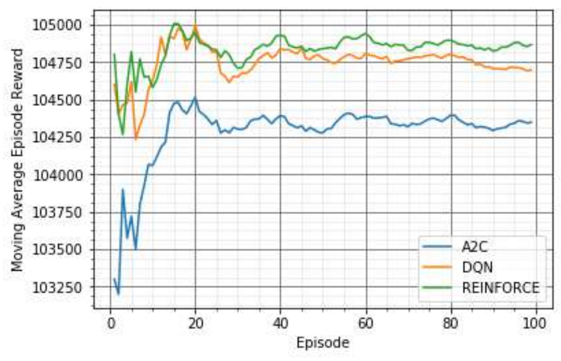
$$f(a^{(j)} \mid s) = \begin{cases} 1, & \text{if } a^{(j)} \text{ is allowed in state s,} \\ 0 & \text{otherwise.} \end{cases}$$

Each agent has been trained for 100 episodes on a CPU of 3.60 GHz with 32 GB of RAM. Table 4.5 gives an overview of the average training time for each agent setup. We can see that the A2C takes on average twice the time on the CONUS as compared to the DQN to train, this can be due to the A2C workflow process. The actor will take time to make its action and additional time will be required to let the critic also makes its action. These two repetitive pairs of actions can explain this observation.

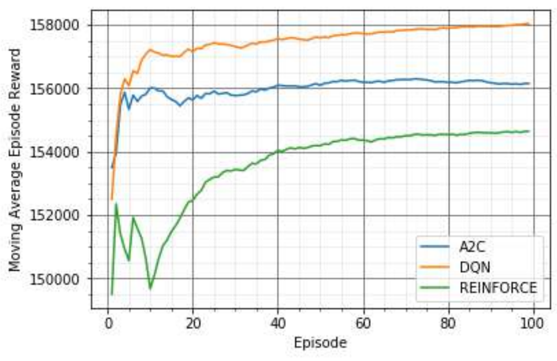
Table 4.5: Training time.

Dataset	DQN	REINFORCE	A2C
CONUS	3h13m39s	5h53m18s	7h12m40s
USA	2h51m07s	6h18m49s	7h22m47s
NTT	4h52m15s	5h15m02s	5h57m05s
GERMANY	3h12m00s	3h05m59s	5h10m29s

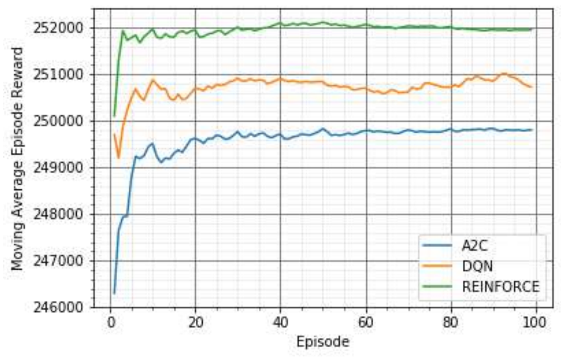
During the training phase, the observation of the evolution of the moving average reward against the episode helps to evaluate if the model really tries to maximize the total cumulative rewards. Figure 4.6 shows the moving average reward of the three algorithms on the four different topologies. It shows that both of the algorithms on average increase the cumulative reward and converge after a certain number of episodes. REINFORCE algorithm got a slightly higher reward on CONUS and NTT topologies while A2C and DQN got higher rewards respectively on GERMANY topology and USA topology. All the algorithms have an increasing shape and converge after at the end. The training phase can then be validated.



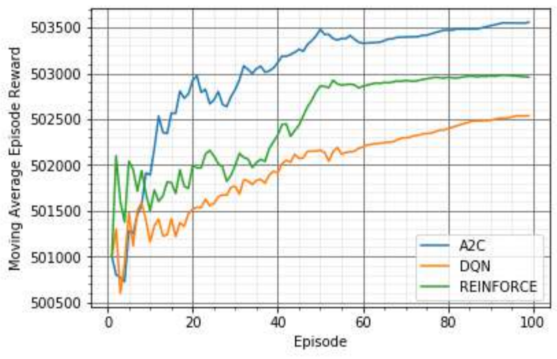
(a) Training results On CONUS Topology.



