INSTITUT POLYTECHNIQUE DE PARIS

NNT : 20XXIPPAXXXX

Thèse de doctorat

Apprentissage par renforcement profond pour l'optimisation conjointe de l'allocation de ressource et de l'ordonnancement pour les réseaux ad hoc

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom ParisTech

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Sciences et Technologies de l'information et de la communication (STIC)

Thèse présentée et soutenue à Palaiseau, le 11 décembre 2025, par

SYLVAIN NÉRONDAT

Composition du Jury :

Inbar Fijalkow Professeur, ENSEA (ETIS)	Président
Olivier Berder Professeur, IRISA	Rapporteur
Didier Le Ruyet Professeur, CNAM	Rapporteur
Philippe Mary Professeur, INSA Rennes	Examinateur
Lila Boukhatem Professeur, University Paris-Saclay	Examinateur
Philippe Ciblat Professeur, Télécom Paris (LTCI)	Directeur de thèse
Christophe Le Martret Expert, Thales	Co-directeur de thèse
Xavier Leturc Ingénieur, Thales	Encadrant

Contents

List of acronyms **6**

Résumé en français **9**

General introduction **13**

1 General context and state of the art on scheduling **17**
1.1 Introduction . . . 17
1.2 General system model . . . 17
1.2.1 Additional information and precisions on the general system model in the different chapters . . . 19
1.2.1.1 Additional information and precisions for Chapter 2 . . . 20
1.2.1.2 Additional information and precisions for Chapter 3 . . . 20
1.2.1.3 Additional information and precisions for Chapter 4 . . . 20
1.2.1.4 Synthesis of the additional information and precisions . . . 20
1.2.2 Mapping with the 5G model . . . 21
1.3 State of the art of heuristics for scheduling . . . 23
1.3.1 Round-robin . . . 26
1.3.2 Earliest deadline first . . . 26
1.3.3 Proportional fair . . . 26
1.3.4 MLWDF . . . 27
1.3.5 LOG-rule . . . 27
1.3.6 EXP-rule . . . 28
1.3.7 Knapsack . . . 28
1.4 State of the art of DRL solutions for scheduling . . . 29
1.4.1 Classification of the DRL schedulers . . . 29
1.4.1.1 Classification based on the action space . . . 29
1.4.1.2 Classification based on the state space . . . 31
1.4.1.3 Classification based on the reward . . . 32
1.4.1.4 Classification based on the performance metrics during the inference phase 33
1.4.2 Analysis of the scheduler properties . . . 33
1.5 Conclusion . . . 35

2 Slot-based scheduling **37**
2.1 Introduction . . . 37
2.2 System model . . . 37
2.3 Problem formulation . . . 38

2.3.1 MDP formulation . . . 38
2.3.2 State space . . . 38
2.3.2.1 State-HoL (S-HoL) . . . 39
2.3.2.2 State-xHoL (S-xHoL) . . . 39
2.3.2.3 State-APD (S-APD) . . . 39
2.3.3 Action space . . . 40
2.3.4 MDP model . . . 40
2.3.5 Reward . . . 40
2.4 Problem solution . . . 40
2.4.1 Learning procedure . . . 40
2.4.2 Encoder only transformer architecture . . . 41
2.4.3 Fully connected architecture . . . 42
2.4.4 Adaptation of the heuristics . . . 43
2.4.4.1 Round-robin . . . 43
2.4.4.2 Proportional fair . . . 43
2.4.4.3 MLWDF . . . 43
2.4.4.4 LOG-rule . . . 44
2.4.4.5 EXP-rule . . . 44
2.4.4.6 Knapsack . . . 44
2.5 Performance evaluation . . . 45
2.5.1 Simulation settings . . . 45
2.5.1.1 Evaluated methods . . . 45
2.5.1.2 Training setup . . . 45
2.5.1.3 Inference setup . . . 46
2.5.2 Performance metrics . . . 47
2.6 Performance analysis . . . 48
2.6.1 Training analysis . . . 48
2.6.2 Inference analysis . . . 51
2.6.2.1 Tested inferences . . . 51
2.6.2.2 Generalization wrt Λ . . . 52
2.6.2.3 Performance per link or traffic for $\Lambda = 1.6$ . . . 56
2.6.2.4 Performance wrt n_L . . . 60
2.6.2.5 Generalization wrt n_L and $\mathcal{C}$ . . . 63
2.7 Conclusion . . . 67

3 Frame-based scheduling **68**
3.1 Introduction . . . 68
3.2 Problem formulation . . . 69
3.2.1 State spaces . . . 69
3.2.2 Action space . . . 69
3.2.3 Reward . . . 69
3.3 Problem solution . . . 69
3.3.1 Deep neural network architecture . . . 69
3.3.2 Action masking procedure . . . 71
3.4 Performance evaluation . . . 72
3.4.1 Wireless Suite environment . . . 72

3.4.2 Communication model . . . 73
3.4.3 State spaces and reward of WS-TFRA and proposed adaptations . . . 74
3.4.3.1 State spaces . . . 74
3.4.3.2 Reward . . . 75
3.4.4 Heuristics used in wireless suite . . . 76
3.4.4.1 Proportional fair . . . 76
3.4.4.2 Knapsack . . . 76
3.4.4.3 Bosch agent . . . 77
3.4.5 Communication setup . . . 77
3.4.6 Training and inference setup . . . 78
3.4.7 Performance metrics . . . 78
3.5 Performance analysis . . . 79
3.5.1 Inference performance on the training setup . . . 79
3.5.2 On the importance of the mask for the proposed architecture . . . 89
3.5.3 Generalization vs. β . . . 91
3.6 Conclusion . . . 93

4 Joint scheduling and MCS selection **95**
4.1 Introduction . . . 95
4.2 System model . . . 95
4.2.1 Buffer model . . . 96
4.2.2 Channel model . . . 96
4.2.2.1 Procedure for the average PER . . . 97
4.3 Problem formulation for JSM and DSM approaches . . . 98
4.3.1 State spaces . . . 98
4.3.2 Action spaces for both JSM and DSM approaches . . . 99
4.3.3 Reward . . . 99
4.4 Problem solutions . . . 100
4.4.1 Solution approaches for MCS and buffer selections . . . 100
4.4.2 Adaptation of the heuristics for buffer selection . . . 102
4.4.2.1 Round-robin . . . 102
4.4.2.2 Proportional fair . . . 102
4.4.2.3 MLWDF . . . 102
4.4.2.4 LOG-rule . . . 103
4.4.2.5 Knapsack . . . 103
4.4.3 Adaptation of the EOT-AB for JSM and DSM-LBS approaches . . . 103
4.5 Performance evaluation . . . 105
4.5.1 Simulation settings . . . 105
4.5.1.1 Training setup . . . 108
4.5.1.2 Inference setup . . . 108
4.5.2 Performance Metrics . . . 108
4.6 Performance analysis . . . 110
4.6.1 Training analysis . . . 110
4.6.2 Inference performance analysis . . . 111
4.7 Conclusion . . . 116

Conclusions and perspectives **117**

A Background on machine learning **119**
A.1 Introduction . . . 119
A.2 Reinforcement learning . . . 119
A.2.1 Introduction . . . 119
A.2.2 Finite Markov chain . . . 120
A.2.3 Finite Markov Decision Process . . . 121
A.2.4 Optimal control framework and MDP . . . 123
A.2.5 Optimal policy . . . 126
A.2.6 Value iteration . . . 129
A.2.7 Q-Learning . . . 131
A.3 Deep reinforcement learning . . . 133
A.4 Deep neural network architectures . . . 136
A.4.1 Fully connected architecture . . . 136
A.4.2 Transformer architecture . . . 137
A.4.3 Action branching architecture . . . 144
A.5 Conclusion . . . 146

B Number of possible states for APD **147**

C Figures for Chapter 2 **149**
C.1 Figure for DC traffic . . . 150
C.1.1 Heuristics . . . 150
C.1.2 Fully connected . . . 155
C.1.3 EOT . . . 158
C.2 Figure for BE traffic . . . 161
C.2.1 Heuristics . . . 161
C.2.2 Fully connected . . . 166
C.2.3 EOT . . . 169

Bibliography **172**

List of acronyms

5G	fifth generation
6G	sixth generation
AB	action branching
AI	artificial intelligence
APD	all packet delays
ARQ	automatic repeat request
AWGN	additive white Gaussian noise
BA	Bosch agent
BE	best-effort
BERT	bidirectional encoder representations from transformer
BET	blind equal throughput
BO	buffer overflow
BS	base station
CA	channel-aware
cdf	cumulative distribution function
CQI	channel quality indicator
CSI	channel state information
CU	channel-unaware
CV	computer vision
CW	codeword
DC	delay constraint
DDQL	double deep Q-Learning
DNN	deep neural network
DOT	decoder only transformer
DP	dynamic programming
DQL	deep Q-Learning
DQN	deep Q-network
DRB	data radio bearer

DRL deep reinforcement learning
DSM disjoint scheduling and MCS
DV delay violation
EDF earliest deadline first
EDR expected discounted return
EE energy efficiency
EIRP effective isotropic radiated power
EOT encoder-only transformer
ESM effective SNR mapping
EXP-rule exponential rule
FC fully connected
FFN feed forward network
FIFO first in, first out
FLOPs floating point operations
GAR global arrival rate
GBM global buffer management
GBR guaranteed bit rate
GNN graph neural network
GPF generalized PF
GPT generative pre-trained transformer
GRS global RB scheduling
GRS-I GRS-index
GRS-P GRS-proportion
HARQ hybrid automatic repeat request
HBS heuristic for the buffer selection
HMS heuristic for the MCS selection
HoL head of line
IP internet protocol
JSM joint scheduling and MCS
KP knapsack
LBS learning for buffer selection
LDPC low-density parity check
LLM large language model
LOG-rule logarithmic rule
LSTM long-short term memory

LTE long term evolution
LUT look-up table
LWDF largest weighted delay first
MAC medium access control
MCS modulation and coding schemes
MDP Markovian decision process
MHA multi-head attention
ML machine learning
MLWDF modified largest weighted delay first
MSI matrix state input
MT max throughput
NLI number of links independent
NLP natural language processing
NOMA non-orthogonal multiple access
NP number of packets
OSI open systems interconnection
PD packet delay
PDB packet delay budget
PDCP packet data convergence protocol
pdf probability density function
PE permutation equivariant
PER packet error rate
PF proportional fair
PHY PHYsical
PL packet loss
PLR packet loss rate
pmf probability mass function
PPO proximal policy optimization
QA QoS-aware
QFI QoS flow identifier
QoS quality of service
QU QoS-unaware
RAN radio access network
RB resource block
ReLU rectified linear unit

RL reinforcement learning
RLC radio link control
RNN recurrent neural network
ROHC robust header compression
RR round-robin
RRM radio resource manager
SAC soft actor critic
SINR signal-to-interference-plus-noise ratio
SNR signal-to-noise ratio
SotA state of the art
SDAP service data adaptation protocol
SRS sequential RB scheduling
TD temporal difference
TFRA *TimeFreqResourceAllocation-v0*
TFT traffic flow template
UE user equipment
UPF user plane function
VI value iteration
VSI vector state input
wrt with respect to
WS wireless suite
WT waiting time
xHoL extended HoL

Résumé en français

Les réseaux de communication modernes sont confrontés à des défis croissants en raison de la diversité des applications, allant des services ultra-fiables à faible latence aux communications massives de type machine dans le cadre de l'Internet des objets. Ces applications variées génèrent des types de trafic hétérogènes qui exigent des stratégies de gestion des ressources, telles que l'ordonnancement ou l'allocation de ressources, capables de s'adapter à leurs besoins spécifiques. Cette thèse se concentre principalement sur l'ordonnancement des paquets, l'allocation de ressources n'étant considérée qu'à la fin de la thèse.

L'ordonnancement étudié dans cette thèse concerne à la fois le domaine temporel et fréquentiel. Il définit comment les ressources radio dans ces dimensions sont attribuées aux différents dispositifs, appelés équipements utilisateur (UEs), afin d'atteindre des objectifs de performance tels que le débit, l'équité et la latence. Il joue un rôle central dans les réseaux sans fil, en garantissant que plusieurs appareils puissent accéder au médium de manière coordonnée et efficace. Les ressources radio élémentaires à attribuer se composent de créneaux temporels (time slots) dans le domaine temporel et de blocs de ressources (RBs) dans le domaine fréquentiel. Nous supposons dans cette thèse que l'ordonnancement est géré par une unité centrale appelée gestionnaire de ressources radio (RRM). Dans les réseaux cellulaires, le RRM correspond à la station de base (BS). Dans les réseaux ad hoc, les requêtes d'ordonnancement sont centralisées dans un nœud spécifique, désigné comme RRM. Dans les réseaux cellulaires, la BS sert de nœud central reliant les différents liens, ce qui permet d'accéder à l'information instantanée de l'état du canal (CSI). En revanche, dans les réseaux ad hoc, le RRM n'a accès qu'à des informations statistiques du canal.

À mesure que le nombre de types de services augmente, les techniques d'ordonnancement classiques, qui reposent souvent sur des heuristiques ou sur l'optimisation de métriques, deviennent moins adaptées. Ces méthodes manquent de flexibilité pour équilibrer plusieurs objectifs, souvent contradictoires, et ne sont pas conçues pour gérer conjointement différents types de ressources. En revanche, les méthodes basées sur l'intelligence artificielle (AI) peuvent apprendre à gérer de tels compromis de manière adaptative. Cette évolution est cohérente avec les principes de conception des systèmes de cinquième génération (5G) et de sixième génération (6G) à venir, où l'AI jouera un rôle essentiel dans la conception et l'optimisation des architectures, protocoles et opérations.

Parmi les paradigmes d'AI, l'apprentissage profond a gagné en importance grâce aux avancées des architectures de réseaux de neurones profonds (DNN). Lorsqu'il est combiné à l'apprentissage par renforcement (RL), formant ainsi l'apprentissage par renforcement profond (DRL), il a démontré des capacités remarquables dans les tâches de prise de décision, initialement popularisé par les succès dans les jeux Atari, puis conforté par son rôle dans l'entraînement des grands modèles de langage (LLM). Suivant cette tendance dans de nombreux domaines de recherche, le DRL est naturellement exploré pour l'allocation de ressources pour les communications sans fil.

Cette thèse est consacrée au développement de méthodes basées sur l'AI pour l'ordonnancement dans les réseaux de communication sans fil. L'objectif est d'explorer comment le DRL peut être utilisé efficacement pour optimiser l'allocation de ressources dans des environnements complexes caractérisés par des modèles de trafic variés et des conditions de canal dynamiques. Pour atteindre cet objectif, nous commençons par identifier les propriétés essentielles qu'une architecture de DNN pour l'ordonnancement doit satisfaire. Nous validons d'abord cette architecture dans un environnement simplifié, où l'ordonnancement est réalisé slot par slot et où le domaine fréquentiel se limite à un seul RB. Nous étendons ensuite l'étude à un cas plus général avec plusieurs RBs qui sont attribués simultanément. Ces approches sont évaluées dans deux scénarios : 1) avec CSI instantané, correspondant aux systèmes 5G ; 2) avec CSI statistique, correspondant aux réseaux ad hoc.

L'organisation du document de thèse est la suivante.

Le chapitre 1 présente le modèle de système général adopté et passe en revue l'état de l'art des ordonnanceurs heuristiques et basés sur le DRL. Il introduit une classification des ordonnanceurs DRL en fonction de leurs entrées (caractéristiques des UEs) et de leurs sorties. Le chapitre identifie également trois propriétés essentielles qu'une architecture NN efficace doit satisfaire pour l'ordonnancement et l'allocation de ressources :

- L'architecture doit pouvoir gérer un nombre variable de liens.
- L'architecture doit être équivariante par permutation.
- L'architecture doit prendre en compte les buffers de tous les UEs conjointement.

Ce chapitre classe les architectures existantes selon ces critères. Nous identifions que le transformeur basé uniquement sur l'encodeur (EOT) satisfait ces trois propriétés grâce à son mécanisme d'attention et l'utilisons ensuite comme base de l'architecture proposée.

Le chapitre 2 formule le problème d'ordonnancement slot par slot pour un seul RB comme un processus de décision markovien (MDP), en supposant un canal sans erreur et de capacité fixe où des erreurs peuvent survenir à cause de violation de délai ou de buffer overflow. Il introduit une solution basée sur l'EOT, respectant les trois propriétés clés identifiées au chapitre 1. Nous avons entraîné l'EOT pour un nombre spécifique de liens et pour une valeur spécifique du taux d'arrivée du trafic, puis nous avons comparé ses performances à celles d'heuristiques et d'un ordonnanceur conventionnel basé sur un réseau de neurones entièrement connecté (FC DNN). Nous avons ensuite évalué la capacité de généralisation de l'ordonnanceur EOT par rapport au taux d'arrivée du trafic et au nombre de liens dans des réseaux non rencontrés lors de l'entraînement. Les performances ont été évaluées en termes de taux perte de paquets (PLR), de débit, d'équité et de délai de transmission. Nos résultats ont montré que : 1) l'EOT surpasse les heuristiques ; 2) l'EOT surpasse également les ordonnanceurs basés sur les FC, bien que ces derniers aient été entraînés spécifiquement pour un nombre donné de liens, c'est-à-dire un ordonnanceur FC distinct pour chaque configuration entraînée/testée. De plus, l'EOT maintient des performances robustes même lorsqu'il est testé sur des configurations de liens qu'il n'avait jamais rencontrées lors de l'entraînement, ce qui confirme sa scalabilité, son adaptabilité et sa forte capacité de généralisation.

Le chapitre 3 étend le chapitre 2 en considérant l'ordonnancement sur plusieurs RBs (appelé trame), sous l'hypothèse d'un canal sans erreur à capacité variable dans le temps. Deux approches sont examinées : 1) une sélection conjointe des UEs pour tous les RBs simultanément, ce qui constitue la principale contribution de ce chapitre ; 2) une sélection séquentielle d'un UE par RB, comme au chapitre 2.

L'espace d'actions de 1) étant très grand, nous proposons d'utiliser l'architecture de ramification d'actions (AB) pour réduire la cardinalité de l'espace d'actions grâce à une décomposition. Combinée à l'EOT, cela conduit à une nouvelle architecture NN appelée EOT-AB. Pour éviter l'allocation de RB à des buffers vides, nous avons intégré un masquage adaptatif des actions. La solution 1) est évaluée par rapport aux heuristiques et à la solution 2), mettant en avant l'importance d'une sélection conjointe des UEs. De plus, nous avons démontré l'efficacité du masquage adaptatif des actions pendant l'inférence, soulignant son importance pour améliorer les performances pendant les phases d'inférence.

Le chapitre 4 examine le problème de l'ordonnancement avec sélection des schémas de modulation et de codage (MCS) sur une trame, où des erreurs de transmission peuvent se produire en plus des erreurs liées à la violation de délai et au buffer overflow, dans un canal à évanouissement plat de Rayleigh, en supposant un CSI statistique. La probabilité de transmission correcte des paquets dépend du MCS sélectionné. La solution proposée s'appuie sur l'architecture EOT-AB introduite au chapitre 3. Deux approches sont comparées : 1) une solution conjointe qui réalise simultanément l'ordonnancement et la sélection du MCS ; 2) une solution où le DRL est utilisé pour l'ordonnancement tandis qu'une heuristique détermine le MCS. Les résultats expérimentaux ont démontré que la solution conjointe surpasse systématiquement la solution disjointe en termes de PLR pour différents taux d'arrivée de trafic. Cette comparaison met en évidence les avantages d'une solution réalisant conjointement l'ordonancement et la sélection du MCS.