(b) Training results On USA Topology.



(c) Training results On NTT Topology.



(d) Training results On GERMANY Topology.

Figure 4.6: Moving Average Episode Reward of the Three different methods on each of the topology.

4.3.2 Test Results.

To check the generalisation capability of the model, we run the different models on two types of traffic. In the first one, we ran the models with demands generated following the same traffic as the training phase (non uniform rate traffic). The results as shown in Table 4.6 shows that the different models perform quite well with more than 92% of the demand's throughput well provisioned in each case. The REINFORCE model performs better on each of the topologies with the granted throughput of 99.62% on CONUS, 99.81% on USA, 96.84% on NTT and 96.77% on Germany and therefore is the model that gives the best results.

Table 4.6: Average Test result on non uniform rate traffic.

Dataset	Heuristic	DQN	REINFORCE	A2C
CONUS	108,000 (100%)	106,800 (98.88%)	107,600 (99.62%)	106,000 (98.14%)
USA	161,500 (100%)	161,200 (99.81%)	161,200 (99.81%)	157,800 (97.70%)
NTT	253,500 (100%)	235,500 (92.89%)	245,500 (96.84%)	233,800 (92.28%)
GERMANY	534,000 (99.98%)	504,600 (94.49%)	516,900 (96.77%)	513,700 (96.18%)

In the second experimental process, knowing that it can happen at certain hours of the day that the traffic changes and becomes uniform, we also evaluated the performance of the models on traffic different from the one used to train them. Table 4.7 shows the results on different network topologies. We can surprisingly see that the three algorithms perform even better than on the first experimental setup with approximately 100% on each network topologies. This means that the different models learned precious and useful information on the network topology such as the policy that they follow can be applied and work pretty good on a different traffic type. This is very important since the traffic type is continuously evolving and changing. Such a model that can perform very well on different traffic matrices and different traffic types is really needed. With 100% almost everywhere, the REINFORCE model in the two experimental setups generalises better than the two others method. The direct choice of the path (REINFORCE) instead of the choice of the value of a path (DQN) works better for this problem.

Table 4.7: Average Test result on uniform rate traffic.

Dataset	Heuristic	DQN	REINFORCE	A2C
CONUS	107,900 (100%)	107,900 (100%)	107,900 (100%)	107,900 (100%)
USA	161,300 (100%)	161,300 (100%)	161,300 (100%)	161,300 (100%)
NTT	253,600 (100%)	251,900 (99.32%)	253,600 (100%)	247,000 (97.01%)
GERMANY	534,300 (100%)	531,200 (99.41%)	531,400 (99.45%)	533,800 (99.90%)

4.4 Qualitative Evaluation

Given that the models give quite an acceptable percentage of handled throughput, the visualisation of the provisioning strategy with the spectrum usage gives and insight on the quality of the solution and the satisfaction of the constraints. Indeed, it helps to visually evaluate links occupations and therefore evaluate if the obtained solution can be easily improved using additional methods such as defragmentation [31] for the online version of the problem. Figure 4.7 presents the obtained spectrum usage on the CONUS network topology. The X-axis represents the links and the Y-axis the frequency slots. The lecture of the graph starts from the top to the bottom.

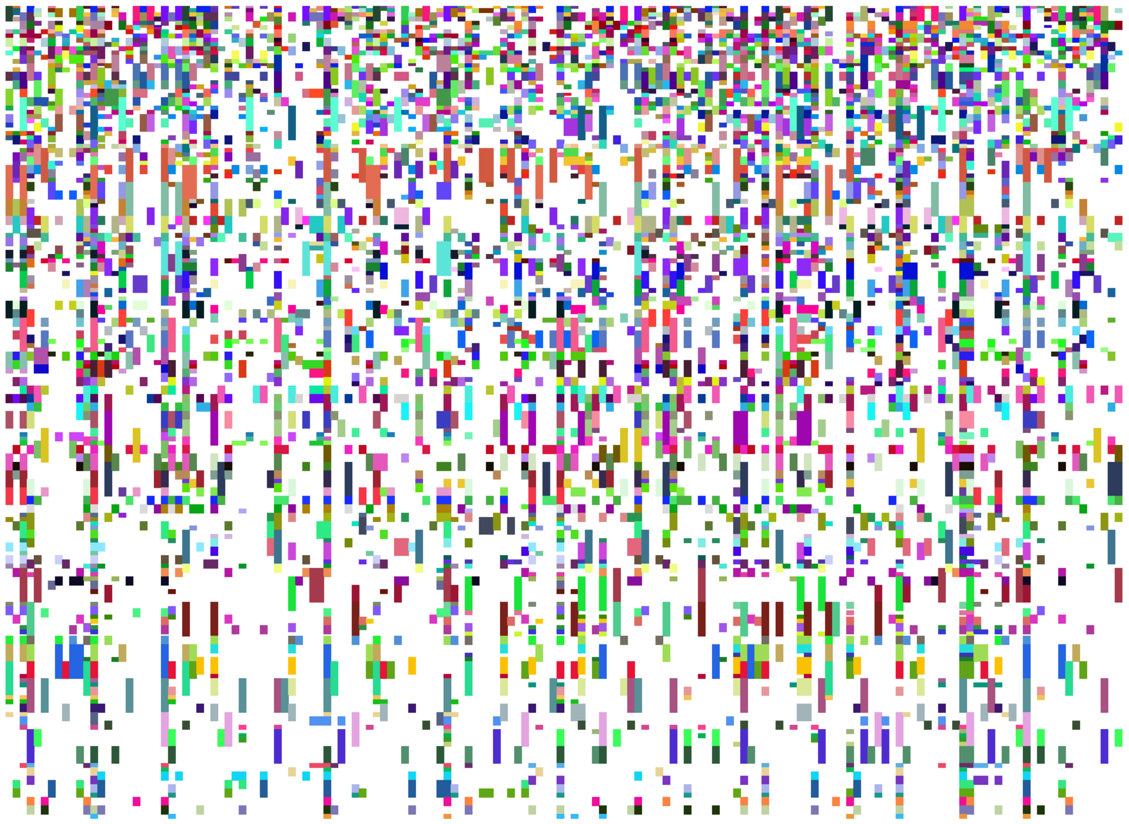


Figure 4.7: Spectrum Usage: X-axis represents the links and the Y-axis the frequency slots, it starts from the up to the bottom.

In Figure 4.7, each demand is represented by one color. On each link (vertical lines), the frequency slots used by a demand is colored with the color of the corresponding demand. White spaces represent empty slots (unused slots). We can first notice that all the constraints of continuity, contiguity, and non-overlapping are satisfied. Secondly, we can notice that the agent mostly starts by granting demand with a very small number of slots needed (as seen at the top of the image where most of the demand use smaller frequency slots per link) and then, proceed with larger ones. This means its policy aims, firstly to select path with a high modulation level when possible to use the least spectrum as possible that led to more and more accepted bitrate. We can also notice from this figure that even though the agent tries to adapt his policy according to the constraints related to the number of slots, there is always a very large number of gaps, i.e. unused slots. This means the agent's policy in some way fails in provisioning the demands since there are enough unused slots but all the demands were not granted.

Conclusion and Future Works

In this work, we propose a reinforcement learning framework to solve a crucial problem facing telecommunications operators today, that of routing and spectrum allocation for transmissions in elastic optical networks. The idea was to design an intelligent agent with a global vision on the network's activities and great adaptability. Any new request characterized by its bit-rate, source, and destination in the network must meet the traffic routing constraints. The intelligent agent, therefore, has to associate each request with an optical path to be taken, modulation and spectral resources in order to maximize the total throughput of the well-routed requests.