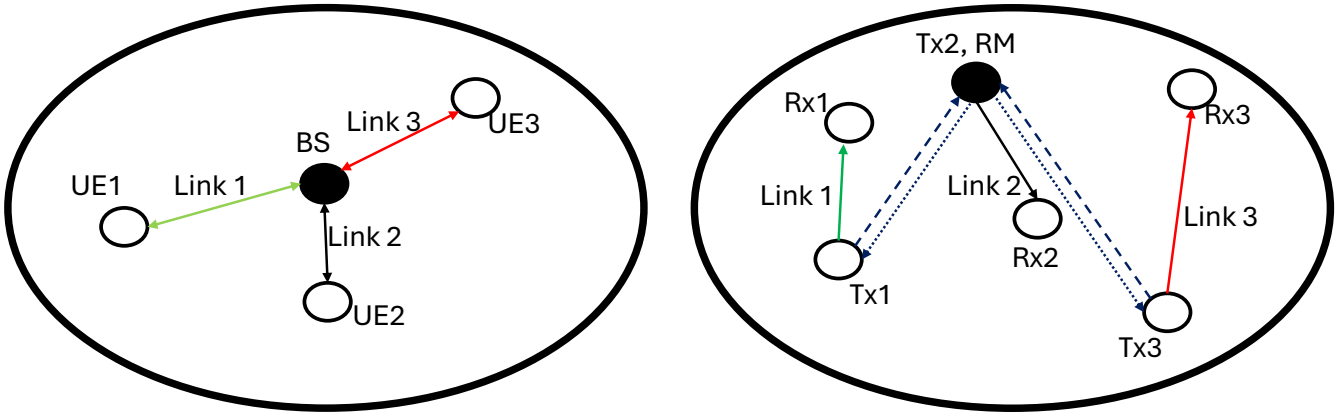
General introduction

Problem statement

Modern communication networks face growing challenges due to the increasing diversity of applications, ranging from ultra-reliable low-latency communications to massive machine type communications in the context of the Internet of things. These diverse applications generate heterogeneous types of traffic, which demand efficient resource management strategies capable of adapting to their specific requirements, such as scheduling or resource allocation. This thesis primarily focuses on packet scheduling, with resource allocation considered only at specific points.

The scheduling considered in this thesis involves both the time and frequency domains. It specifies how radio resources in these dimensions are allocated among devices, called user equipments (UEs), to meet performance objectives such as throughput, fairness, and latency. It plays a central role in wireless networks, ensuring that multiple devices can access the medium in a coordinated and efficient manner. The elementary radio resources to be allocated among UEs consist of time slots in the time domain and resource blocks (RBs) in the frequency domain.

We assume in this thesis that the scheduling is handled by a central unit, called radio resource manager (RRM). In cellular networks, such as represented in Figure 1a, the RRM is the base station (BS). In ad hoc networks, such as illustrated in Figure 1b, scheduling requests are centralized at a specific node, denoted by RRM. In cellular networks, the BS serves as the central node connecting the different links, which may allow to access to instantaneous channel state information (CSI). In contrast, in ad hoc networks, the RRM is assumed to have access only to statistical CSI [1].



(a) Cellular network.

(b) Ad hoc network.

Figure 1: Different types of networks. The plain lines represent the links, the dashed lines represent the CSI feedback and the dotted lines represent the scheduling instruction.

As the number of types of service grows, conventional scheduling techniques, that often rely on heuristics or metric optimization, become less suitable. These methods lack the flexibility required to balance multiple and often conflicting objectives, and they are not well adapted to the joint management of heterogeneous resource types. In contrast, artificial intelligence (AI)-driven methods can learn to adaptively manage such trade-offs. This development is consistent with the design principles of emerging fifth generation (5G) and planned sixth generation (6G) systems, where AI is expected to play a critical role in designing and optimizing 6G architectures, protocols, and operations [2].

Among AI paradigms, deep learning has gained substantial traction due to the advances in deep neural network (DNN) architectures. When combined with reinforcement learning (RL), leading to deep reinforcement learning (DRL), it has demonstrated remarkable capabilities in decision-making tasks, originally popularized through successes in Atari games [3] and more recently strengthened by its role in training large language model (LLM) [4]. Following this trend across many research domains, the DRL is then naturally explored for wireless resource allocation.

This thesis is devoted to the development of AI-based methods for scheduling in wireless communication networks. The objective is to explore how DRL can be effectively employed to optimize scheduling and resource allocation in complex environments characterized by diverse traffic patterns and dynamic channel conditions.

To achieve this objective, we begin by identifying the essential properties that a DNN architecture for scheduling should satisfy. We first validate this architecture in a simplified environment, where scheduling is performed slot by slot and the frequency domain consists of a single RB. We then extend the study to a more general case with multiple RBs, which are allocated simultaneously. These approaches are evaluated under two scenarios:

- With instantaneous CSI, corresponding to 5G systems.
- With statistical CSI, corresponding to ad hoc networks case.

Outline and contributions

This thesis is organized as follows.

Chapter 1 presents the general system model adopted throughout this thesis and reviews the state of the art in heuristic and DRL-based schedulers. It introduces a classification of DRL schedulers according to their inputs (i.e. UE or link characteristics) and their outputs. Furthermore, the chapter identifies three essential properties that an effective DNN architecture should satisfy for scheduling and resource allocation, and classifies existing architectures with respect to these properties. We identify that the encoder-only transformer (EOT) DNN satisfies these three properties and is then used as the core of our proposed architecture for the next chapters.

Chapter 2 formulates the scheduling problem slot by slot for a single RB, as an Markovian decision process (MDP), under the assumption of an error-free channel with fixed capacity, where packet losses may occur due to delay violation (DV) or buffer overflow (BO). It introduces an EOT-based solution satisfying the three key properties identified in Chapter 1. This solution is evaluated against heuristics and alternative DRL methods lacking these three properties, highlighting their critical role in achieving effective scheduling.

Chapter 3 extends Chapter 2 by considering the scheduling problem over multiple RBs (called frame), under the assumption of an error-free channel with time varying capacity. Two approaches are examined: 1) a joint selection of UEs for all RBs simultaneously, which constitutes the main contribution of this chapter, and 2) a sequential selection of one UE per RB, as in Chapter 2. The number of actions for 1) is very large, thus, we propose leveraging the action branching (AB) architecture to reduce the cardinality of the action space through action decomposition. Used alongside the EOT architecture, this leads to a new DNN architecture called EOT-AB. The solution of 1) is evaluated against both heuristic baselines and the solution of 2), highlighting the importance of performing a joint selection of UEs for all RBs simultaneously.

Chapter 4 investigates the scheduling problem and modulation and coding schemes (MCS) selection over a frame, where transmission errors may occur, in addition to packet losses causes by DV and BO, under a Rayleigh flat fading propagation channel, assuming that statistical CSI is available. The proposed solution builds on the EOT-AB architecture introduced in Chapter 3. Two approaches are compared: 1) a joint solution that simultaneously performs scheduling and MCS selection, and 2) a solution where DRL is used for scheduling while a heuristic determines the MCS. This comparison highlights the benefits of joint scheduling and MCS allocation.

Appendix A provides the foundations of machine learning (ML) concepts such as MDP, DRL, and DNN architectures. Reading this appendix is recommended for readers unfamiliar with these concepts.

Publications

The work of this thesis has led to the following publications.

International conference

IC1. S. Nérondat, X. Leturc, C. J. Le Martret and P. Ciblat, "Transformer-Based Packet Scheduling Under Strict Delay and Buffer Constraints," *IEEE Wireless Communications and Networking Conference (WCNC)*, Milan (Italy), March 2025.

IC2. S. Nérondat, X. Leturc, P. Ciblat and C. J. Le Martret, "Efficient 5G Resource Block Scheduling Using Action Branching and Transformer Networks," *IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, Barcelona (Spain), May 2025.

French conference

NC1. S. Nérondat, X. Leturc, C. J. Le Martret and P. Ciblat, "Deep Q-Learning pour l'ordonnancement de paquets sous contraintes strictes de latence et de taille de buffer," *Colloque GRETSI*, Grenoble (France), September 2023.

NC2. S. Nérondat, X. Leturc, C. J. Le Martret and P. Ciblat, "Ordonnancement et ACM conjoint sur canal aléatoire basé sur un transformer entrainé par apprentissage profond par renforcement," *Colloque GRETSI*, Strasbourg (France), August 2025.

Patent

P1. S. Nérondat, X. Leturc, C. J. Le Martret and P. Ciblat, "Procédé d'ordonnancement dynamique de communications entre une pluralité d'équipements utilisateurs," FR 24 11578, October 2024.

Chapter 1

General context and state of the art on scheduling

1.1 Introduction

This chapter introduces both conventional heuristics and DRL-based solutions for scheduling and resource allocation in wireless communications systems. Packet scheduling is a critical mechanism, determining how data packets are prioritized and transmitted to ensure efficient resource utilization and/or low latency and/or high-quality service.

Scheduling and resource allocation are performed centrally by the RRM. In this thesis, we assume that scheduling prioritizes data flows over the considered dimensions (time and frequency), whereas resource allocation refers, for instance, to the selection of MCS and transmit power.

Historically, scheduling algorithms have been based on heuristics tailored to specific goals (more details are provided in the rest of the chapter). These heuristics operate with limited information on flows and buffers, such as the average or instantaneous data rate, the packet delay, or the number of packets per buffer for instance.

More recently, approaches utilizing DNN architectures trained with DRL have emerged to tackle the scheduling challenge. A key advantage of DNN architectures is their flexibility in incorporating numerous features, offering greater flexibility than conventional heuristics. Additionally, DRL enables defining customized objective function through the reward.

This chapter is organized as follows. Section 1.2 introduces a general system model and the notations. Section 1.3 reviews the state of the art (SotA) in the existing heuristic scheduling methods. Section 1.4 provides a SotA review, a classification, and a critical analysis of the existing DRL schedulers. Finally, Section 1.5 draws concluding remarks.

1.2 General system model

We consider a wireless communication system with n_L UEs, each supporting n_{QoS} types of traffic. Data associated with each type of traffic are stored in the form of packets in first in, first out (FIFO) buffers of finite capacity of B packets. The total number of buffers is thus $n_Q = n_L n_{\mathrm{QoS}}$. The index of the tth buffer of the ℓth link is $i = \ell n_{\mathrm{QoS}} + t$, with $t \in \{0, \ldots, n_{\mathrm{QoS}} - 1\}$ and $\ell \in \{0, \ldots, n_L - 1\}$. Remark that $\ell = \left\lfloor \frac{i}{n_{\mathrm{QoS}}} \right\rfloor$ and $t = i \bmod n_{\mathrm{QoS}}$. In the rest of the thesis, we use equivalently the index i or the pair (ℓ_i, t_i) for the buffer i, where ℓ_i represents the link associated with buffer i and t_i indicates its type

of traffic. We assume that the n_{QoS} traffic flows can be categorized into two types:

- n_{DC} delay constraint (DC) traffic flows, which require data transmission within a specified delay threshold,
- $n_{\mathrm{BE}} = n_{\mathrm{QoS}} - n_{\mathrm{DC}}$ traffic flows that have no strict delay constraint. For simplicity, we refer to the latter as best-effort (BE) traffic.

Let us note the corresponding indexes of DC traffic $t_i \in \{0, \ldots, n_{\mathrm{DC}} - 1\}$ and the corresponding indexes of BE traffic $t_i \in \{n_{\mathrm{DC}}, \ldots, n_{\mathrm{QoS}} - 1\}$.

We define the slot as the smallest unit of time. Each packet in each buffer i is characterized by its waiting time (WT) that is initially set to 0, and which is incremented by one at each slot. This WT must respect a constraint D_{t_i}, which depends on the quality of service (QoS) of buffer i. For DC traffic, i.e. $t_i < n_{\mathrm{DC}}$, $D_{t_i} < +\infty$, while for BE traffic, $D_{t_i} = +\infty$. Packets from DC traffic must be sent before reaching D_{t_i}. If a packet's WT exceeds D_{t_i}, a DV occurs. In Chapters 2 and 4, this results in the packet being dropped, whereas in Chapter 3, the packet remains stored in the buffer.

Let $d_{u,i} \in \{-1, 0, \ldots, D_{t_i}\}$ be the WT of the uth packet in the ith buffer with $u \in \{0, \ldots, B-1\}$ assuming that the packets are ordered in the buffer according to their WT in the decreasing order, i.e. $d_{0,i} \geq d_{1,i} \geq \cdots \geq d_{B-1,i}$, and by convention -1 represents an empty entry. Let n_i be the number of packets in buffer i for a given slot i.e. the number of entries $d_{u,i}$ greater than -1. In the scheduling literature, the oldest packet is conventionally called the head of line (HoL). As a consequence, $d_{0,i}$ is identified as the HoL delay of buffer i.

Remark: the value of $d_{u,i}$ and n_i depends on the slot index, which is omitted to lighten the notations. Let us note n_i^{HoL} the number of packet with a WT equal to $d_{0,i}$ in the buffer i. Figure 1.1 depicts an example of a buffer with n_i packets.

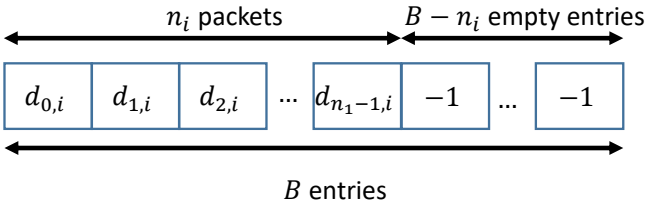


Figure 1.1: Representation of buffer i.

The transmission bandwidth of the considered system is denoted by W and is divided into N_f RBs. A central scheduler called RRM is responsible for allocating these RBs to the various buffers. During each scheduling interval, N_f buffers are selected, where the same buffer can be allocated multiple RBs. Then, the bits of the oldest packets from the selected buffers are extracted and transmitted over the channel on their assigned RBs. The number of extracted bits depends on the channel capacity of the corresponding UEs on the RBs. After the transmission of the bits, the WT of the remaining packets in the buffers is incremented by one. Then, new packets arrive in the buffers and their WT is set to 0. Packets arriving in a given buffer once it is full are discarded and a BO occurs, thus leading to packet loss.

Figure 1.2 represents a global view of the considered system model. The buffers are shown at the top of the figure along with their status and corresponding CSI. The CSI is obtained through feedback from the receiver represented at the center of the figure. These inputs are processed and provided to a scheduler responsible for assigning buffers to the available RBs. A MCS is selected for each transmission. This selection can be performed either jointly with buffer scheduling, i.e. using a single module, or separately by employing two distinct modules, i.e. one for buffer scheduling and another for MCS selection. The

frame is then built with the different selected buffers and MCS. To do that, the different RBs for each slot are allocated to the different UEs, represented at the bottom left of the figure. The corresponding bits of the selected buffers are transmitted over the propagation channel. The receiver attempts to decode the received bits, which may or may not result in errors depending on the channel and Rx model, as illustrated on the right side of the figure. Other losses may occur due to DV and BO. This system model aligns with the 5G framework described in Section 1.2.2.

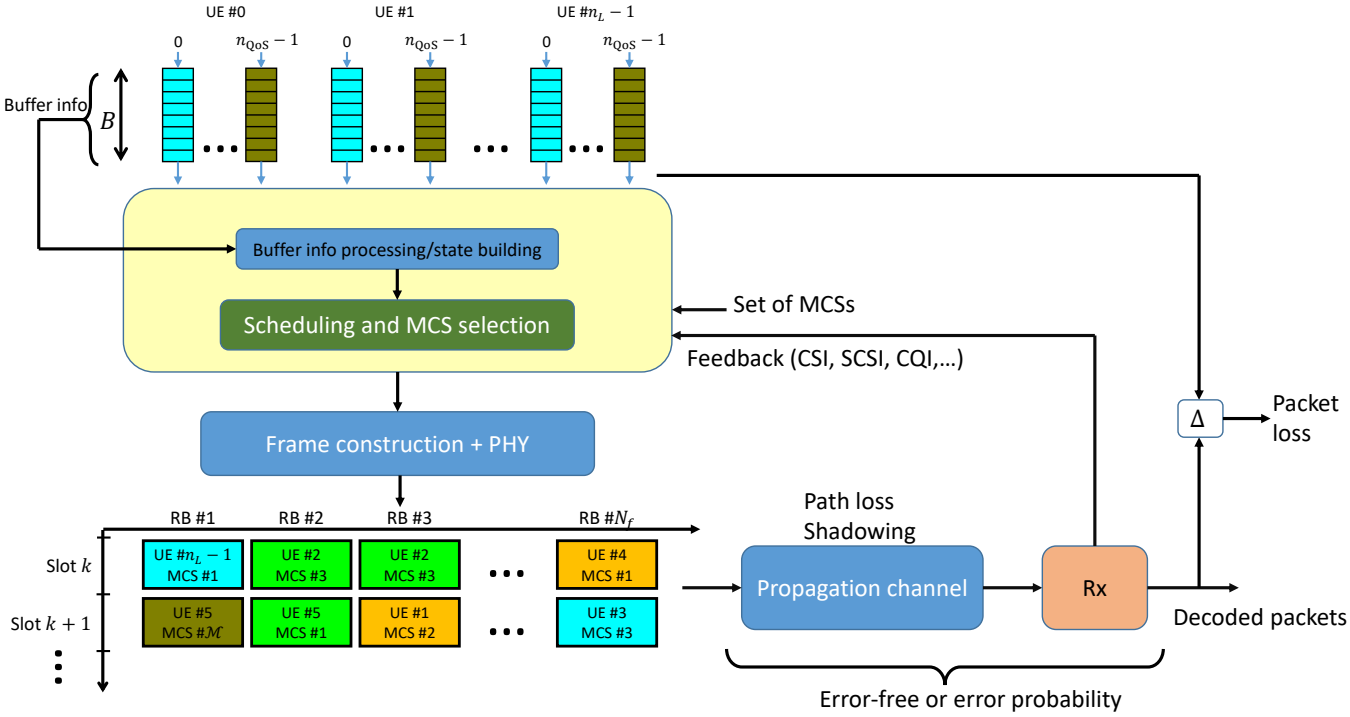


Figure 1.2: Illustration of the general system model.

Additional information and precision of the system model are provided in the dedicated chapters. These refinements are related to the considered types of traffic, the packet arrival model, and the channel model.

1.2.1 Additional information and precisions on the general system model in the different chapters

The proposed scheduling solutions studied in this thesis are discussed across three chapters, Chapter 2 to Chapter 4. The system models used in these chapters differ, which may complicate the reading, as there is no continuous transition between them. The reasons for this are as follows:

- Chapter 2 presents the first work in the thesis, where the system model is inspired by [5], with two-class traffic and a single RB.
- Chapter 3 introduces the second work, where we opted for a system model closer to the 5G system. We used the WS environment designed for this context, which required a system model imposed by the wireless suite (WS) framework—namely, multiple RBs ($N_f > 1$) and a four-class traffic model.

- Chapter 4 presents the final work in the thesis, which focuses on joint scheduling and MCS allocation in a frame-based context. Since we needed to implement a Rayleigh channel model to account for packet errors at the receiver side, adapting the WS framework to this context would have required modifications beyond the scope of the thesis. Instead, we used the same traffic model as in Chapter 2, but with multiple RB ($N_f > 1$) and a packet-based error model.

Thus, this section is dedicated to explaining the different system models used in these three chapters.

1.2.1.1 Additional information and precisions for Chapter 2

In Chapter 2, we assume that there is a single RB, i.e. $N_f = 1$. Thus the scheduling is performed only in the time dimension and is referred to as *slot-based* model. Two types of traffic are considered, i.e. $n_{\mathrm{QoS}} = 2$ with one DC and one BE. The packets of the different traffic are of the same size and arrive in each buffer i according to a Poisson distribution of parameter λ_i. The considered channel is error-free and allows to transmit a fixed number of packets, different for each UE.

1.2.1.2 Additional information and precisions for Chapter 3

In Chapter 2, we assume that there are several RBs to be allocated, i.e. $N_f > 1$. For a given slot, the set of RBs is called a *frame*. Thus the scheduling is performed in both time and RB dimensions, and is referred to as *frame-based* model. The WS environment [6] is used for the simulations. As a consequence, four types of traffic are considered, i.e. $n_{\mathrm{QoS}} = 4$: three guaranteed bit rate (GBR) and one non-GBR. The packets of the different traffic are of different size and the packet arrival depends on the type of traffic. The considered channel is error-free and the number of bits that can be transmitted for each RB depends on the channel quality indicator (CQI) of the allocated UE.

1.2.1.3 Additional information and precisions for Chapter 4

In this chapter, we assume a frame-based model, along with the traffic model of Chapter 2, i.e. $n_{\mathrm{QoS}} = 2$ with one DC and one BE. We assume a Rayleigh channel varying from RB to RB, with a fixed average signal-to-noise ratio (SNR) per UE that is known from the RRM. In this chapter, the scheduler also performs MCS selection, and packets are subject to errors depending on the channel realizations.

1.2.1.4 Synthesis of the additional information and precisions

Tables 1.1, 1.2 and 1.3 summarize the principal changes of the system model across the different chapters for the buffer model, the channel model and the bit management model respectively.

Table 1.1: Comparison of the buffer models across the different chapters.

	n_{QoS}	n_Q	Packet arrival
Chapter 2	2	$2n_L$	λ (Poisson)
Chapter 3	4	$4k$ with $k \in \mathbb{N}$	Depends on the type of traffic
Chapter 4	2	$2n_L$	λ (Poisson)

Table 1.2: Comparison of the channel models across the different chapters.