We parametrize the RMSA policies with a CNN and train the CNN on several traffic demand sets with experiences from static light-path provisioning. By taking into account the unique characteristics of the RMSA problem, we developed three RL-algorithms: DQN, REINFORCE and A2C. Experimental results show that the proposed algorithms provide good solutions, with great generalization capability on different traffic matrices and types. With 100% of well routed throughput's demand almost everywhere, The REINFORCE algorithm outperforms the other RL algorithms and appears to be the one with best performance and generalization capability. It achieves granted average throughput of 96% in the worst situation and 100% on some topologies for different traffic matrices and types.

We also proposed a very efficient Heuristic algorithm for solving large scale RMSA problems, which gives very good solutions on data sets with up to 60 nodes with on average approximately 100% of well routed demand.

Results finally show that the REINFORCE algorithm is not so far from that sophisticated heuristic algorithm.

Furthering this investigation will essentially rely on four main axes:

- Building an agent with high adaptability that can really in an optimized way manage the routing and spectrum assignment problem, by intensifying the training process and hyper-parameters tuning,
- Improving the scalability of the proposed solution scheme and a better feature representation on the state by adding compactness measure,
- Accounting the use of spectrum in the definition of the reward function to enhance the usage of the spectrum,
- Accounting for the dynamic case of the problem and the addition of the impairments in the model to have a more realistic solution.

Appendix A. Some additional data

A.0.1 Dijkstra Algorithm(adapted from [37]).

Algorithm 3 Dijkstra Algorithm.

Input:
- A directed graph $G(N, L)$ where each edge has a real-valued positive weight c.
- A source node s and target t in the directed graph.

Output: Shortest path form s to t

Begin

```
1:  for All $n \in N$ do
2:      $dist[n]$=INFINITY
3:      $prev[n]$=UNDEFINED
4:  end for
5:
6:  $dist[s] = 0$
7:  $Q = N$
8:  while $Q \neq \emptyset$ do
9:      $u$= vertex in $Q$ with min $dist[u]$
10:     remove $u$ from $Q$
11:     if $u = t$ then
12:         Break
13:     end if
14:     for each neighbor $v \in Q$ of $u$ do
15:         $alt = dist[u] + c(u, v)$
16:         if $dist[n] > alt$ then
17:             $d[v] = alt$
18:             $prev[v] = u$
19:         end if
20:     end for
21: end while
22:
23: Let $S = \emptyset$
24: if prev[u] is DEFINED or $u = s$ then
25:     while u is DEFINED do
26:         Insert u at the beginning of $S$
27:         $u = prev[u]$
28:     end while
29: end if
30: return the Shortest paths S
```

End.

A.0.2 Deep Q-Learning with experience replay (pseudo code from [19]).

Algorithm 4 Deep Q-Learning with experience replay.

```
1:  Initialize replay memory $D$ to capacity $N$
2:  Initialize weights $\theta$ of the action-value Q function
3:  for episode=1,$M$ do
4:      Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
5:      while $t \leq T$ and Not End episode do
6:          With probability $\epsilon$ select a random action $a_t$
7:          Otherwise select $a_t = argmax_a Q(\phi(s_t), a, ; \theta)$
8:          Execute action $a_t$ in the emulator and observe reward $r_t$ and image $x_{t+1}$
9:          Set $s_{t+1} = s_t,\ a_t,\ x_{t+1}$ and proprocess $\phi_{t+1} = \phi(s_{t+1})$
10:         Store $(\phi,\ a_t,\ r_t,\ \phi_{t+1})$ in $D$
11:         Sample random minibatch of transitions $(\phi,\ a_t,\ r_t,\ \phi_{t+1})$ from $D$
12:         Set $y_j = \begin{cases} r_j, & \text{If episode terminates at step } j+1, \\ r_j + \gamma.max_{a'} Q(\phi_{j+1},\ a^{'},\ \theta) & \text{Otherwise .} \end{cases}$
13:         Perform a gradient descent step on $(y_j - Q(\phi_j,\ a_j,\ \theta))^2$ with respect to the network parameters $\theta$
14:
15:     end while
16: end for
```

A.0.3 REINFORCE (pseudo code from [29]).

Algorithm 5 REINFORCE.