	Channel model	Errors	Variability
Chapter 2	Capacity limited	No	Fixed
Chapter 3	Capacity limited	No	Random (shadowing)
Chapter 4	Rayleigh	Yes	Random (fixed stat.)

Table 1.3: Comparison of the bit management models across the different chapters.

	MCS selection	Extracted bits
Chapter 2	No	Fixed
Chapter 3	No (determined by the CQI)	Variable (function of the CQI)
Chapter 4	Yes	Variable (function of MCS)

1.2.2 Mapping with the 5G model

In practical communication systems such as 5G, data belongs to different applications. The data, in the form of packets of bits, originate from the IP layer and are transmitted to layer 2 of the open systems interconnection (OSI) model.

Figure 1.3 depicts the four sub-layers composing layer 2 in 5G. A complete description of these layers is provided in [7, Chapter 6] and in [8, 9, 10, 11] for service data adaptation protocol (SDAP), packet data convergence protocol (PDCP), radio link control (RLC) and medium access control (MAC) sub-layers respectively.

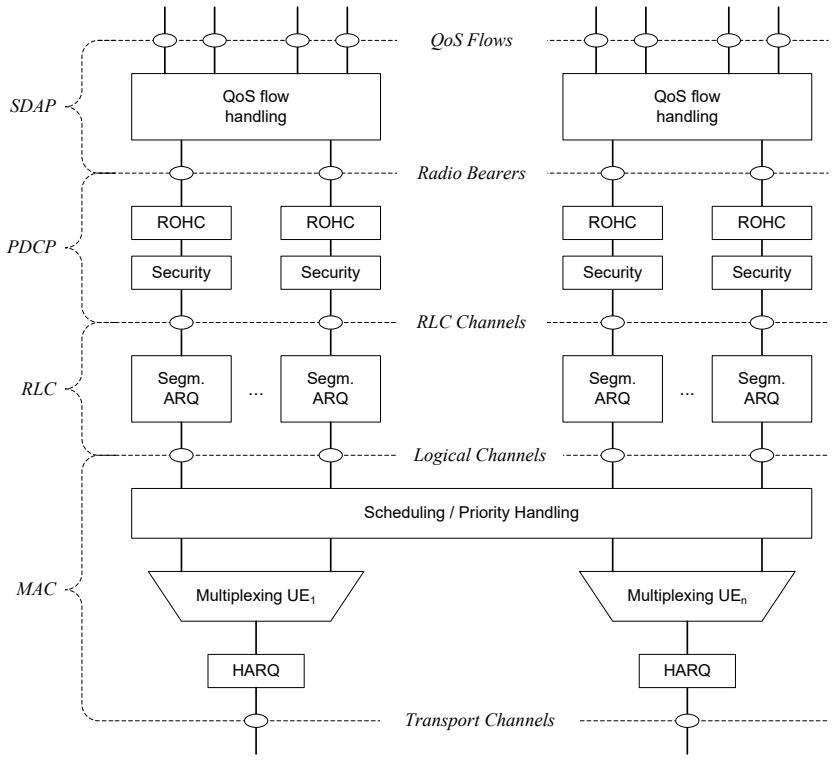


Figure 1.3: Downlink Layer 2 structure (from [12, Section 6.1]).

The different sub-layers work as follows:

- The SDAP sub-layer [8] maps the different QoS flows to appropriate data radio bearer (DRB), ensuring that each flow is handled according to its QoS requirements, as identified by its QoS flow identifier (QFI). Figure 1.4 provides an illustrative example using real application traffic such as WhatsApp, Skype, YouTube, and Netflix. It shows how IP flows are classified through traffic flow template (TFT), associated with specific QFIs, and mapped to corresponding DRBs. The user plane function (UPF) is a gateway between the radio access network (RAN) and external networks such as the internet [7]. The SDAP layer was introduced in the 5G. In this thesis, we considered that the QoS flows are already mapped into radio bearers and by abuse of language we sometimes refer to the radio bearers as QoS flows or types of traffic.
- The PDCP sub-layer [9] which performs robust header compression (ROHC) and ciphering.
- The RLC sub-layer [10] which performs internet protocol (IP) packets fragmentation and automatic repeat request (ARQ). In this thesis, we assume that the packets are already fragmented and we aim to transmit correctly the fragment that we call by abuse of language "packets".
- The MAC sub-layer [11] which performs scheduling and resource allocation of the different fragment and manages hybrid automatic repeat request (HARQ). The *multiplexing* operation of the MAC layer in Figure 1.3 corresponds to taking the data from several logical channels (control or user data) that have been granted resources by the scheduler, and assembling them into one *transport block*. A transport block is the basic data unit exchanged between the PHYsical (PHY) and the MAC layers, according to the definition given in [13]. The power level and MCS are selected at the MAC layer and are then applied at the PHY layer (layer 1 of the OSI model).

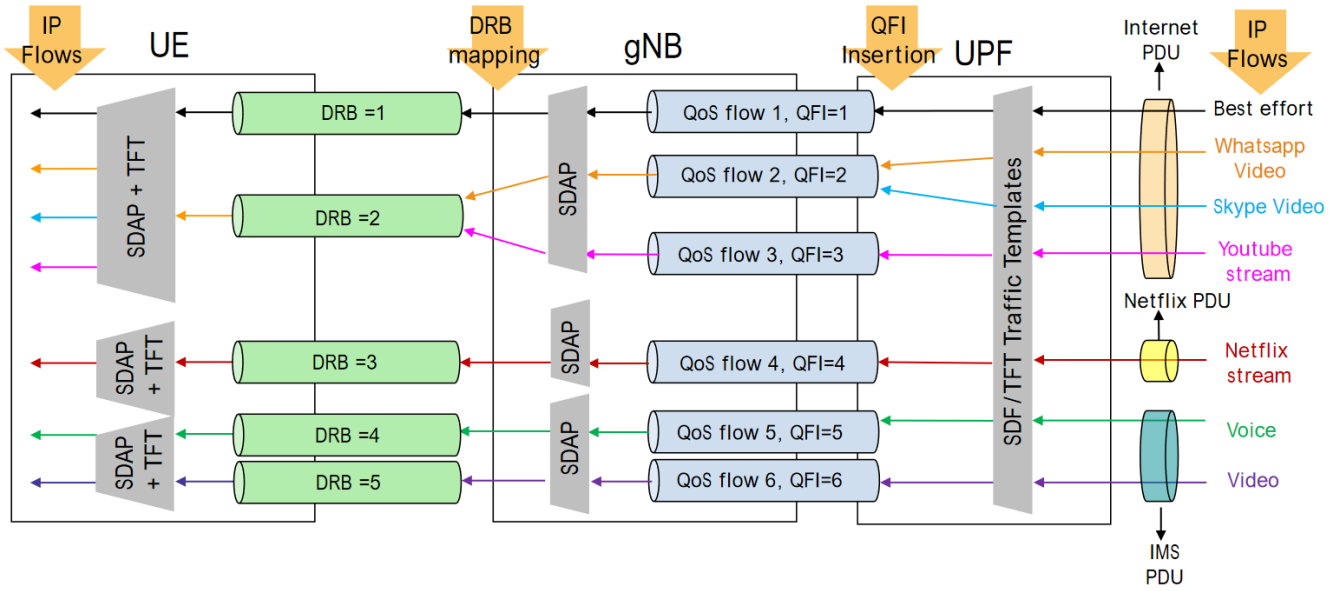


Figure 1.4: QoS flows to radio bearers mapping (from `https://devopedia.org/5g-quality-of-service`).

1.3 State of the art of heuristics for scheduling

This section reviews existing work on heuristic methods, which will serve as baseline comparisons for packet scheduling. Several surveys have already been conducted on scheduling algorithms in both long term evolution (LTE) and 5G networks. For instance, [14] focuses on heuristics only for LTE, while [15] provides a review for both heuristic and DRL schedulers for 5G. More recently, [16] has introduced a LLM-based approach that generates scheduling heuristics from natural language intents and offers an overview of various existing algorithms.

The survey in [14] categorizes scheduling algorithms along the following classification:

- Channel-unaware (CU) or channel-aware (CA),
- QoS-unaware (QU) or QoS-aware (QA),

leading to four groups:

1. CU-QU,
2. CA-QU,
3. CU-QA,
4. CA-QA.

In contrast, [15] classifies scheduling algorithms based on both their parameters and their performance objectives. The classification proposed in [15] provides a fine level of granularity, but tends to be overly specific, often resulting in characterizing each algorithm individually. By contrast, the classification from [14] offers a more coherent grouping, allowing for meaningful comparisons while still capturing the essential characteristics of each algorithm. This is why we adopt the classification from [14] in this chapter.

The general buffer selection rule of heuristic based schedulers can be written as [14, Eq. (1)]:

$$i_k^* = \arg\max_i h_{\text{heur}}(\mathbf{x}_{i,k}), \tag{1.1}$$

where $\mathbf{x}_{i,k}$ are the features of buffer i of the scheduling index k (i.e. combining slot and RB) and h_{heur} is the function computing the metric of heuristic heur.

Table 1.4 presents a subset of popular heuristics schedulers. Other heuristics can be found in [14], [15] or [16] for instance.

Table 1.4: Subset of popular heuristics schedulers.

Heuristic	Date	References	Channel-aware	QoS-aware
Round-Robin	1964	[17]	No	No
EDF	1969	[18]	No	Yes
PF	1999 (presentation in Bell labs - see [19, ref. 3])	[20]	Yes	No
LWDF	2001 (submitted in 1999)	[21]	No	Yes
EXP-rule	2000	[22] [23]	Yes	Yes
MLWDF	2004 (submitted in 2000)	[24]	Yes	Yes
EXP/PF	2003	[25]	Yes	Yes
GPF	2005	[26]	Yes	No
LOG-rule	2009	[27] [28]	Yes	Yes
Knapsack	2013	[29], [30] (and adapted in [6])	Yes	Yes

A criterion to evaluate the performance of a scheduling algorithm (or policy) is the analysis of its stability region, which characterizes the set of arrival rates for which the scheduling policy keeps all buffers stable. The definitions of stability region and stable buffer assume that the buffers have infinite capacity, which does not hold in our case. In the case of an infinite-capacity buffer, the number of packets $n_{i,k+1}$ at step $k+1$ depends on the number of packets the channel allows to transmit, denoted $c_{i,k}$, at step k, the number of arriving packets $n^r_{i,k}$ (a random variable), and the number of packets already in the buffer $n_{i,k}$ at step k:

$$n_{i,k+1} = \max(n_{i,k} - c_{i,k}, 0) + n^r_{i,k}. \tag{1.2}$$

Note that in case of finite buffers with length B, (1.2) becomes:

$$n_{i,k+1} = \min\left(\max(n_{i,k} - c_{i,k}, 0) + n^r_{i,k}, B\right). \tag{1.3}$$

Remark: We explicitly write the index k for the number of packets $n_{i,k}$ for the sake of clarity, unlike in Section 1.2 where it was omitted to lighten the notation.

Let us note λ_i and $\bar{c}_i$ the arrival rate, i.e. the number of packets arriving in average at each slot, and the "service" rate, i.e. the average number of packets that it is possible to extract at each slot, for buffer i. Mathematically:

$$\lambda_i = \lim_{K\to\infty} \frac{1}{K+1} \sum_{k=0}^{K} \mathbb{E}[n^r_{i,k}] \tag{1.4}$$

and

$$\bar{c}_i = \lim_{K\to\infty} \frac{1}{K+1} \sum_{k=0}^{K} \mathbb{E}[c_{i,k}] \tag{1.5}$$

Let us define the notions of stability. The definition of a *strongly stable buffer* is given in [31, Definition 3.1], and signifies that the number of packets in the buffer does not grow indefinitely.

Definition 1 ([31, Definition 3.1])**.** A buffer i is called *strongly stable* if:

$$\limsup_{K\to\infty} \frac{1}{K+1} \sum_{k=0}^{K} \mathbb{E}\left[n_{i,k}\right] < \infty.$$

According to [31, Lemma 3.6]:

$$\lambda_i \leq \bar{c}_i \tag{1.6}$$

is a necessary condition for strong stability, and:

$$\lambda_i < \bar{c}_i \tag{1.7}$$

is a sufficient condition for strong stability.

Now, let us define the *stability* of a network.

Definition 2 ([31, Definition 3.2])**.** A network is *strongly stable* if all individual buffer of the network are strongly stable.

With n_Q buffer the condition for the network stability, for a single server (i.e. in our case one RB), is [32, Section 3.4.2] [33]:

$$\sum_{i=0}^{n_Q-1} \frac{\lambda_i}{\bar{c}_i} < 1. \tag{1.8}$$

For N_f servers (i.e. N_f RBs), in a time-slotted system, the condition ensuring network stability can be expressed as [34]:

$$\sum_{i=0}^{n_Q-1} \frac{\lambda_i}{\bar{c}_i} < N_f. \tag{1.9}$$

In contrast, for non time-slotted systems, where packets remain in service until completion, if there are multiple servers and different service rates for the different buffers, the analysis becomes quite intractable [32, Section 3.4.2.3].

Now let us define the stability region.

Definition 3 ([35, Adaptation of Definition 3.2])**.** The stability region $\mathbf{\Lambda}_\pi$ of policy π is the set of multiclass arrival rate $\boldsymbol{\lambda} = [\lambda_0, \ldots, \lambda_{n_Q-1}]$ for which the system (i.e. a queuing network) is stable under policy π.

Now, let us define a *throughput-optimal* policy:

Definition 4 (Adaptation of [24])**.** A throughput-optimal policy is a policy maximizing the stability region.

Definition 4 implies that, under a throughput-optimal policy, the number of packets in the buffer does not grow indefinitely.

Among the different policies from Table 1.4, the modified largest weighted delay first (MLWDF), the exponential rule (EXP-rule) and the logarithmic rule (LOG-rule) are *throughput-optimal* [24]. However, policies such as round-robin (RR) or proportional fair (PF) are not throughput-optimal.

The rest of this section is organized as follows. Section 1.3.1 presents the RR. Section 1.3.2 presents the earliest deadline first (EDF). Section 1.3.3 presents the PF and the generalized PF (GPF). Section 1.3.4 presents the largest weighted delay first (LWDF) and the MLWDF. Section 1.3.5 presents

the LOG-rule. Section 1.3.6 presents the EXP-rule and the EXP/PF. Section 1.3.7 presents the knapsack (KP).

Remark: In the following, the value of $d_{0,i}$ depends on the index k, which was omitted in Section 1.2 for simplicity of notation but is made explicit here for clarity. Accordingly, we denote $d_{0,i,k}$ the HoL of buffer i at index k.

1.3.1 Round-robin

The RR is a CU-QU scheduling algorithm aiming to assign equal time or resource shares to buffers in a cyclic order. The expression of the metric of RR is:

$$h_{\mathrm{RR}}(\mathbf{x}_{i,k}) := T_{i,k}, \tag{1.10}$$

where $T_{i,k}$ corresponds to the number of RBs that have elapsed since buffer i was served. It is set to 0 when it is scheduled, yielding:

$$T_{i,k} = \begin{cases} T_{i,k-1} + 1 & \text{if buffer } i \text{ is not scheduled at slot } k \\ 0 & \text{if buffer } i \text{ is scheduled.} \end{cases} \tag{1.11}$$

1.3.2 Earliest deadline first

The EDF is a CU-QA scheduling algorithm aiming to schedule buffers with the shortest deadline. The expression of the metric of EDF is:

$$h_{\mathrm{EDF}}(\mathbf{x}_{i,k}) := \frac{1}{D_{t_i} - d_{0,i,k}}. \tag{1.12}$$

Note that for BE traffic introduced in Section 1.2, $D_{t_i} = +\infty$. As a result, $h_{\mathrm{EDF}}(\mathbf{x}_{i,k}) = 0$ for all the BE buffers, meaning that they will never be scheduled. Therefore, the EDF metric is not suitable to handle both DC and BE traffic flows.

1.3.3 Proportional fair

PF [20] is a CA-QU scheduling algorithm selecting UE with favorable channel conditions while maintaining fairness between UE/buffers by considering the proportional gain relative to past average achieved rate. The expression of the metric of PF is:

$$h_{\mathrm{PF}}(\mathbf{x}_{i,k}) = \frac{c_{i,k}}{\bar{c}_{i,k-1}}, \tag{1.13}$$

where $\bar{c}_{i,k}$ is the average achieved rate for buffer i during the previous slots. In this thesis, we consider the empirical average rate:

$$\bar{c}_{i,k} = \frac{1}{k} \sum_{k'=1}^{k} c_{i,k'} \delta_{k'}(i), \tag{1.14}$$

with,

$$\delta_k(i) = \begin{cases} 1 & \text{if } i \text{ is selected for the combination RB/slot } k \\ 0 & \text{otherwise} \end{cases}. \tag{1.15}$$

Note that in the literature, $\bar{c}_{i,k-1}$ in (1.13) is conventionally replaced by the average rate estimation $\tilde{c}_{i,k-1}$, which uses a sliding window and can be written as:

$$\tilde{c}_{i,k} = (1 - \tau_{\mathrm{PF}})\tilde{c}_{i,k-1} + \delta_k(i)\tau_{\mathrm{PF}}c_{i,k}. \tag{1.16}$$

Equation (1.16) requires tuning an hyperparameter $\tau_{\mathrm{PF}} \in]0, 1[$, which is not necessary in the estimator (1.14).

In [26], a general version of the PF algorithm is introduced. This version called GPF is given by:

$$h_{\mathrm{GPF}}(\mathbf{x}_{i,k}) = \frac{(c_{i,k})^{\alpha}}{(\bar{c}_{i,k})^{\beta}}, \tag{1.17}$$

where $\alpha \geq 0$ and $\beta \geq 0$ are parameters tuning the trade-off between fairness and throughput. With $\alpha = \beta = 1$, conventional PF is achieved. With $\alpha > 0$ and $\beta = 0$, only the achievable throughput is taken into account, yielding max throughput (MT) (also called "max rate" or "max SNR" or "best CQI") scheduling algorithm. With $\alpha = 0$ and $\beta > 0$, only the past average throughput is taken into account, yielding blind equal throughput (BET) scheduling algorithm [14].

1.3.4 MLWDF

Let us first introduce the LWDF [21] scheduling algorithm, which is a CU-QA scheduling algorithm whose expression is:

$$h_{\mathrm{LWDF}}(\mathbf{x}_{i,k}) := \eta_i d_{0,i,k}, \tag{1.18}$$

where η_i is a weight which depends on the type of traffic of buffer i.

The MLWDF [24, 36], which is derived from LWDF, takes into account user specific time-varying channel for wireless communications. Therefore, MLWDF is a CA-QA scheduling algorithm that prioritizes users based on a weighted metric combining channel quality and the HoL. The expression of the metric of MLWDF is:

$$h_{\mathrm{MLWDF}}(\mathbf{x}_{i,k}) := \alpha_{i,k}\eta_i c_{i,k} d_{0,i,k}, \tag{1.19}$$

where $\alpha_{i,k}$ and η_i are hand-tuned parameters. Common choices are $\alpha_{i,k} = \frac{1}{\bar{c}_{i,k-1}}$ and $\eta_i = \frac{-\log(10^{-2})}{D_{t_i}}$ [14].

Note that for BE traffic introduced in Section 1.2, $D_{t_i} = +\infty$ and thus $\eta_i = 0$. As a result, $h_{\mathrm{MLWDF}}(\mathbf{x}_{i,k}) = 0$ for all BE buffers, meaning they will never be scheduled. To alleviate this undesirable behavior, we follow the recommendation from [14] suggesting that MLWDF is used for DC traffic and PF is used for BE traffic. Since the PF is not throughput-optimal (Definition 4), BE buffers may become unstable, potentially leading to significant BO.

1.3.5 LOG-rule

LOG-rule [27, 28] is also a CA-QA scheduling algorithm that prioritized user based on the combination of channel quality metric and a logarithm function of the HoL packet delay. The expression of the metric of LOG-rule is:

$$h_{\mathrm{LOG-rule}}(\mathbf{x}_{i,k}) := \alpha_{i,k} c_{i,k} \log\left(\beta_i + \eta_i d_{0,i,k}\right), \tag{1.20}$$

where $\alpha_{i,k}$, β_i and η_i are hand-tuned parameters. Common choices are $\alpha_{i,k} = \frac{1}{\bar{c}_{i,k-1}}$, $\beta_i = 1.1$ and $\eta_i = \frac{5}{0.99 D_{t_i}}$ such as recommended in [37] and in [14].

Note that for BE traffic, $\eta_i = 0$ because $D_{t_i} = +\infty$. As a result, for BE buffers the LOG-rule metric is equal to $\frac{c_{i,k}}{\bar{c}_{i,k-1}} \log(1.1)$, which is the PF metric multiplied by a factor close to 0, meaning that they would rarely be scheduled. As for MLWDF, the LOG-rule expression is used for DC traffic and PF expression is used to handle BE traffic.

1.3.6 EXP-rule

EXP-rule [23] is also a CA-QA scheduling algorithm that prioritizes users based on the combination of channel quality metric and an exponential function of the HoL packet delay. The expression of the metric of EXP-rule is:

$$h_{\mathrm{EXP-rule}}(\mathbf{x}_{i,k}) := \alpha_{i,k} c_{i,k} \exp\left(\frac{\eta_i d_{0,i,k}}{1 + \sqrt{\overline{\chi}_k}}\right) \tag{1.21}$$

where α_i and η_i are hand-tuned parameters. Common choices are $\alpha_{i,k} = \frac{1}{\bar{c}_{i,k-1}}$, $\eta_i \in \left[\frac{5}{0.99 D_{t_i}}, \frac{10}{0.99 D_{t_i}}\right]$ and $\overline{\chi}_k = \frac{1}{n_L n_{\mathrm{DC}}} \sum_{\ell=0}^{n_L-1} \sum_{t=0}^{n_{\mathrm{DC}}-1} \eta_{(\ell,t)} d_{0,(\ell,t),k}$ in [37], corresponding to the average weighted HoL WT of DC buffers. Thanks to the exponential, the buffers with urgent delay are more prioritized than with the MLWDF or LOG-rule. The denominator in the exponential tends to smooth the delay, in order not to give it too much importance compared to the channel condition. Another expression of the metric of EXP-rule is given in [23]:

$$h_{\mathrm{EXP-rule}}(\mathbf{x}_{i,k}) := \alpha_{i,k} c_{i,k} \exp\left(\frac{\eta_i d_{0,i,k} - \overline{\chi}_k}{1 + \sqrt{\overline{\chi}_k}}\right) \tag{1.22}$$

where the parameters $\alpha_{i,k}$, η_i and $\overline{\chi}_k$ are the same as for (1.21). It is worth noting that if there are only DC traffic, $\overline{\chi}_k$ at the numerator can be dropped without changing the rule as it is common for all buffers.

Another version of the EXP-rule, called in the literature EXP/PF, handle the DC traffic with (1.22) and the BE traffic with (1.13). In this case, the $\overline{\chi}_k$ at the numerator tends to de-prioritize DC traffic to serve more often BE traffic.

1.3.7 Knapsack

The KP from [29] and [30] is defined as the weighted sum of hyperbolic tangent. The expression of the metric of KP is:

$$h_{\mathrm{KP}}(\mathbf{x}_{i,k}) := \sum_{j=1}^{4} \alpha_j \tanh\left(v_j(\mathbf{x}_{i,k})\right), \tag{1.23}$$

with $\alpha_j > 0$ for $j \in \{1, 2, 3, 4\}$ and

$$v_1(\mathbf{x}_{i,k}) := \frac{d_{0,i,k}}{D_{t_i}} \tag{1.24}$$

$$v_2(\mathbf{x}_{i,k}) := \frac{\xi_{i,k}}{\xi_i^{\mathrm{target}}} \tag{1.25}$$

$$v_3(\mathbf{x}_{i,k}) := \frac{10 - \rho_i}{9} \tag{1.26}$$

$$v_4(\mathbf{x}_{i,k}) := \frac{n_{i,k}}{B}, \tag{1.27}$$

where $\xi_{i,k}$ is the achieved packet loss rate (PLR) for buffer i at step k, ξ_i^{target} is the acceptable PLR defined by in the QoS tables [38] and $\rho_i \in \{1, \ldots, 9\}$ is the bearer priority of buffer i with 1 being the highest priority. If we consider that $\xi_{i,k}$ in (1.25) includes the loss due to channel, this version of KP is a CA-QA scheduling algorithm. Otherwise, this version of KP is a CU-QA scheduling algorithm.

1.4 State of the art of DRL solutions for scheduling

To the best of our knowledge, new approaches based on DNN architectures trained using DRL have been introduced to tackle the scheduling problem since 2017 with [39]. One advantage of DNN architectures is their ability to accommodate as many features as desired. This makes it straightforward to increase the number of features at the input of the DNN compared to heuristics. Moreover, since DRL operates in a model-free context, it allows for the design of customized objectives through the reward function.

1.4.1 Classification of the DRL schedulers

One can find several surveys on scheduling with DRL in the literature. For example, [15] focuses primarily on performance objectives, without addressing the design of state and action spaces. Similarly, [40] provides a more comprehensive review, but is limited to a specific family of RB allocation strategies, namely the fine-grained approach as defined in [41].

In this thesis, we propose a new classification of the DRL approaches according to their action spaces, state spaces and rewards. The action space characterizes how the RBs are allocated. The state space includes the features used for decision-making and can be compared to the channel and QoS awareness of the heuristics, as it will be discussed later in this section. The reward is defined according to the performance goal.

1.4.1.1 Classification based on the action space

The outputs of the DNN determine the possible actions, and as a consequence determine how the RBs are allocated. The action may, for example, correspond to the index of the buffer to be scheduled, the index of a specific buffer/RB combination, or the proportion of RBs allocated to each UE as determined by a softmax. Consequently, the classification of RB allocation approaches is essentially a characterization of the action space used by the different DRL-based schedulers.

In the taxonomy of [41], two families are distinguished: the fine-grained approach, where resource allocation is performed directly by the DNN architecture, and the coarse-grained approach, where the DNN architecture selects an heuristic that carries out the allocation.

One of the contributions of this thesis is to extend the classification from [41] by introducing additional sub-classes, thereby refining the taxonomy while maintaining its overall coherence. Figure 1.5 represent the proposed new categories:

- We add the sub-category *tune heuristic* in the coarse-grained category, to include heuristic parameters tuning.
- We add two sub-categories in the fine-grained:
  - The sequential RB scheduling (SRS) approach, where the agent performs allocation RB per RB. To fill the whole frequency band, composed by N_f RBs, the agent has to perform N_f inference steps.
  - The global RB scheduling (GRS) approach, where the agent allocates directly the set of RBs to the different buffers, in one inference step. We identified two possibilities:

* The architecture provides a proportion of RBs to be allocated per UEs, which is referred to as GRS-proportion (GRS-P).
* The architecture selects the UE to be scheduled for each RB, which is referred to as GRS-index (GRS-I).

It should be noted that the acronyms SRS, GRS-I, and GRS-P along with the corresponding classifications are newly proposed and were first introduced in our prior work [42].

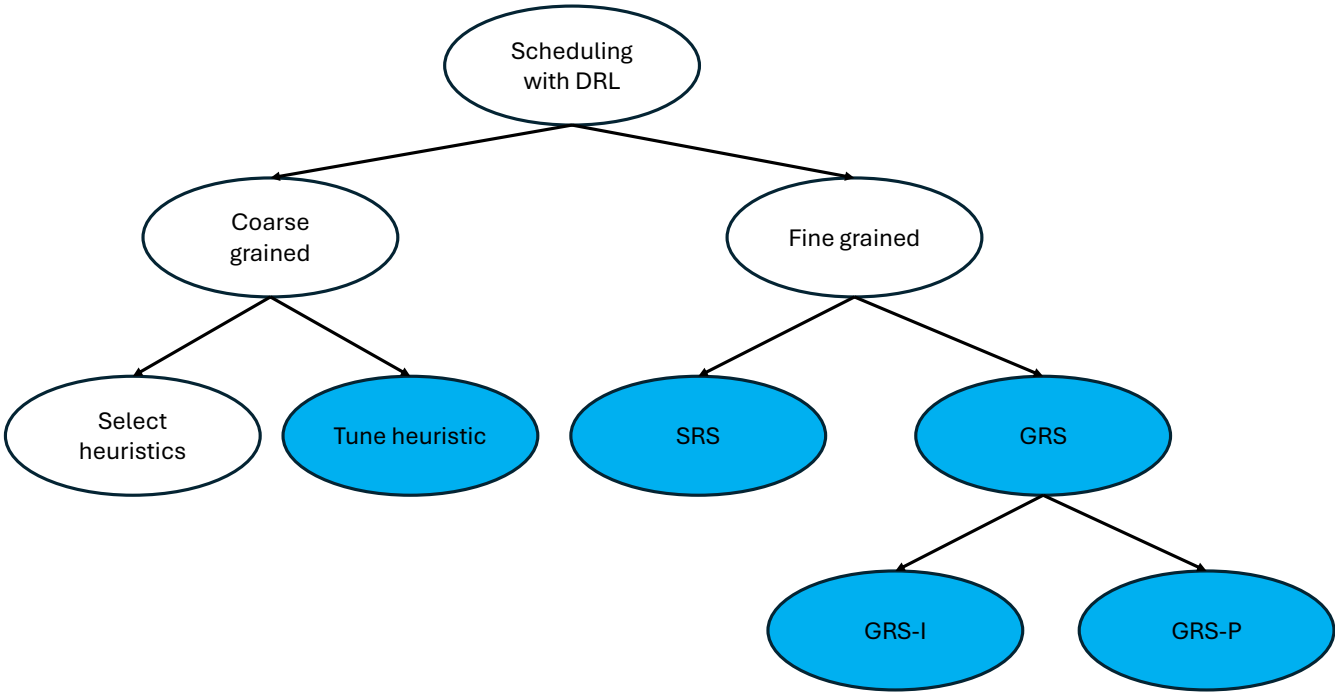


Figure 1.5: Classification of the different DRL methods for scheduling. White ovals are from [41], and blue ones represent our proposed new classification categories.

Let us classifies the different DRL schedulers based on their action space, state space, reward and studied performances.

Table 1.5 presents the action space of different DRL schedulers from the literature into three main categories: coarse-grained, fine-grained, and other approaches.

Table 1.5: Classification based on the action space.

Coarse-grained approach		Fined grained approach			Other
Selection of heuristic	Heuristic parameter tuning	SRS	GRS-P	GRS-I	
[39] [43] [44]	[45] [46]	[47] [48][49] [50] [41] [51] [52] [53] [54]	[55] [56] [57] [58] [59] [60]	[61] [62] [63] [64] [65] [66]	[67] [68]

In coarse-grained methods, at each time step, the agent either selects a scheduling heuristic from a predefined set [39, 43, 44] or tunes the parameters of a chosen heuristic [45, 46]. In particular, [46] tunes

parameters for each buffer, while [45] tunes the α, β and τ_{PF} parameters of the GPF scheduler given by (1.17). This design reduces the size of the action space, simplifying learning, but may limit flexibility and optimality.

Fine-grained approaches allow more direct control over the decisions:

- SRS which needs a forward pass for each RB allocation [48, 47, 49, 50, 41, 51, 52, 53].
- GRS-P outputs proportion of RB for each UE [55, 56, 57, 58, 59, 60]. However, an additional operation is needed to provide an integer number of RB for each UE.
- GRS-I selects UEs for each RB with a single forward pass [61, 62, 63, 64, 54, 65]. Multiple approaches are possible: [61, 64] output values for each combination UE/RB and select the one maximizing this value. [62] has N_f outputs and rescales it in $[1, n_L]$, and uses the ceiling operator to select a UE per RB. [63] uses pointer networks to select UE for the different RB. Multiple works leverage the AB architecture, such as [54, 65, 66], to handle the large number of action. In particular, [54] leverages AB architecture in a non-orthogonal multiple access (NOMA) framework, where all the UEs share a band and can be simultaneously scheduled within it. [65] integrates graph neural network (GNN) and AB to allocate combinations of UEs for each sub-band, where a sub-band is a node of the GNN.[66] proposes a DRL method combining the AB architecture with stable matching to jointly select the MCS and UEs for different RBs.

In the other category, [67] aims to deliver packets to dedicated applications. The action is thus the selected application (it does not consider PHY layer and by extension RB). [68] does not provide clear information about its action space.

Overall, the literature reveals a trade-off between granularity and complexity: coarse-grained approach, which may be more predictable since it relies on well-known heuristics but may be suboptimal, whereas fine-grained designs offer higher adaptability at the expense of larger action spaces and longer training times.

1.4.1.2 Classification based on the state space

The state space defines the input of the DNN, specifying the features used for optimization. Similar to heuristic methods, it also determines whether the DNN is CU or CA and QU or QA. Table 1.6 summarizes the used features in the state space in recent DRL-based scheduling approaches for wireless networks, categorizing them into six main features:

- QA relative features
  - HoL delay.
  - Number of packets (NP) per buffer.
  - All packet delays (APD).
- CA relative features
  - CQI (or SNR or instantaneous rate or throughput).
  - Average Rate.
- Other features
  - Features that do not fall into the previously defined sub-categories. They may belong to CA, QA, or neither, but still differ from the other sub-categories.

It should be noted that the acronyms NP and APD are newly proposed and were first introduced in our prior work [69].

Table 1.6: Classification based on the state space.

QA features			CA features		Other
HoL	NP	APD	CQI	Average rate	Other
[55] [49] [61] [57] [44] [58] [64] [68] [46] [59] [60]	[43] [55] [67] [49] [50] [61] [62] [63] [51] [45] [58] [64] [68] [59] [53] [65]	[51] [54]	[39] [43] [47] [55] [49] [50] [48] [61] [56] [62] [41] [57] [63] [44] [51] [45] [58] [52] [64] [68] [46] [59] [54] [53] [65] [66] [60]	[47] [55] [49] [50] [48] [45] [46] [53]	[39] [43] [50] [48] [61] [56] [41] [44] [52] [68] [59] [54] [66] [60]

The HoL delay is one of the most frequently used state features, appearing in numerous works such as [55, 49, 61, 57, 44, 58, 64, 68, 46, 59, 60]. Its prevalence reflects the importance of tracking the waiting time of the oldest packet in the buffer, which is directly related to meeting delay constraints.

Similarly, the NP in the buffer is widely used [43, 55, 67, 49, 50, 61, 62, 63, 51, 45, 58, 64, 68, 59, 53, 65] as a measure of buffer occupancy and congestion level.

The APD [51, 54] appears less frequently. Even if it is possible to use all this information thanks to DNN, the HoL is preferred maybe due to its simplicity.

The CQI is the most represented PHY layer related metric, used extensively [39, 43, 47, 55, 49, 50, 48, 61, 56, 62, 41, 57, 63, 44, 51, 45, 58, 52, 64, 68, 46, 59, 54, 53, 65, 66, 60], confirming its role as a key indicator for scheduling and resource allocation.

The average rate is less frequent but still present in several works [47, 55, 49, 50, 48, 45, 46, 53], typically serving as a throughput-oriented performance measure.

Finally, the Other category encompasses additional variables such as fairness metrics, number of arriving packets, user priorities, or QoS class identifiers [39, 43, 50, 48, 61, 56, 41, 44, 52, 68, 59, 54, 66].

Overall, the table shows a strong emphasis on simple buffer-related features (HoL, NP) combined with link quality indicators (such as the CQI). This reflects a consensus in the literature that effective DRL schedulers must jointly capture both buffer dynamics and channel conditions to optimize performance.

1.4.1.3 Classification based on the reward

The reward specifies the objectives optimized by the DRL-based scheduler. These objectives may include minimizing packet loss and delay, maximizing throughput and fairness, or a combination of these objectives.

Table 1.7 presents the reward optimized by different DRL schedulers from the literature.

Table 1.7: Classification based on the reward.

Packet loss	Throughput	Fairness	Delay	Other
[39] [43] [67] [49] [64] [53] [54]	[39] [47] [55] [49] [50] [48] [41] [45] [58] [52] [64] [68] [59] [53] [65] [66]	[47] [55] [49] [48] [41] [52] [53] [65]	[39] [43] [55] [61] [57] [44] [51] [45] [58] [46] [59]	[56] [62] [63] [51] [58] [64] [68] [59] [60]

Many studies target packet loss minimization [39, 43, 67, 49, 64, 53, 54]. Throughput maximization is another widely used reward criterion [39, 47, 55, 49, 50, 48, 41, 45, 58, 52, 64, 68, 59, 53, 65, 66], reflecting the goal of efficient resource utilization.

To ensure equitable resource distribution, several approaches incorporate fairness metrics in their rewards [47, 55, 49, 48, 41, 52, 53, 65]. Furthermore, delay-sensitive rewards are integrated in delay-critical applications to meet stringent latency requirements [39, 43, 55, 61, 57, 44, 51, 45, 58, 46, 59].

Some works extend reward formulation to capture different objectives, the number of remaining packets in the buffers [62].

Considering multiple objectives (i.e., multiple criteria in the reward function) is common in scheduling problems. However, optimizing these objectives requires selecting appropriate weights for each criterion, which requires careful weight tuning. To the best of our knowledge, [53] is the only work proposing a multi-objective solution that can adaptively handle different weightings for the various objectives.

1.4.1.4 Classification based on the performance metrics during the inference phase

DRL-schedulers are trained to maximize their expected discounted return (EDR) (Section A.2.5). However, this does not necessarily guarantee good performance during inference: 1) with respect to the objectives of the reward function, 2) with respect to other relevant telecommunication metrics. It is therefore essential to evaluate performance against these metrics during the inference phase.

Table 1.8 provides the evaluated performance metric of different DRL schedulers from the literature.

Table 1.8: Classification based on the performance during the inference phase.

Training reward	Packet loss	Throughput	Fairness	Delay	Other
[43] [49] [41] [57] [44] [51] [45] [58] [52] [64] [68] [46] [65]	[43] [55] [67] [49] [57] [44] [45] [58] [64] [53]	[47] [55] [49] [50] [48] [61] [62] [41] [63] [45] [58] [52] [64] [46] [59] [53] [66] [60]	[47] [55] [49] [48] [56] [62] [41] [63] [52] [53]	[43] [55] [61] [62] [57] [44] [64] [68] [46] [59]	[39] [61] [56] [44] [64] [68] [59] [54] [66] [60]

Most works evaluate their methods using the throughput, reflecting spectral efficiency in wireless networks. Then, the reward is the second most studied performance, reflecting the training of the DNN and if the reward aligns with other performance goals. The packet loss, the fairness and the delay are also well-studied. The packet loss reflects the reliability of the trained scheduler. The fairness reflects how the resource allocation is balanced among users, which is critical in multi-user scenarios.

A subset of works include other performance indicators, such as energy efficiency or buffer stability, to capture broader system objectives.

Overall, this table highlights the diverse evaluation criteria adopted by existing DRL schedulers, demonstrating that SotA solutions strive to optimize multiple, sometimes conflicting, performance aspects to meet complex network demands.

1.4.2 Analysis of the scheduler properties

Based on our experience in scheduling, we argue that a good DNN architecture for a scheduler should satisfy the following properties:

- It should be able to operate with a variable number of links. Since the number of UEs (or links) may vary over time, the DNN architecture must be sufficiently flexible to accommodate such changes. We denote this property as number of links independent (NLI).
- It should be permutation equivariant (PE) with regards to the inputs since this property provides superior performance for both learning and inference phases [70].
- It should jointly consider the buffer information of all UEs (or links) when performing scheduling, in order to provide a global and thus potentially optimal solution. Indeed, it is reasonable to assume that maximizing the EDR by taking into account all the buffers yields better performance than relying on only a subset of the buffers. We denote this property as global buffer management (GBM).

Remark: The NLI property enables the architecture to handle a varying number of links, but it does not necessarily guarantee good performance. The performance depends on the training process (algorithm, hyperparameters, dataset...) and the architecture's ability to generalize, which shall be evaluated by simulations.

One important feature to classify the solutions proposed in the literature is the way the state vectors are fed at the input of the DNN architecture. Let $\mathbf{f}_\ell$ denote the $n_S \times 1$ column state vector of link ℓ where n_S is the number of the state components. We can identify two approaches:

1. The state vectors are defined as single column $n_L n_S \times 1$ vectors $[\mathbf{f}_0^T, \mathbf{f}_2^T, \ldots, \mathbf{f}_{n_L-1}^T]^T$ where T denotes the transposition operator and n_L is the number of links. We refer to this case as vector state input (VSI).
2. The state vectors are fed in series (one after the other) corresponding to gathering the vectors into the $n_S \times n_L$ matrix $[\mathbf{f}_0, \mathbf{f}_2, \ldots, \mathbf{f}_{n_L-1}]$. We refer to this case as matrix state input (MSI).

It should be noted that the acronyms NLI, GBM, VSI, and MSI along with the corresponding classifications are newly proposed and were first introduced in our prior work [69].