Input:
- A differentiable policy parameterization $\pi(a \mid s, \theta)$

Begin

```
1: Initialize weights $\theta$
2: for episode=1,$M$ do
3:     Generate an episode $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, rT$, following $\pi(.|., \theta)$
4:     for each step of the episode t=0,$T-1$ do
5:         $G_t$ = return from $t$
6:         $\theta = \theta + \alpha\gamma^t G_t \nabla_\theta log\pi(a_t \mid s_t, \theta)$
7:     end for
8: end for
```

End.

A.0.4 Actor-Critic Algorithm (pseudo code from [29]).

Algorithm 6 One-Step Actor-Critic.

```
Input:
- A differentiable policy parameterization $\pi(a \mid s, \theta)$
- A differentiable state-value or state-action value function parameterization $\hat{v}(s, \omega)$
Begin
 1: Initialize weights $\theta$ of the actor
 2: Initialize weights $\omega$ of the critic
 3: for episode=1,$M$ do
 4:     Initialize $s_t$ (first state of the episode)
 5:     $I = 1$
 6:     while $t \leq T$ and Not End episode do
 7:         Perform action $a_t$ according to policy $\pi(a_t \mid s_t; \theta)$
 8:         Receive reward $r_t$ and new state $s_{t+1}$
 9:         $\sigma = r_t + \gamma\hat{v}(s_{t+1}, \omega) - \hat{v}(s_t, \omega)$
10:         $\omega = \omega + \alpha_\omega \sigma \nabla \hat{v}(s_t, \omega)$
11:         $\theta = \theta + \alpha_\theta I \sigma \nabla ln\pi(a_t \mid s_t, \theta)$
12:         $I = \gamma I$
13:         $s_t = s_{t+1}$
14:     end while
15: end for
End.
```

Acknowledgements

I am really grateful to God for giving me good health, grace, and illumination to pursue my studies.

I would like to express my sincere gratitude to my supervisor professor, Brigitte Jaumard, for being such a wonderful person, she has always been available to answer my questions, to guide me and share her knowledge with me. She provided me with an excellent atmosphere for conducting this work

Thank you to IVADO for giving me the opportunity to perform my research at the Group for Research in Decision Analysis (GERAD), Montreal, Canada. Thank you to the IVADO staff and especially to Jihane Lamouri, Laurence Renault and Craig Cag. Without their assistance, my research internship could not have been successfully conducted.

A special appreciation to the Centre president of AIMS-Cameroon, Professor. Dr. Mama Foupouagnigni; Academic Director, Professor Marco Andrea Garuti; Cooperative Education Manager, Mr. Honore Youfegnuy, staff, lecturers, and tutors of AIMS-Cameroon for making my stay here worthwhile. I would like to extend my sincere appreciation and thanks to Mastercard Foundation and all AIMS partners for the invaluable support that sustained me throughout this program.

I sincerely appreciate my mentor, Rockefeller, for making himself available always, for helpful conversations, and also for reading and correcting drafts of this essay several times. Thank you as well to Sarah Njomo for correcting this essay. All these were great motivations for my essay.

My days at AIMS-Cameroon would not have been this great without the touch of all AIMS-Cameroon 2018/2019 batch more especially Gaelle Patricia Talotsing, Guy Tsafack, and Avotra Santatriniaina. I appreciate your love, support, and encouragement. Thank you all.

References

[1] R.W. Alaskar, I. Ahmad, and A. Alyatama. Offline routing and spectrum allocation algorithms for elastic optical networks. *Optical Switching and Networking*, 21:79–92, 2016.

[2] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[3] Xiaoliang Chen, Jiannan Guo, Zuqing Zhu, Roberto Proietti, Alberto Castro, and SJB Yoo. Deeprmsa: A deep-reinforcement-learning routing, modulation and spectrum assignment agent for elastic optical networks. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*, pages 1–3. IEEE, 2018.

[4] K Christodoulopoulos, I Tomkos, and E Varvarigos. Spectrally/bitrate flexible optical network planning. In *36th European Conference and Exhibition on Optical Communication*, pages 1–3. IEEE, 2010.