Table 1.9 classifies the DRL schedulers from the literature according to their properties (NLI, PE, and GBM) and the form of their input state (VSI or MSI).

Table 1.9: Properties of the DNN used in the SotA.

	NLI	PE	GBM	NLI+PE	NLI+GBM	PE+GBM	NLI+PE+GBM
VSI	[50] [68]		[39] [47] [51] [67] [49] [61] [56] [62] [41] [57] [44] [51] [58] [52] [64] [46] [54] [53] [65] [66]	[43]	None	None	None
MSI	None	None	None	[55] [59] [45]	[48] [63] [45]	None	[60]

The majority of the different works uses a VSI and are GBM. These works are by construction neither NLI nor PE. Even if [51] mimics the PE property by performing permutation of the buffers at the input of the DNN, it is not strictly speaking a PE architecture.

To get the NLI property while using a fully connected (FC) DNN, [50] proposes to first select a fixed number of links using a PF heuristic and then to build a VSI with the selected links as input to the network. Since a FC is used, the architecture is not PE and since only a subset of the links are used to perform the scheduling it is not GBM.

[68] seems to be NLI according to their simulation. However, the authors do not provide clear information about their state space and DNN architecture to determine if it is also PE and/or GBM and if the state input is really a vector or a matrix (by default, we consider that it is a vector).

[45] is not clear on how the characteristics of the UEs are used by the DNN. Depending on how it is proceeded, it is either NLI and PE, or NLI and GBM.

References [63, 48] use *pointer networks* [71] that involves internal memory (or hidden states). These solutions are NLI, by construction, and GBM thanks to the hidden state, but are not PE since the scheduling decision depends on the input order of the state vectors [72].

[43] and [59] mimic the GBM property by using statistics of the network, such as the average HoL and its standard deviation for instance, for the state space vector for [39] or in the state vector of each UE in addition of their individual features at the input of the DNN. However, it is not strictly speaking GBM. [43] is NLI even if it uses VSI since it selects an heuristics to perform the scheduling.

We propose in this Thesis to resort to the EOT architecture, which is described in Section A.4.2, since it satisfies the three previously-listed properties, thanks to its attention mechanism.

Similar to the approach we develop in the following chapters, the method proposed in [60] satisfies the NLI, PE, and GBM properties. However, their system model differs from ours. Specifically, they assume that each UE has only a single packet to transmit, whereas our model accounts for multiple types of traffic per UE, with packets arrivals, as described in Section 1.2. The key component enabling their architecture to satisfy the NLI, PE, and GBM properties is the use of Deep Sets [70]. It is worth noting that the term $\frac{1}{n_L}\mathbf{1}\mathbf{1}^T$ in [60, Equation 10] functions as a uniform attention mechanism, analogous to that described in Section A.4.2 with (A.86), as it assigns equal importance to all UEs when pooling their features. Furthermore, the term $x\Gamma$ in [60, Equation 10] can be interpreted as a learned projection, analogous to the 'value' component in the attention mechanism.

Transformers (Section A.4.2) extend this pooling by replacing the uniform weighting with a learned attention mechanism that dynamically adjusts the importance of each UE based on its current state. For example, a UE with a large number of packets nearing their deadlines may be assigned more attention than one with a single packet and a relaxed delay constraint.

1.5 Conclusion

In this chapter, we presented the general system model underlying our study, detailing both the buffer and channel configurations used across Chapter 2 to Chapter 4. We introduced a range of heuristic approaches as well as an overview of existing DRL-based methods for scheduling and resource allocation. Additionally, we demonstrated how the proposed system model aligns with the 5G framework, establishing its relevance for current and future wireless communication standards.

We categorized the various DRL-based methods based on their level of scheduling granularity. In coarse-grained approaches, the DRL agent either selects a scheduling heuristic or tunes its parameters. In contrast, fine-grained approaches involve the DRL agent directly selecting the buffers to be scheduled. We proposed to classify these fine-grained methods into three categories:

- SRS: a separate forward pass is executed for each RB, selecting one buffer at a time.
- GRS-P: a single forward pass outputs a proportion of the bandwidth allocated to each buffer.
- GRS-I: a single forward pass determines which buffer is assigned to each RB individually.

We recommended that an effective DNN architecture for scheduling must satisfy three key properties:

- NLI: the ability to handle a variable number of buffers without requiring retraining.

- PE: the output should preserve the order of inputs, meaning that permuting the input buffers results in the same permutation in the output.
- GBM: the architecture must take into account all buffers in the network when making scheduling decisions.

From the SotA analysis, we observed there is no reference using VSI and having the three properties GBM, NLI and PE, with the fine-grained approach. Except [60] which uses Deep sets, solutions based on MSI are NLI by construction but cannot achieve both PE and GBM at the same time. The solutions proposed in this thesis (Chapter 2, Chapter 3, and Chapter 4) are based on the EOT architecture, which differentiates itself from the SotA by simultaneously satisfying the three required properties and enhancing the pooling operation of Deep Sets through its attention mechanism. We also identified the AB architecture as a suitable approach to handle large action spaces and adopted it in Chapter 3 and Chapter 4.

Chapter 2

Slot-based scheduling

2.1 Introduction

This chapter addresses the problem of slot-based scheduling, which corresponds to a slot-by-slot allocation considering a single RB ($N_f = 1$).

In this chapter, a scheduling relying on an EOT architecture trained using a deep Q-Learning (DQL) algorithm is proposed. We have selected an EOT architecture that possesses the three properties mentioned in Section 1.4.2: PE, NLI, and GBM. In DRL, the DNNs take as input vectors belonging to the state space, which plays a key role in system performance and is defined by the solution designer. Most of the schedulers proposed in the literature (both heuristics and DNNs) primarily use the HoL or NP in the buffers, or both. In this Chapter we investigate two new state space models for the DC traffic:

- An extended version of the HoL, denoted extended HoL (xHoL), as the HoL value augmented by its multiplicity, i.e., the number of packets sharing the same HoL value. The number of entries in the state vector increase by one for each DC buffer compared to the conventional HoL.
- A state model that considers the WT information of all packets in the buffers, referred to as APD. This state has higher cardinality than the HoL or xHoL but is expected to improve performance.

The main contributions of this chapter are:

1. The proposal of a DRL scheduling solution that mitigates the PLR due to BO and DV using an EOT architecture.
2. The performance evaluation and comparison of three different state spaces for DC traffic, i.e. the conventional HoL, the proposed xHoL and APD.
3. The performance assessment of the proposed architecture in terms of PLR concerning packet arrival rate and the number of links.

The rest of the chapter is organized as follows. Section 2.2 describes the system model. Section 2.3 introduces the optimization problem. Section 2.4 describes the implementation of the evaluated solutions. Section 2.5 presents the methodology of evaluation. Section 2.6 presents and analyzes the numerical results. Finally, Section 2.7 offers concluding remarks.

2.2 System model

We consider a communication network depicted in Section 1.2 with n_L active links, each characterized by two kinds of traffic: DC and BE.

Let $\mathcal{C} := \{n_0^c, \cdots, n_{n_L-1}^c\}$ be the set of channel capacities for the different links, where n_ℓ^c is the

number of packets that the channel associated with link ℓ can support for an error-free transmission. We note $n^c_{\max} := \max \mathcal{C}$.

A buffer i is selected by the scheduler at each time slot. Then, the $n^t_i := \min(n^c_{\ell_i}, n_i)$ oldest packets are extracted from this buffer and transmitted through the channel. Let n^d_i be the number of packets with WT equal to D_{t_i} after the extraction. A DV occurs and these packets are discarded, thus leading to packet loss (PL). It is worth noting that $n^d_i = 0$ for BE buffers, since there is no DV for this traffic. Let $n^e_i := n^t_i + n^d_i$ be the total number of packets leaving buffer i. Then, the WT time of each remaining packet is incremented by 1, and the number of remaining empty entries in the buffer is equal to $\kappa_i = B - n_i + n^e_i$. After that, n^r_i packets arrive in buffer i following a Poisson distribution with parameter $\lambda_i \in [0, \lambda_{\max}]$. We note $\Lambda = \sum_{i=0}^{n_Q-1} \lambda_i$, the global arrival rate (GAR). There are $n^o_i = \max(0, n^r_i - \kappa_i)$ additional discarded packets due to BO. This process is repeated at each slot.

Our objective is to design a scheduler mitigating the PLR due to BO and DV.

2.3 Problem formulation

The objective to mitigate the PLR is translated into minimizing the number of lost packets due to BO and DV over an infinite horizon by modeling the scheduling problem as an MDP.

2.3.1 MDP formulation

Such as stated in Section A.2.4, an MDP models decision-making problems partly under control and partly random due to a random perturbation. The random perturbation w_k is in our case the number of packets arriving in the buffers $\mathbf{n}^r_k = \{n^r_{0,k}, \dots, n^r_{n_Q-1,k}\}$ at step k ($w_k \in \mathbb{N}^{2n_L}$).

In the following, we define tailored state and action spaces and reward that define an MDP fitting the scheduling problem.

2.3.2 State space

Let $\mathbb{S}_\ell$ be the set of all the possible states for link ℓ, which can be decomposed as $\mathbb{S}_\ell = \mathbb{S}^{\text{DC}}_\ell \times \mathbb{S}^{\text{BE}}_\ell \times \mathbb{S}^{\text{capa}}_\ell$ with $\mathbb{S}^{\text{DC}}_\ell$, $\mathbb{S}^{\text{BE}}_\ell$ and $\mathbb{S}^{\text{capa}}_\ell$ are the set of state for the DC buffer, for BE buffer and for channel capacity respectively. The number of possible states for each set is:

- $|\mathbb{S}^{\text{DC}}_\ell|$ depends on the considered representation. The details for each are given in the dedicated paragraph.
- $|\mathbb{S}^{\text{BE}}_\ell| = B + 1$, corresponding to the possible number of packets in the BE buffer,
- $|\mathbb{S}^{\text{capa}}_\ell| = n^c_{\max}$.

The set of the states of the whole system is $\mathbb{S} = \prod_{\ell=0}^{n_L-1} \mathbb{S}_\ell$.

Since the buffer characteristics are identical for all links, the set of states is the same for each link, i.e. $\mathbb{S}_0 = \mathbb{S}_1 = \cdots = \mathbb{S}_{n_L-1}$. As a consequence, the total number of states is:

$$|\mathbb{S}| = \left(|\mathbb{S}^{\text{DC}}_\ell| \times (B+1) \times n^c_{\max}\right)^{n_L}. \tag{2.1}$$

The state of the system at time k is defined by the $n_{\text{features}} \times n_L$ matrix $\mathbf{s}_k := [\mathbf{f}_0, \dots, \mathbf{f}_{n_L-1}]$ where $\mathbf{f}_\ell$ is the $n_{\text{features}} \times 1$ state vector of the ℓth link with n_{features} its number of elements (for the sake of notation clarity we drop the k index for $\mathbf{f}_\ell$). The value of n_{features} depends on the considered features to represent the state vector. This state vector can be expressed as:

$$\mathbf{f}_\ell = \left[\mathbf{f}^{\text{DC-}x}_\ell, \mathbf{f}^{\text{BE}}_\ell, \mathbf{f}^{\text{capa}}_\ell\right]^T, \tag{2.2}$$

where $\mathbf{f}_\ell^{\text{capa}} \in \mathbb{S}_\ell^{\text{capa}}$ is given by $\frac{n_\ell^c}{n_{\max}^c}$, and $\mathbf{f}_\ell^{\text{BE}} \in \mathbb{S}_\ell^{\text{BE}}$ is $\frac{n_{2\ell+1}}{B}$, where the normalization in $[0,1]$ is done to provide a better training with the DNNs. This kind of normalization is performed systematically for the other features introduced later.

For the DC traffic we consider three state space representations as introduced at the beginning of this chapter, i.e. the state-HoL, the state-xHoL and the state-APD, yielding:

$$\mathbf{f}_\ell := \left[\mathbf{f}_\ell^{\text{DC-}x}, \frac{n_{2\ell+1}}{B}, \frac{n_\ell^c}{n_{\max}^c}\right]^T, \tag{2.3}$$

where $\mathbf{f}_\ell^{\text{DC-}x}$ is the vector composed of the DC traffic features for the state space of type x. n_{features} is thus equal to the number of entries of $\mathbf{f}_\ell^{\text{DC-}x}$ plus 2. We now define the $\mathbf{f}_\ell^{\text{DC}-x}$ vectors for the three state space representations.

2.3.2.1 State-HoL (S-HoL)

This model considers the features used in the SotA, i.e. the NP and the HoL, yielding:

$$\mathbf{f}_\ell^{\text{DC-S-HoL}} := \left[\frac{d_{0,2\ell}}{D_0}, \frac{n_{2\ell}}{B}\right]. \tag{2.4}$$

In that case we have $n_{\text{features}} = 4$ and $|\mathbb{S}_\ell^{\text{DC}}| = B(D_0+1)$.

2.3.2.2 State-xHoL (S-xHoL)

This model slightly improves the S-HoL one by adding the HoL multiplicity, i.e. $n_{2\ell}^{\text{HoL}}$ the number of packets having the HoL value:

$$\mathbf{f}_\ell^{\text{DC-S-xHoL}} := \left[\frac{n_{2\ell}^{\text{HoL}}}{B}, \frac{d_{0,2\ell}}{D_0}, \frac{n_{2\ell}}{B}\right]. \tag{2.5}$$

In that case we have $n_{\text{features}} = 5$ and $|\mathbb{S}_\ell^{\text{DC}}| = B^2(D_0+1)$.

2.3.2.3 State-APD (S-APD)

In this model we consider the full DC buffer information. One possibility is to set the state vector with all the packet WT values, i.e. $[d_{0,2\ell}, \ldots, d_{B-1,2\ell}]$ of length B. Another possibility is to set the state vector with the instantaneous distribution of the packet WT values $[p_{0,2\ell}, p_{1,2\ell}, \ldots, p_{D_0+1,2\ell}]$, $p_{d,2\ell} \in [0,1]$, of length D_0+2 defined as the WT histogram normalized by B. Both solutions contain the same information, but we propose to use the second one. Indeed, it is very likely that $B \gg D_0+2$ and thus the second representation offers a smaller size for the input vectors, easing the implementation and training. Moreover, the size of the link state is independent of the buffer size. Therefore, we have:

$$\mathbf{f}_\ell^{\text{DC-S-APD}} := [p_{0,2\ell}, p_{1,2\ell}, \ldots, p_{D_0+1,2\ell}]. \tag{2.6}$$

In that case we have $n_{\text{features}} = D_0+4$, and $|\mathbb{S}_\ell^{\text{DC}}| = \binom{B+D_0+1}{D_0+1}$.

The impact of the size of the state vectors on the complexity with regard to the space state model is discussed at the end of Section 2.4.2.

2.3.3 Action space

Let $\mathbb{A} = \{0, \cdots, n_Q - 1\}$ be the action space, i.e. the set of available actions for the scheduler. We note $a_k := \pi(\mathbf{s}_k)$ the action belonging to $\mathbb{A}$ chosen by the scheduler when applying the deterministic policy π on the state $\mathbf{s}_k$. The action $a_k = i$ corresponds to selecting the buffer i at slot k triggering the transmission of n_i^t packets over the channel.

2.3.4 MDP model

The transition from a state $\mathbf{s}_k$ to the next state $\mathbf{s}_{k+1}$ depends on the number of extracted packets and the number of arriving packets in each buffer. From the state and action space definitions in Section 2.3.2 and Section 2.3.3, and following the same approach as in [57], one can prove that the considered state spaces hold the Markov property, and thus the scheduling problem is an MDP.

2.3.5 Reward

Let $\mathfrak{R}_o$ be the reward associated with BO that we define as the opposite of the number of packets discarded due to BO:

$$\mathfrak{R}_o(\mathbf{s}_k, a_k, \mathbf{n}_k^r) := -\sum_{i=0}^{n_Q-1} n_{i,k}^o, \tag{2.7}$$

and $\mathfrak{R}_d$ be the reward associated with DV that we define as the opposite of the number of packets discarded due to DV:

$$\mathfrak{R}_d(\mathbf{s}_k, a_k, \mathbf{n}_k^r) := -\sum_{i=0}^{n_Q-1} n_{i,k}^d \tag{2.8}$$

$$= -\sum_{\ell=0}^{n_L-1} n_{2\ell,k}^d, \tag{2.9}$$

since DV only occurs for DC traffic.

It is worth noticing that $n_{i,k}^o$ and $n_{i,k}^d$ are the number of lost packets due to BO and DV respectively, such as defined in Section 2.2, at step k.

In order both to penalizes the packets loss and to map the reward in $[0, 1]$, we define the reward r_k at step k as the sum of the exponential of (2.7) and (2.9), yielding:

$$r_k = \frac{1}{2}\left(e^{\omega\mathfrak{R}_d(\mathbf{s}_k, a_k, \mathbf{n}_k^r)} + e^{\omega\mathfrak{R}_o(\mathbf{s}_k, a_k, \mathbf{n}_k^r)}\right), \tag{2.10}$$

where the parameter $\omega > 0$ is an hyperparameter controlling the behavior of the exponential function.

2.4 Problem solution

2.4.1 Learning procedure

Conventional tabular techniques such as value iteration (VI) or Q-Learning cannot be used to optimally solve the MDP described in the previous section due to the high cardinality of the state space. Consequently, employing function approximation methods, such as DNN, becomes essential for deriving

π^*. To achieve this, we implement the DQL algorithm, which utilizes a neural network known as deep Q-network (DQN) to arrive at the solution.

In this section, we present both used architectures for the DQN:

- The EOT architecture which is the proposed one.
- The FC architecture which corresponds to an architecture from the SotA.

2.4.2 Encoder only transformer architecture

The MSI approach is NLI by construction but neither a FC nor a DNN architecture with hidden states (such as long-short term memory (LSTM)) allows to achieve both PE and GBM properties at the same time. This is why we propose to use an EOT architecture [73] that is able to achieve the three properties:

1. First, it is NLI since we are in the MSI case.
2. Second the EOT is GBM by construction thanks to the attention mechanism that combines all the input vectors through the scale dot product.
3. Third, we render the EOT PE by removing the positional encoding [74].

The output of this architecture for each input $\mathbf{f}_\ell$ is a 2×1 vector $\mathbf{Q}_\ell$ representing the Q-values for the DC and BE traffics of link ℓ. Then, the buffer to be scheduled is determined by an $\arg\max$ over the n_L Q-value vectors. In addition, to avoid choosing an empty buffer, we apply action masking [75] before selecting an action.

The proposed architecture is depicted in Figure 2.1 and synthesized in Algorithm 1.

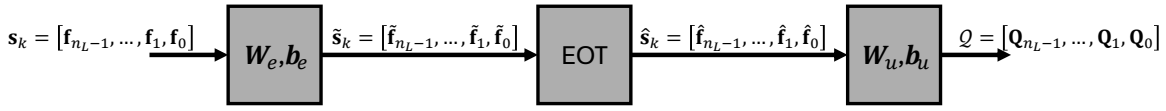


Figure 2.1: The proposed architecture with the EOT (gray boxes are learned).

The input state vectors are first multiplied by a $d_e \times n_{\text{features}}$ matrix $\boldsymbol{W}_e$ yielding as an output $\tilde{\mathbf{f}}_\ell := \boldsymbol{W}_e \mathbf{f}_\ell + \boldsymbol{b}_e$, $\forall \ell \in [0, 1, ..., n_L - 1]$, where $\boldsymbol{W}_e$ is a weight matrix of size $d_e \times n_{\text{features}}$ and $\boldsymbol{b}_e$ corresponds to a vector of size d_e for the bias, corresponding to an affine embedding that is learned during the training. Note that we assume $d_e > n_{\text{features}}$. The $d_e \times 1$ vectors $\tilde{\mathbf{f}}_\ell$ are then entered in the EOT block. The EOT is detailed in Section A.4.2. The EOT performs multi-head attention on H heads and the size for each head is d_{attn}. The FC DNN in the EOT has d_e inputs, the hidden layer has d_{mlp} neurons with a rectified linear unit (ReLU) non-linearity, and the number of outputs is equal to d_e. The $d_e \times 1$ output vectors $\hat{\mathbf{f}}_\ell$ of the EOT are multiplied by a $2 \times d_e$ matrix $\boldsymbol{W}_u$ leading to the $\mathbf{Q}_\ell$ vectors, $\mathbf{Q}_\ell := \boldsymbol{W}_u \hat{\mathbf{f}}_\ell + \boldsymbol{b}_u$, $\forall \ell \in [0, 1, ..., n_{L-1}]$, $\boldsymbol{W}_u$ is a weight matrix of size $2 \times d_e$ and $\boldsymbol{b}_u$ is the bias vector of size 2, since we have two types of traffic per link[1]. Defining $\mathcal{Q} := [\mathbf{Q}_0(0), \mathbf{Q}_0(1), \ldots, \mathbf{Q}_{n_L-1}(0), \mathbf{Q}_{n_L-1}(1)]$, the buffer i^* to the scheduled is then deduced by $i^* := \arg\max_i \mathcal{Q}$.