[5] Cisco. *Cisco Visual Networking Index: Forecast and Trends, 2017-2022*. Ciscos White papers, 2017.

[6] Ori Gerstel, Masahiko Jinno, Andrew Lord, and SJ Ben Yoo. Elastic optical networking: A new dawn for the optical layer? *IEEE Communications Magazine*, 50(2):s12–s20, 2012.

[7] Miniwatts Marketing Group. World internet user statistics and 2019 world population statistics. https://www.internetworldstats.com/stats.htm, 2019. Accessed: October 29, 2019.

[8] Adam W Harley, Konstantinos G Derpanis, and Iasonas Kokkinos. Segmentation-aware convolutional networks using local attention masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5038–5047, 2017.

[9] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[10] Jonathan Hui. Rl — policy gradient explained. https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146, 2019. Accessed: August 10, 2019.

[11] G ITU. 694.1: Spectral grids for wdm applications: Dwdm frequency grid. *Std., Feb*, 2012.

[12] Masahiko Jinno, Hidehiko Takara, Bartlomiej Kozicki, Yukio Tsukishima, Yoshiaki Sone, and Shinji Matsuoka. Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. *IEEE communications magazine*, 47(11):66–73, 2009.

[13] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[14] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition. 2016. *URL http://cs231n. github. io*, 50, 2017.

[15] Miroslaw Klinkowski and Krzysztof Walkowiak. Routing and spectrum assignment in spectrum sliced elastic optical path network. *IEEE Communications Letters*, 15(8):884–886, 2011.

[16] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[21] Shanlan Nie, Zhiguo Jiang, Haopeng Zhang, Bowen Cai, and Yuan Yao. Inshore ship detection based on mask r-cnn. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 693–696. IEEE, 2018.

[22] Fiber Optic. Modulation formats for 100g and beyond. https://www.fiberoptics4sale.com, 2019. Accessed: October 29, 2019.

[23] Sebastian Orlowski, Roland Wessäly, Michal Pióro, and Artur Tomaszewski. Sndlib 1.0—survivable network design library. *Networks: An International Journal*, 55(3):276–286, 2010.

[24] Physics and Radio-Electronics. Multiplexing – definition – types of multiplexing: Fdm, wdm, tdm. https://www.physics-and-radio-electronics.com/blog/multiplexing/, 2019. Accessed: October 29, 2019.

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[27] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[28] Terabit Super-Channels. The evolution of next-gen optical networks.

[29] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

[30] S. Talebi, E. Bampis, G. Lucarelli, I. Katib, and G.N. Rouskas. Spectrum assignment in optical networks: A multiprocessor scheduling perspective. *Journal of Optical Communications and Networking*, 6(8):754–763, 2014.

[31] Ioannis Tomkos, Siamak Azodolmolky, Josep Sole-Pareta, Davide Careglio, and Eleni Palkopoulou. A tutorial on the flexible optical networking paradigm: State of the art, trends, and research challenges. *Proceedings of the IEEE*, 102(9):1317–1337, 2014.

[32] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[33] Luis Velasco, Miroslaw Klinkowski, Marc Ruiz, and Jaume Comellas. Modeling the routing and spectrum allocation problem for flexgrid optical networks. *Photonic Network Communications*, 24(3):177–186, 2012.

[34] Luis Velasco and Marc Ruiz. *Provisioning, Recovery, and In-Operation Planning in Elastic Optical Networks*. John Wiley & Sons, 2017.

[35] Yang Wang, Xiaojun Cao, and Yi Pan. A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *2011 Proceedings Ieee Infocom*, pages 1503–1511. IEEE, 2011.

[36] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[37] Wikipedia contributors. Dijkstra's algorithm — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Dijkstra%27s_algorithm&oldid=930487962, 2019. [Online; accessed 14-December-2019].

[38] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[39] Jin Y Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, 27(4):526–530, 1970.

[40] Guoying Zhang, Marc De Leenheer, Annalisa Morea, and Biswanath Mukherjee. A survey on ofdm-based elastic core optical networking. *IEEE Communications Surveys & Tutorials*, 15(1):65–87, 2012.