Note that due to the embedding $\boldsymbol{W}_e$, the size of the space vectors n_{features} has a limited impact on the global complexity. Indeed, the embedding transforms the $n_{\text{features}} \times 1$ vectors into $d_e \times 1$ ones with $d_e > n_{\text{features}}$ which is a constant regardless of the state model. Thus the complexity is mainly driven by d_e.

[1]Another methods is to consider the buffer separately, i.e. performing these operations with n_Q vectors instead of n_L. In that case, the $\boldsymbol{W}_u$ should be a $1 \times d_e$ matrix and $\boldsymbol{b}_u$ a scalar.

Algorithm 1: Forward pass of architecture depicted in Figure 2.1.

Input: $\{\mathbf{f}_\ell\}_{\ell=0}^{n_L-1}$
Output: i^*
for $\ell \in \{0, \dots, n_L - 1\}$: $\tilde{\mathbf{f}}_\ell \leftarrow \boldsymbol{W}_e \mathbf{f}_\ell + \boldsymbol{b}_e$
$\{\hat{\mathbf{f}}_0, \dots, \hat{\mathbf{f}}_{n_L-1}\} \leftarrow \mathrm{EOT}(\tilde{\mathbf{f}}_0, \dots, \tilde{\mathbf{f}}_{n_L-1})$
for $\ell \in \{0, \dots, n_L - 1\}$: $\mathbf{Q}_\ell \leftarrow \boldsymbol{W}_u \hat{\mathbf{f}}_\ell + \boldsymbol{b}_u$
$\mathcal{Q} \leftarrow \{\mathbf{Q}_\ell, \dots, \mathbf{Q}_{n_L-1}\}$
return $i^* = \arg\max_i \mathcal{Q}$

The proposed architecture complexity in floating point operations (FLOPs) is detailed in Table 2.1.

Table 2.1: Number of FLOPs of the proposed architecture.

Operation	Number of FLOPs
Embedding $\boldsymbol{W}_e$	$2 \times d_e \times n_{\text{features}} \times n_L$
EOT (from Table A.1)	$2n_L\left(d_e(4d_e + 2n_L + 2d_{\text{mlp}} + 9) + 2Hn_L\right) - n_L^2 - n_L d_e$
Projection into Q-values $\boldsymbol{W}_u$	$2 \times 2 \times d_e \times n_L$
Total	$2n_L\left[d_e(n_{\text{features}} + 4d_e + 2n_L + 2d_{\text{mlp}} + 11) + 2Hn_L\right] - n_L^2 - n_L d_e$

2.4.3 Fully connected architecture

For the sake of comparison with the SotA, in addition to the heuristics, we also implemented a VSI scheduler using a FC DNN trained using DQL. The used FC architecture is illustrated in Figure 2.2 and consists of two hidden layers with ReLU activation function and an output layer, and thus the Q-values are obtained with the following operations:

$$\mathcal{Q} = \boldsymbol{W}_3 \mathrm{ReLU}\left(\boldsymbol{W}_2 \mathrm{ReLU}\left(\boldsymbol{W}_1 \mathbf{s}_k + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right) + \boldsymbol{b}_3. \tag{2.11}$$

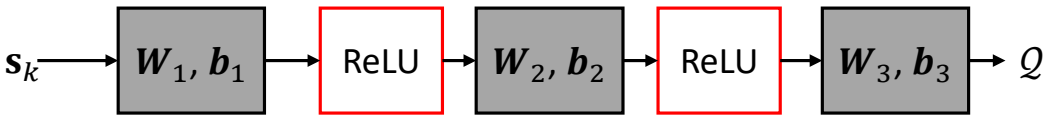


Figure 2.2: The proposed architecture with the FC (gray boxes are learned).

The trainable weight matrices $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ and $\boldsymbol{W}_3$ are of dimension $d_{\text{fc}} \times n_{\text{features}} n_L$, $d_{\text{fc}} \times d_{\text{fc}}$ and $n_Q \times d_{\text{fc}}$, respectively. The trainable biases $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ and $\boldsymbol{b}_3$ are of dimension d_{fc}, d_{fc} and n_Q, respectively. The FC architecture complexity in FLOPs is detailed in Table 2.2.

Table 2.2: Number of FLOPs of the FC architecture.

Operation	Number of FLOPs
First layer $\boldsymbol{W}_1$	$2 \times d_{\text{fc}} \times n_{\text{features}} \times n_L$
Second layer $\boldsymbol{W}_2$	$2 \times d_{\text{fc}} \times d_{\text{fc}}$
Third layer $\boldsymbol{W}_3$	$2 \times 2n_L \times d_{\text{fc}}$
ReLU activations (2 times)	$2d_{\text{fc}}$
Total	$2d_{\text{fc}}\left[n_{\text{features}}n_L + d_{\text{fc}} + n_L + 2n_L + 1\right]$

2.4.4 Adaptation of the heuristics

In this section, we present the proposed adaptations to the heuristics introduced in Section 1.3 in order to account for the slot-based context:

1. The channel capacities, expressed as the maximum number of packets that can be transmitted over the different links, are incorporated into PF, MLWDF, LOG-rule, and EXP-rule.
2. According to Sections 1.3.4, 1.3.5 and 1.3.6, we must distinguish between DC and BE traffic. Therefore, MLWDF, LOG-rule and EXP-rule should be modified accordingly.
3. The KP presented in Section 1.3.7 requires a PLR target and a bearer priority, which are not defined in this chapter. Consequently, this method also requires adaptation.

Accordingly, the achievable rate $c_{i,k}$ for buffer i in slot k, defined in Section 1.3, corresponds to n_i^c. Therefore, for all the heuristics, we set:

$$c_{i,k} = n_i^c. \tag{2.12}$$

2.4.4.1 Round-robin

The RR does not depend on the channel and thus does not require adaptation.

2.4.4.2 Proportional fair

The PF uses the instantaneous rate information for each buffer i, noted $c_{i,k}$ (for link ℓ_i). The PF expression is thus obtained using (1.13) as:

$$h_{\text{PF}}(\mathbf{x}_{i,k}) = \frac{n_i^c}{\bar{c}_{i,k-1}}, \tag{2.13}$$

where $\bar{c}_{i,k}$ is given by (1.14).

2.4.4.3 MLWDF

Using (2.12) in (1.19), and based on the discussion in Section 1.3.4, which indicates that the metric for BE buffers is equal to zero and that the PF metric must be used to handle this type of traffic, we obtain:

$$h_{\text{MLWDF}}(\mathbf{x}_{i,k}) := \begin{cases} \alpha_{i,k}\eta_i n_i^c d_{0,i,k} & \text{if } i \bmod 2 = 0 \text{ (if } i \text{ is a DC buffer)} \\ h_{\text{PF}}(\mathbf{x}_{i,k}) & \text{if } i \bmod 2 = 1 \text{ (if } i \text{ is a BE buffer)} \end{cases}, \tag{2.14}$$

where $\alpha_{i,k}$ is set as in Section 1.3.

2.4.4.4 LOG-rule

Using (2.12) in (1.20), and based on the discussion in Section 1.3.5, which indicates that the metric for BE buffers is multiplied by a factor close to zero and that the PF metric must be used to handle this type of traffic, we obtain:

$$h_{\mathrm{LOG-rule}}(\mathbf{x}_{i,k}) := \begin{cases} \alpha_{i,k} n_i^c \log\left(\beta_i + \eta_i d_{0,i,k}\right) & \text{if } i \bmod 2 = 0 \text{ (if } i \text{ is a DC buffer)} \\ h_{\mathrm{PF}}(\mathbf{x}_{i,k}) & \text{if } i \bmod 2 = 1 \text{ (if } i \text{ is a BE buffer)} \end{cases}, \tag{2.15}$$

where $\alpha_{i,k}$, β_i and η_i are set as in Section 1.3.

2.4.4.5 EXP-rule

Using (2.12) in (1.22), and based on the discussion in Section 1.3.6, which indicates that the DC buffers are handled using the EXP-rule metric defined in (1.22) and the BE buffers are handled using the PF metric defined (2.13), we obtain:

$$h_{\mathrm{EXP-rule}}(\mathbf{x}_{i,k}) := \begin{cases} \alpha_{i,k} n_i^c \exp\left(\frac{\eta_i d_{0,i,k} - \overline{\chi}_k}{1+\sqrt{\overline{\chi}_k}}\right) & \text{if } i \bmod 2 = 0 \text{ (if } i \text{ is a DC buffer)} \\ h_{\mathrm{PF}}(\mathbf{x}_{i,k}) & \text{if } i \bmod 2 = 1 \text{ (if } i \text{ is a BE buffer)} \end{cases}, \tag{2.16}$$

where $\alpha_{i,k}$, $\overline{\chi}_k$ and η_i are set as in Section 1.3.

2.4.4.6 Knapsack

The adaptation of the KP is a little bit more tricky.

Since a PLR target is not defined for the different traffic types, and because it may represent a channel metric as discussed in Section 1.3.7, we adapt (1.25) using the channel capacity, normalized by $n_{\max}^c$:

$$v_2(\mathbf{x}_{i,k}) := \frac{n_i^c}{n_{\max}^c}. \tag{2.17}$$

Second, since we do not consider bearer priority, we instead use a measure of allocation fairness, such as the KP expression in [6], and accordingly, we adapt (1.26) as follows:

$$v_3(\mathbf{x}_{i,k}) := \frac{1}{1+z_{i,k}}, \tag{2.18}$$

where $z_{i,k}$ is the number of times that buffer i is selected since the beginning of the simulation until k.

The new expression of KP adapted to our context is given by (1.23) along with (1.24), (2.17), (2.18), and (1.27), with $\alpha_j = 1$ for $j \in \{1,2,3,4\}$.

It is worth noting that for BE buffers, (1.24) is equal to 0. Because KP expression is a sum (1.23), it does not imply that $h_{\mathrm{KP}}(\mathbf{x}_{i,k}) = 0$, conversely to the MLWDF expression (1.19) when $\eta_i = 0$. Thus, there is no need to differentiate between DC and BE traffic, as is the case with MLWDF.

2.5 Performance evaluation

2.5.1 Simulation settings

2.5.1.1 Evaluated methods

We evaluate the performance of the proposed EOT, such as specified in Section 2.4.2, associated with the three types of states defined in Section 2.3.2: S-HoL, S-xHOL, S-APD that are denoted "EOT-HoL", "EOT-xHoL", and "EOT-APD", respectively in the sequel.

The FC architecture, from Section 2.4.3, is also evaluated with the three proposed state spaces, and we denote the resulting schedulers as "FC-HoL", "FC-xHoL", and "FC-APD". The input dimension is $4n_L$ for "FC-HoL", $5n_L$ for "FC-xHoL" and $n_L(D_0+2)$ for "FC-APD". It is worth noting that the FC uses a VSI whereas the EOT uses a MSI.

The DRL schedulers are then compared with the following heuristics defined in Section 2.4.4.

2.5.1.2 Training setup

We train the EOT with a fixed value of the number of links by setting $n_L = 6$, yielding thus $n_Q = 12$ buffers. Since the EOT is NLI, we further investigate its generalization capability to different values of n_L by simulations. We assume that the different buffers have identical arrival rate $\lambda = 0.15$, resulting into a GAR $\Lambda = 1.8$, defined in Section 2.2. We fix the value of $\mathcal{C} = [1, 1, 2, 2, 3, 3]$. Since the FC is not NLI, we train a dedicated FC for each value of n_L.

We set the arrival rate for each buffer as $\lambda = 1.8/(2n_L)$, as for the training of the EOT. We train the specific FCs using the configurations specified in Table 2.3. In each vector $\mathbf{n}_b^c$, the ith entry corresponds to the number of links with capacity i. These capacities are then arranged in ascending order to form the set $\mathcal{C}$, with the links at the input of the FC being sorted in the same way.

Table 2.3: $\mathbf{n}_b^c$ values used for the training.

n_L	4	6	8	10	12
$\mathbf{n}_b^c$	$[1,2,1]$	$[2,2,2]$	$[3,2,3]$	$[3,4,3]$	$[4,4,4]$

An ϵ-greedy approach is used where the exploration parameter ϵ_m, for episode m, is set to $\epsilon_{\max} = 1$ at the first episode and decayed by a factor ϵ_{decay} at each episode until reaching the value $\epsilon_{\min} = 0.01$, i.e.:

$$\epsilon_m = \max(\epsilon_{m-1}\epsilon_{\text{decay}}, \epsilon_{\min}) \tag{2.19}$$

Table 2.4 provides the other parameters used for the training.

Table 2.4: Training parameters.

	Parameters	Values
System model parameters	D_0	20
	B	40
Training parameters	γ	0.99
	Target network update period	50 steps
	Batch size	64
	ϵ_{decay}	0.99
EOT parameters	Learning rate	5×10^{-4}
	Number of heads H	4
	Number of EOT layer	1
	$d_e = d_{\text{mlp}} = H d_{\text{attn}}$	256
	Number of episodes	2000
	Number of steps per episode	7000
FC parameters	d_{fc}	512
	Learning rate	5×10^{-5}
	Number of episodes	4000
	Number of steps per episode	15 000

To optimize weight selection, validation episodes are performed every 100 training episodes using a fixed seed to ensure the same randomness. During validation, the exploration factor is set to 0 and the weights are frozen. If the sum of the rewards exceeds the previous ones, the weights are saved.

2.5.1.3 Inference setup

Three types of inference are conducted:

1. Inference with respect to arrival rates: this evaluates the generalization capabilities of the trained schedulers across different traffic regimes. A single episode of $K = 10^6$ steps is performed for each value of Λ.
2. Inference with respect to the number of links: this assesses the generalization ability of the trained EOT across varying numbers of links, as well as the performance of different FCs under their respective training configurations.
3. Inference with respect to both the number of links and channel capacities: this examines the generalization of the schedulers across both n_L and $\mathcal{C}$ for the EOT, and across $\mathcal{C}$ for the FC. To achieve this, 100 episodes of 10^6 steps are performed, totaling $K = 10^8$ steps. At the beginning of each episode, the channel capacities are randomly assigned for each number of links.

Table 2.5 summarizes the training and inference setup.

Table 2.5: Summary of the training and inference setup.

	EOT			FC		
	n_L	Λ	$\mathcal{C}$	n_L	Λ	$\mathcal{C}$
Training	6	$\Lambda = 1.8$ and $\lambda = \frac{1.8}{12} = 0.15$	$[1, 1, 2, 2, 3, 3]$	4:2:12	$\Lambda = 1.8$ and $\lambda = \frac{\Lambda}{2n_L}$	Specified in Table 2.3, sorted in increasing order
Inference wrt Λ	6	$\Lambda \in [1.5, 1.85]$	$[1, 1, 2, 2, 3, 3]$	6	$\Lambda \in [1.5, 1.85]$	$[1, 1, 2, 2, 3, 3]$
Inference wrt n_L	4:2:12	$\Lambda = 1.8$	Same as for FC training	4:2:12	$\Lambda = 1.8$	Same as for FC training
Inference wrt n_L and $\mathcal{C}$	4:2:12	$\Lambda = 1.5$	100 draws with median above 1.5	4:2:12	$\Lambda = 1.5$	100 draws with median above 1.5

Table 2.6 provides the number of FLOPs corresponding to each value of n_L for the different architectures considered. One can observe that the EOT requires over ten times more FLOPs than the FC.

Table 2.6: Number of FLOPs of the different architectures for each number of links.

n_L	4	6	8	10	12
EOT-HoL	3 192 256	4 800 688	6 417 408	8 042 416	9 675 712
EOT-xHoL	3 194 304	4 803 760	6 421 504	8 047 536	9 681 856
EOT-APD	3 233 216	4 862 128	6 499 328	8 144 816	9 798 592
FC-HoL	549 888	562 176	574 464	586 752	599 040
FC-xHoL	553 984	568 320	582 656	596 992	611 328
FC-APD	631 808	685 056	738 304	791 552	844 800

2.5.2 Performance metrics

We study the reward defined in (2.10) as well as the following metrics:

- The PLR ξ which is the number of lost packets divided by the total number of packet arriving in the buffers:

$$\xi := \frac{\sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} \left(n_{i,k}^d + n_{i,k}^o\right)}{\sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^r}. \tag{2.20}$$

  In the same way, we define the PLR for each buffer as:

$$\xi_i := \frac{\sum_{k=1}^{K} \left(n_{i,k}^d + n_{i,k}^o\right)}{\sum_{k=1}^{K} n_{i,k}^r}. \tag{2.21}$$

We assume that a PLR below 10^{-2} is acceptable, corresponding for instance to the suitable PLR for conversational voice [38].

- The throughput η which is here defined as the average number of packets sent per unit of time:

$$\eta := \frac{\sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^t}{K}. \tag{2.22}$$

Because the buffers are of limited size, when the simulation is long enough, η can be approximated by $\tilde{\eta}$, defined as the difference between the number of packets arriving in the buffers and the number of lost packets, normalized by the overall duration of the simulation i.e.:

$$\lim_{K\to\infty} \frac{\sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} \left(n_{i,k}^r - n_{i,k}^d - n_{i,k}^o\right)}{K} = \tilde{\eta}. \tag{2.23}$$

It is worth noting that when the simulation time is infinite, we have:

$$\lim_{K\to\infty} \frac{\sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^r}{K} = \Lambda. \tag{2.24}$$

From (2.23) to (2.24), we can rewrite $\tilde{\eta}$ as:

$$\tilde{\eta} = \lim_{K\to\infty} \frac{\sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^r}{K} - \frac{\sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} \left(n_{i,k}^d + n_{i,k}^o\right)}{K} = \Lambda - \delta, \tag{2.25}$$

where $\delta \geq 0$. Therefore, we deduce that:

$$\tilde{\eta} \leq \Lambda, \tag{2.26}$$

indicating that the throughput of the whole network is upper bounded by Λ. Let us also define the throughput per buffer:

$$\eta_i := \frac{\sum_{k=1}^{K} n_{i,k}^t}{K}. \tag{2.27}$$

- The fairness in terms of throughput. For a collection of metric $\mathbf{x} = [x_0, ..., x_{n_Q-1}]$ where x_i is e.g. η_i, the Jain's fairness [76] is defined as:

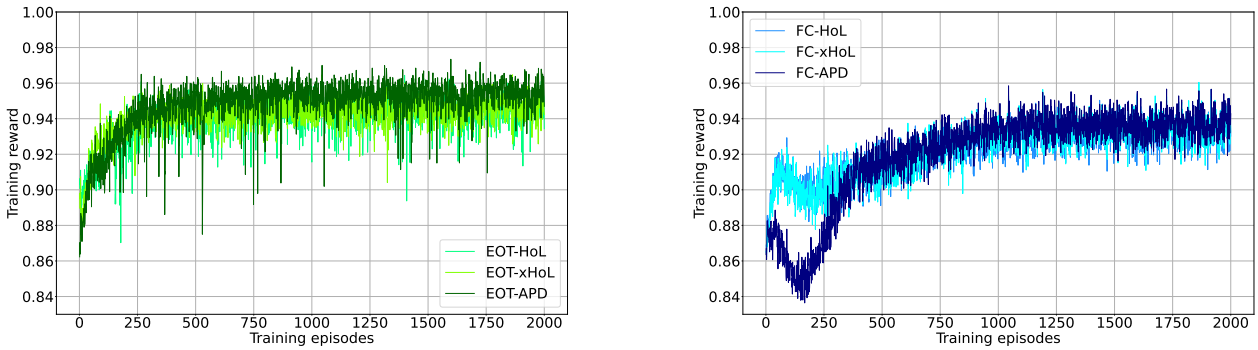
$$F := \frac{\left(\sum_{i=0}^{n_Q-1} x_i\right)^2}{n_Q \sum_{i=0}^{n_Q-1} x_i^2}. \tag{2.28}$$

- The average delay $\bar{D}_i$, is defined for each buffer i as the average WT of the packets at the moment they are sent. We define in the same way the average delay for a traffic $\bar{D}_{\mathrm{DC}}$ and $\bar{D}_{\mathrm{BE}}$ the average delay for respectively DC traffic and BE traffic.

2.6 Performance analysis

2.6.1 Training analysis

Figure 2.3 represents the average training reward per episode for $n_L = 6$ for the different studied methods.



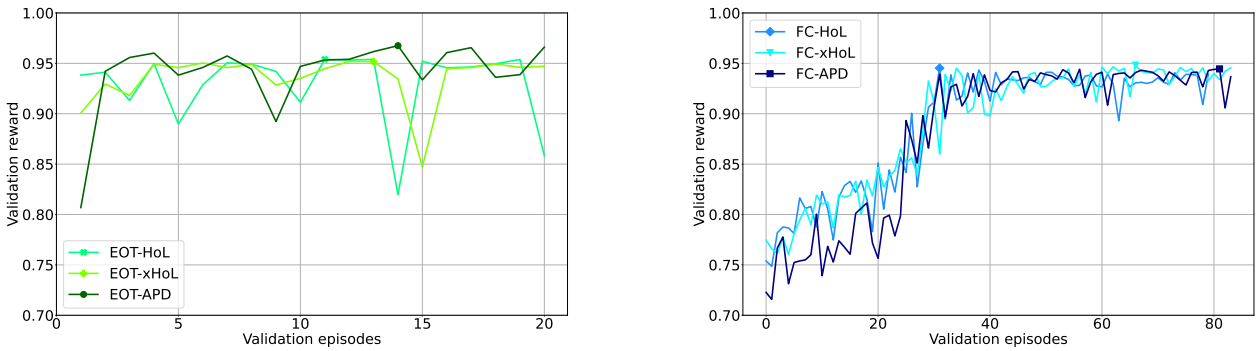
(a) EOT. (b) FC.[2]

Figure 2.3: Training reward of the different architectures for $n_L = 6$.

One can observe that:

1. EOTs and FCs architectures have converged.
2. The EOT-based methods converge faster than the FC based methods. Moreover, the training rewards once convergence is reached for the EOT are better than the training reward of the FC, with EOT converging around 0.96 whereas FC converges to about 0.94.
3. For both EOT and FC, the APD state space offers a slightly better reward compared to the two other state spaces.

Figure 2.4 plots the validation reward for $n_L = 6$ [3]. The markers indicate the highest validation reward and, consequently, the weights applied during inference.



(a) EOT. (b) FC.

Figure 2.4: Validation reward of the different architectures for $n_L = 6$.

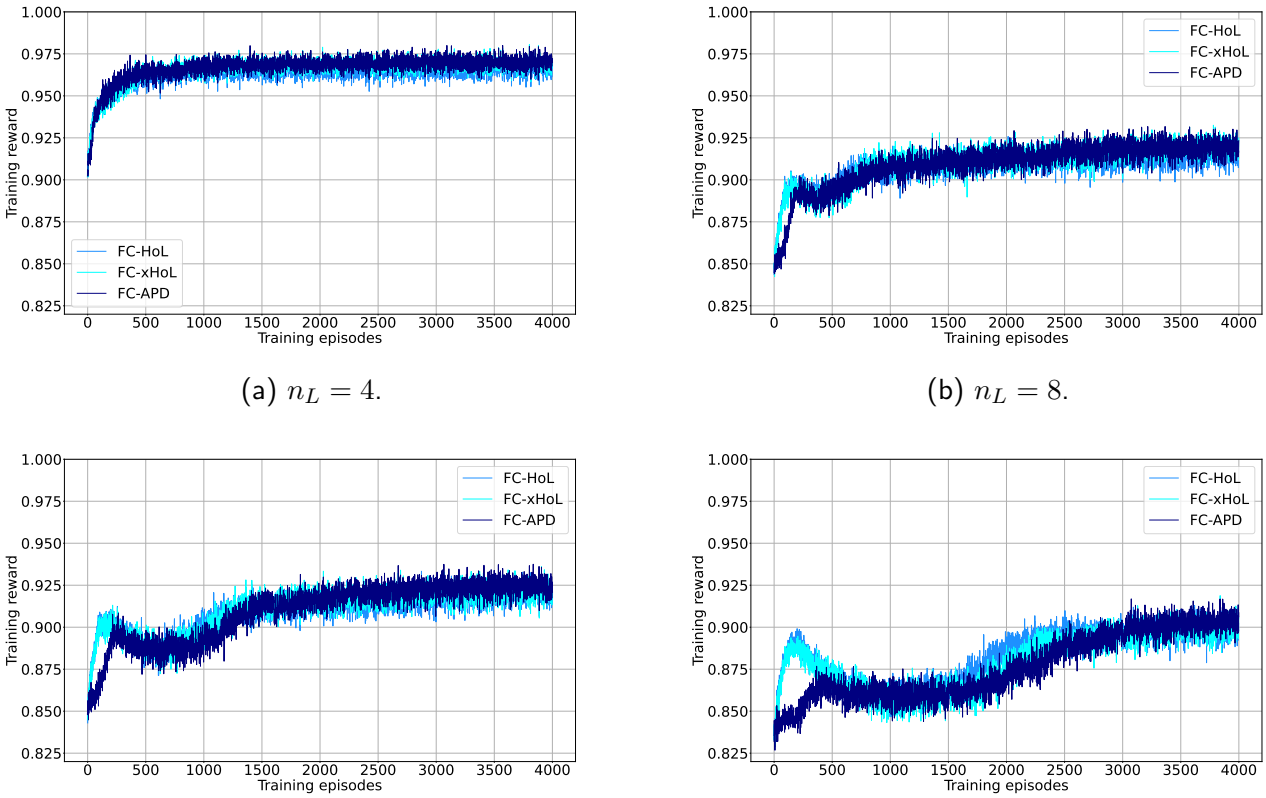
It can be noted that the EOTs converge more rapidly than the FCs. Additionally, the EOTs outperform all FCs, as there exists at least one episode for each EOT in which the validation reward exceeds 0.95, which is superior of any FC reward.

[2]The FCs are trained during 4000 episodes, such as specified in Table 2.4, but for the sake of comparison, we decide to present the first 2000 episodes of the training.

[3]For the EOT, validation episodes are conducted every 100 training episodes. In contrast, for the FC, validation episodes are also performed whenever the current training reward surpasses the best obtained so far. Additionally,we perform more training episodes for the FC than for the EOT, as it requires more time to converge.

Regarding the EOT, which is trained with a specific value of n_L, it is of interest to explore whether it can be generalized to different values of n_L. In addition, we compare the proposed EOTs with different FC trained for specific number of links.

For the sake of comparison with FC that is not NLI, we train a dedicated FC for each value of n_L. Figure 2.5 provides the training reward of the FC for $n_L \in \{4, 8, 10, 12\}$.



(a) $n_L = 4$.

(b) $n_L = 8$.

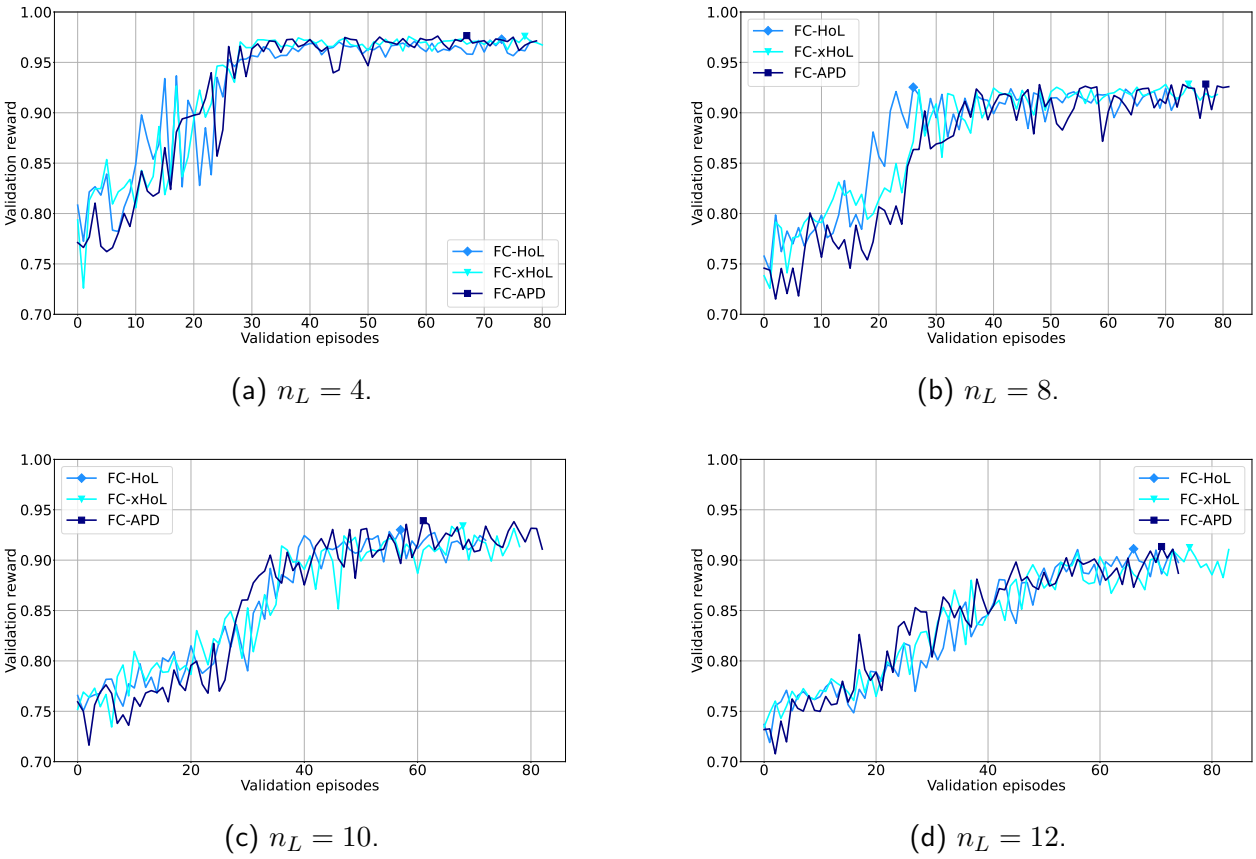
(c) $n_L = 10$.

(d) $n_L = 12$.

Figure 2.5: FC training rewards.

One can see that all the trained FC have converged. As the number of links increases, the architecture exhibits a slower rate of convergence, regardless of the state space employed.

Figure 2.6 plots the validation reward for the FCs for $n_L \in \{4, 8, 10, 12\}$.



(a) $n_L = 4$.

(b) $n_L = 8$.

(c) $n_L = 10$.

(d) $n_L = 12$.

Figure 2.6: FC validation rewards.

As for the training, as the number of links increases, the FC exhibits a slower rate of convergence, regardless of the state space employed. Furthermore, the maximum reward value diminishes as the number of links increases. The markers indicate the highest validation reward and, consequently, the weights applied during inference.

2.6.2 Inference analysis

2.6.2.1 Tested inferences

In this section, we study the generalization capabilities of the different solutions in terms of 1) arrival rates and 2) number of links and channel capacities. Furthermore, a high reward does not necessarily indicate that the other metrics are performing well, indeed. Thus, we compare the performance of the schedulers in terms of PLR, fairness, throughput and delay.

The rest of this section is organized as follows:

- Section 2.6.2.2 studies the generalization with respect to (wrt) the GAR Λ.
- Section 2.6.2.3 emphasizes the performance associated with each link and traffic for $\Lambda = 1.6$, when the DRL methods achieve a PLR below 10^{-2}.
- Section 2.6.2.4 examines the generalization performance of the EOTs wrt n_L. The channel and arrival rates are set as for the training of the FC.
- Section 2.6.2.5 examines the generalization performance of the EOTs wrt n_L and $\mathcal{C}$. The channels are drawn at the beginning of each inference episode, for each value of n_L.

Table 2.7 summarizes the realized inferences.

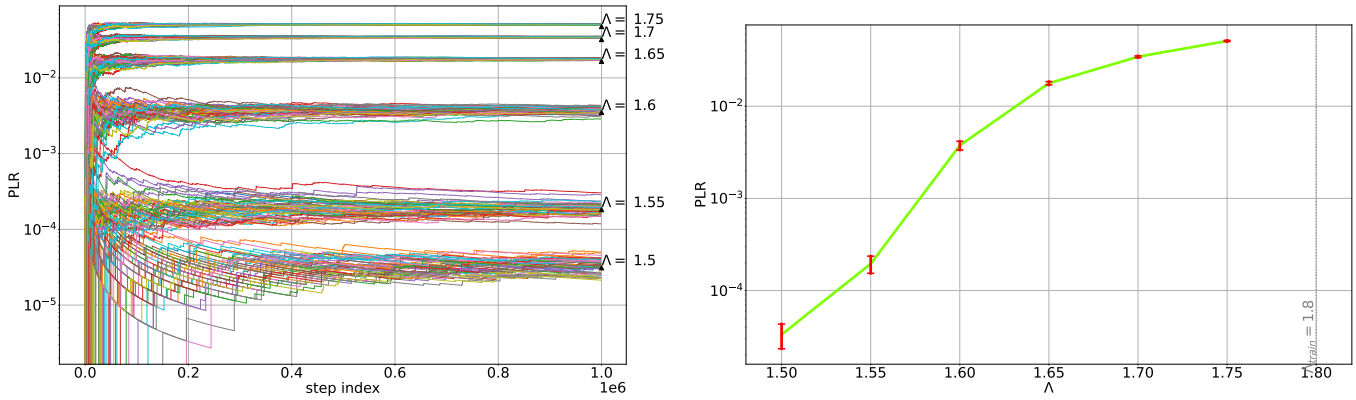
Table 2.7: Summary of the inferences.

Section 2.6.2.2	PLR in function of Λ
	PLR due to DV in function of Λ
	PLR due to BO in function of Λ
	Throughput in function of Λ
	Fairness in function of Λ
Section 2.6.2.3	PLR due to DV for each link
	PLR due to BO for each link
	Average delay for DC and BE buffers
Section 2.6.2.4	PLR in function of n_L
	PLR due to DV in function of n_L
	PLR due to BO in function of n_L
	Throughput in function of n_L
	Fairness in function of n_L
Section 2.6.2.5	PLR in function of n_L and $\mathcal{C}$
	PLR due to DV in function of n_L and $\mathcal{C}$
	PLR due to BO in function of n_L and $\mathcal{C}$
	Throughput in function of n_L and $\mathcal{C}$
	Fairness in function of n_L and $\mathcal{C}$

2.6.2.2 Generalization wrt Λ

In this section, we conduct inference to assess the generalization capabilities of the proposed schedulers across Λ and with $\mathcal{C} = [1, 1, 2, 2, 3, 3]$. The inference is done over one episode of 10^6 steps for each value of Λ. The metrics being evaluated include PLR, throughput, and fairness.

Before analyzing the inference results, we first verify if a single inference (i.e., one seed) yields reliable outcomes. Figure 2.7 presents the inference results obtained using 50 different seeds for the EOT-xHoL. Specifically, Figure 2.7a illustrates the evolution of the PLR over the course of the steps. It shows that after 10^6 steps, the PLR stabilizes around an average value, in contrast to the beginning where it is widely dispersed, highlighting the importance of performing long training episodes. Figure 2.7b displays the final average PLR (green) along with three times the standard deviation (represented by red vertical bars around the mean), providing an estimate of the range within which 99% of the results are expected to fall. The standard deviation tends to increase when Λ decreases. This shows that for high values of Λ, a single inference gives a good estimate of the average PLR, while for lower Λ, the result can vary more, making it less reliable.



(a) Evolution of the PLR.

(b) Mean and standard deviation.

Figure 2.7: Evaluation of the PLR with 50 seeds for the EOT-xHoL.

Figs. 2.8 depicts the PLR achieved for the different schedulers versus the GAR Λ when $n_L = 6$. The channels' capacities are those used during the training, i.e., the values of $\mathcal{C}$ used for Figure 2.7 and provided in Table 2.3. The aim here is to test the generalization capability of the simulated schedulers across Λ, given that they were trained for $\Lambda = 1.8$.

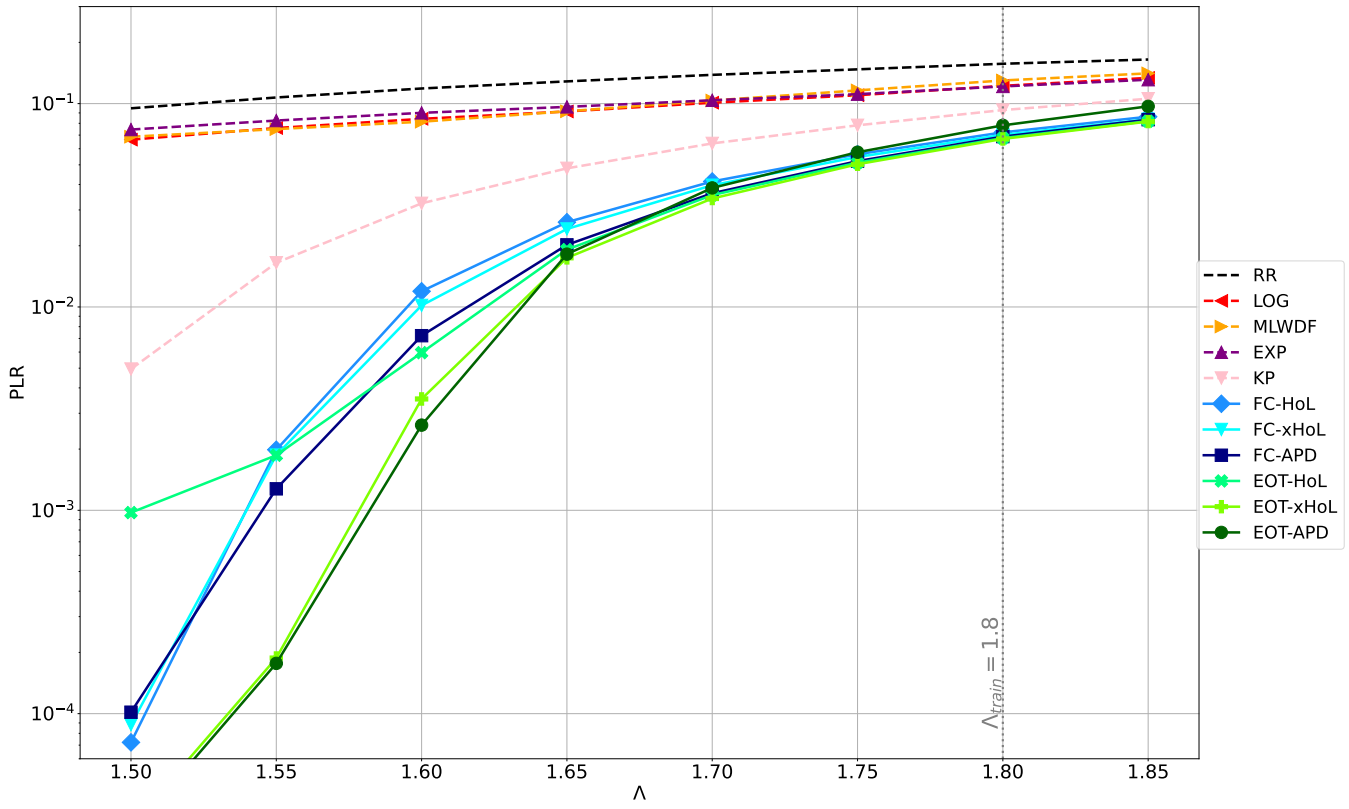


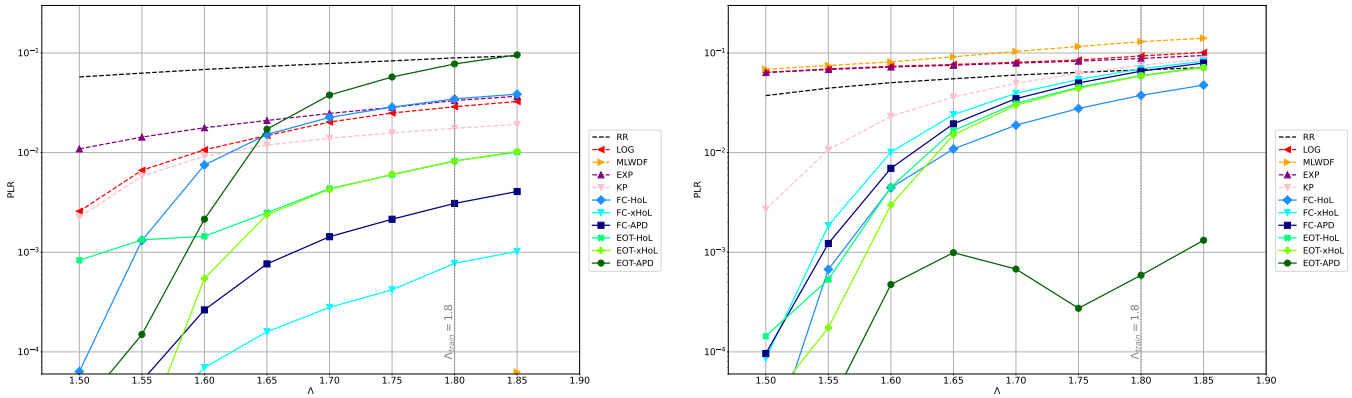
Figure 2.8: Evaluation of the generalization capability in terms of PLR over Λ, for a training with $\Lambda = 1.8$.

One can observe that:

1. The different heuristics have a PLR close to 10^{-1}, except the KP scheduler, defined in Section 1.3, that achieves better performance.

2. The DRL-based methods outperform the heuristic schedulers in terms of PLR.
3. The performance of EOT-APD and EOT-xHoL are very close regardless of Λ.
4. For $\Lambda \in [1.5, 1.65]$, the EOT-xHoL and EOT-APD schedulers provide better performance than their FC counterparts. For instance at $\Lambda = 1.55$, the PLR of EOT-xHoL is about 2×10^{-4} whereas the PLR of FC-APD (which is the best FC-based scheduler at $\Lambda = 1.55$) is about 10^{-3}.
5. The EOT-xHoL performs much better than the EOT-HoL when $\Lambda < 1.6$ and it is very close to the EOT-APD. This confirms the benefit of considering the multiplicity of the HoL in the state space.

Figure 2.9 depicts the PLR for both DV (Figure 2.9a) and BO (Figure 2.9b).



(a) PLR due to DV vs Λ.

(b) PLR due to BO vs. Λ.

Figure 2.9: Evaluation of the generalization capability in terms of PLR due to DV and BO over Λ, for a training with $\Lambda = 1.8$.

One can observe that:

1. The MLWDF scheduler provides the lowest PLR due to DV among the simulated methods. Then, the FC-xHoL and the FC-APD are the schedulers that lose the fewest packets due to DV, followed by the EOT-HoL and the EOT-xHoL for $\Lambda \geq 1.65$. All of these schedulers obtain a PLR due to DV less than 10^{-2}. However, the FC-HoL has a high PLR due to DV, at the same level as for the LOG-rule, for $\Lambda \geq 1.65$ whereas the EOT-APD is slightly above, with a PLR between 10^{-2} and 10^{-1}.
2. Concerning the PLR due to BO, the EXP-rule, the LOG-rule and the MLWDF are the worst scheduler with a PLR close to 10^{-1}. They are followed by the RR. Among the heuristics, only the KP has a PLR below 10^{-2} for $\Lambda \leq 1.5$. Concerning the DRL-based schedulers, all of them have a PLR below 10^{-2} for $\Lambda \leq 1.6$ (so below the PLR of the heuristics), then above 10^{-2} for higher values of Λ, except for the EOT-APD.

Figure 2.10 depicts the throughput wrt Λ for the different methods.

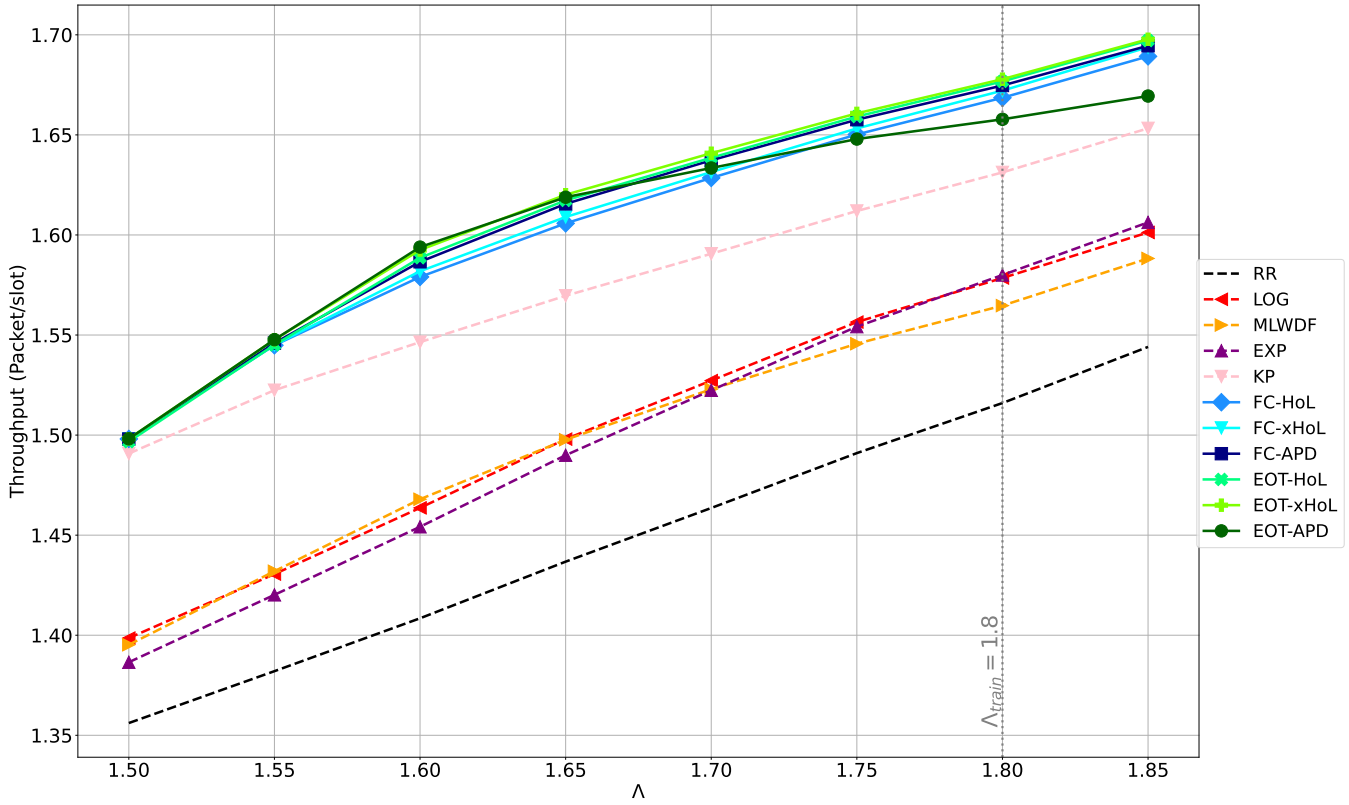


Figure 2.10: Evaluation of the throughput wrt Λ for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ for a training with $\Lambda = 1.8$.

One can see that:

1. The DRL methods outperform the heuristics. This aligns with previous observations where the DRL-based methods demonstrated a lower PLR compared to the heuristics. According to the throughput approximation in (2.25), a reduction in PL, and consequently in PLR, leads to an increase in throughput.
2. The EOT-based methods are slightly better than the FC-based methods. This still aligns with the previous observations.

Figure 2.11 depicts the throughput fairness wrt Λ.

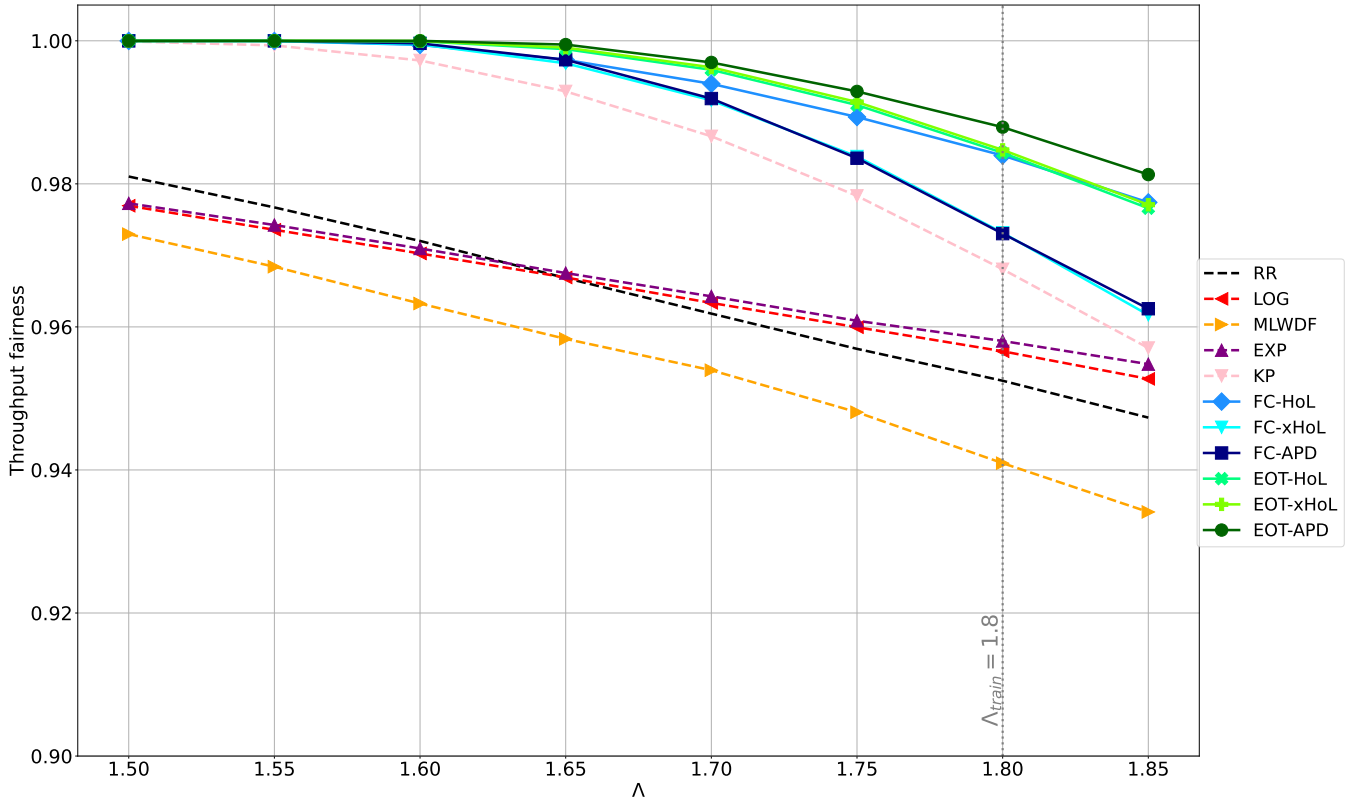


Figure 2.11: Evaluation of the fairness wrt Λ for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$, for a training with $\Lambda = 1.8$.

One can see that:

1. The DRL-based methods outperform the heuristics.
2. The EOT-based methods outperform the FC-based ones.
3. The MLWDF is the worst one. This is attributable to the fact that it effectively serves the DC buffers frequently, resulting in minimal loss associated with DV. However, it does not adequately address the BE buffers, which leads to a significant amount of BO.
4. The fairness decreases with increasing Λ due to the difficulty to obtain the same throughput on all the buffers, because the PL is in majority produced by buffers with a bad channel condition.

Conclusion of Section 2.6.2.2. We observed in this section that EOT shows strong generalization capabilities wrt to Λ, providing better performance in terms of PLR, throughput and fairness as compared with the heuristics. Among the various DRL-based schedulers, the EOT-based ones demonstrate the best performance, highlighting the significance of the DNN architecture in the scheduling process. Since the metrics are averaged over all the buffers, i.e. including the different traffic and different links in one single metric, they provide a global view of the system behavior. However, these global metrics may mask some failures of the schedulers for certain buffers and links, and thus we study in the next section the per link and per traffic performance.

2.6.2.3 Performance per link or traffic for $\Lambda = 1.6$

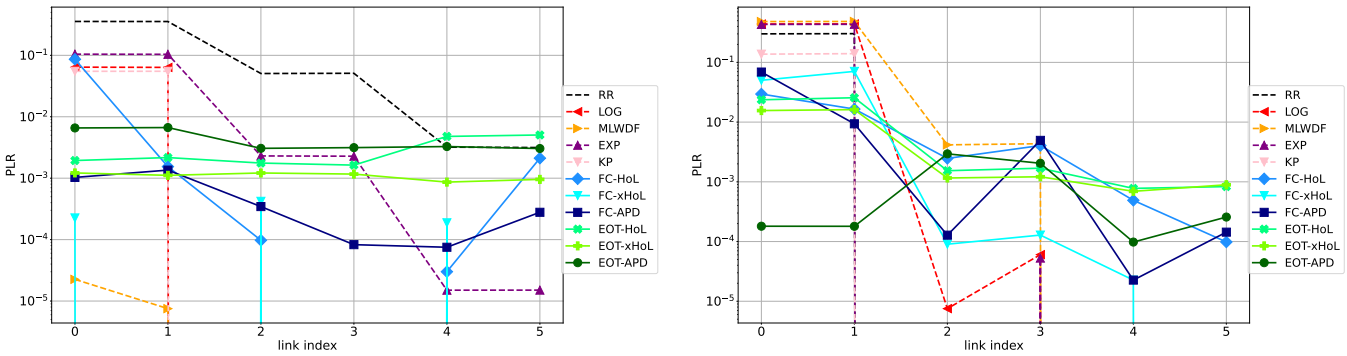
We assess in this section the inference performance of individual links with $\Lambda = 1.6$ and $\mathcal{C} = [1, 1, 2, 2, 3, 3]$. We analyze the PLR for the various links and the average delay for both the different links and the various types of traffic.

Let us examine the PLR and the delay associated with the different buffers and traffic types, with a specific focus on fixing the value of $\Lambda = 1.6$. This value represents the last instance at which DRL schedulers achieved a PLR below 10^{-2}, as illustrated in Figure 2.8.

In Figure 2.12, the PLR for the different links are plotted:

1. The PLR due to DC in Figure. 2.12a. It is worth noting that only DC buffers can have this kind of loss.
2. The PLR due to BO in Figure. 2.12b. Looking in details each buffer PL (not shown in the figure), we remarked that the PL due to BO only occurs for some BE buffers. For example, for Link 4 and Link 5, there is no PL due to BO for heuristics (absence of markers).

The links are sorted based on their channel capacity.



(a) PLR due to DV.

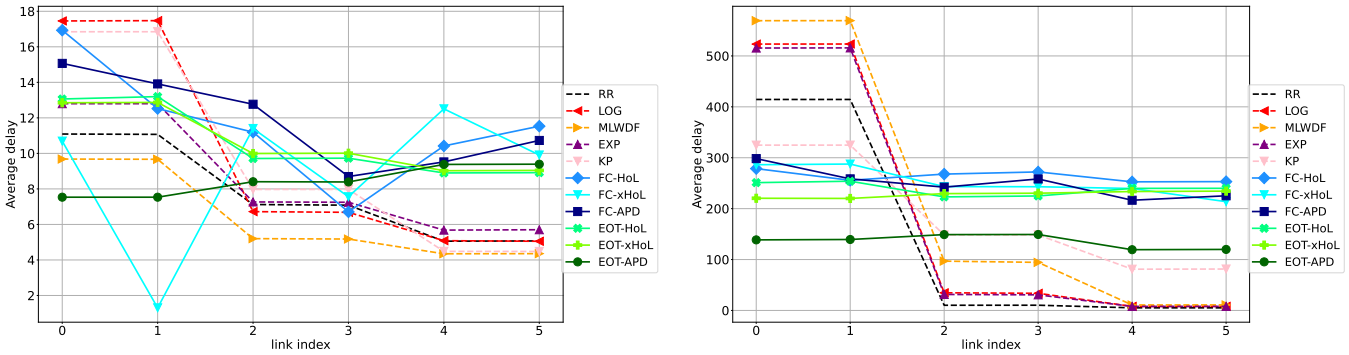
(b) PLR due to BO.

Figure 2.12: Evaluation of the PLR per link due to DV and BO for $\Lambda = 1.6$, $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and for a training with $\Lambda = 1.8$.

One can observe that:

1. For two different links with the same channel capacity, the PLR is roughly the same when a heuristic or an EOT-based scheduler is used, which is not the case when a FC-based scheduler is used. This could be attributed to the PE property of both the heuristics and EOT.
2. The PLR is a decreasing function of the channel capacity for all the heuristics, while it is not systematically the case for the DRL scheduler, for instance the EOT-HoL loses more packets due to DV for links with a channel capacity equal to 3 than for links with a channel capacity equal to 1.
3. The PLR due to DV for each link is below 10^{-2} for the DRL based scheduler (except for the first link with the FC-HoL), thus respecting a QoS of 10^{-2} for DC traffic.
4. The PLR for BE traffic is above 10^{-2} for links with a channel capacity equal to 1 for all the scheduler except the EOT-APD which has a PLR below 10^{-2} for all the links.

Figures 2.13a and 2.13b depict the average delay wrt the different links when $\Lambda = 1.6$.



(a) Average delay for DC buffers.

(b) Average delay for BE buffers.

Figure 2.13: Evaluation of the average delay per link for $\Lambda = 1.6$, $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and for a training with $\Lambda = 1.8$.

One can observe that:

1. For the heuristics, there is an inverse relationship between channel capacity and average delay: when the channel capacity increases, the average delay tends to decrease. This is the case for both BE and DC traffics. This may be due to that more packets can be sent for the links with a higher channel capacity, avoiding congestion and therefore large delays.
2. For DC traffic one can observe that the average delay for FC-based methods is very different even though the channel capacity and the arrival rate are the same. For instance, for the links 0 and 1, the channel capacity is equal to 1, and the FC-xHoL obtains different average: 11 for link 0 and 1.5 for link 1. This may be due to the non-PE property of the FC architecture.
3. The average delay of the DC traffic is a decreasing function of the channel capacity for EOT-HoL and EOT-xHoL but not for the EOT-APD scheduler. Therefore, the trained EOT-APD may serve more often the buffers with a bad channel capacity to avoid congestion, leading to better average delay than the buffers with a good channel capacity which can be emptied faster.
4. For BE traffic, the average delay for the heuristic-based schedulers increases significantly, particularly for buffers experiencing poor channel quality. In contrast, DRL-based methods maintain a stable average delay regardless of the channel capacity. According to Little's law, the average number of packets in a buffer, denoted by $\bar{B}$, is given by the product of the average arrival rate λ and the average delay $\bar{D}$ experienced by the packets:

$$\bar{B} = \lambda \bar{D}. \tag{2.29}$$

This relationship can be used to estimate an upper bound on the average delay that ensures minimal PL due to BO. For BE traffic, where each buffer has a maximum capacity of 40 packets and the average arrival rate is $\lambda = \frac{1.6}{12}$, Little's law gives:

$$\bar{D} = \frac{\bar{B}}{\lambda} = \frac{40}{\frac{1.6}{12}} = 300. \tag{2.30}$$

Thus, to avoid significant loss due to BO, the average delay should remain below 300 time units. This bound is constantly respected by all DRL-based scheduling methods. The DRL-based schedulers have $\bar{D}$ closer to 250. This signifies that the average number of packets in the buffers is 34, allowing these schedulers to keep a margin preventing from buffer overflow.

To enable a more detailed analysis of the packet delay (PD), we study its distribution for each method and each buffer. In order not to overload this chapter with figures, plots are provided in Appendix C.

Figs C.1 to C.11 represent the distribution of the PD and provide the PLR for each link for DC traffic, for the different methods. The red vertical bar indicates the average PD for the corresponding buffer, while the corresponding PLR value is shown on the right of each plot. Notice that the distribution support is $[0, D_0]$, with $D_0 = 20$. One can observe that:

1. For the heuristics: 1) for links with the same channel capacity, the distributions and the PLR are very close, 2) the distributions are more concentrated and left-shifted as the channel capacity increases, 3) the average values decrease as the channel capacity increases, 4) the LOG-rule and MLWDF distributions are more localized, whereas the distributions for the others are more spread.
2. For the FC-based: 1) for links with the same channel capacity, the distributions and the PLR are very different (average, shape and standard deviation), which explains the previously observed differences in PLR caused by DV and average PD, shown in Figures 2.12a and 2.13a. For instance, for the FC-HoL in Figure C.6, Link 0 exhibits a high PLR due to DV and a lot of packets are sent close to their deadline, while for Link 1, which has the same channel capacity as Link 0, exhibits a lower PLR and a distribution further from the deadline than for Link 0.
3. For the EOT-based: 1) for links with the same channel capacity, the distributions and the PLR are very close, which can be attributed to the PE property of the EOT, 2) the average values decrease as the channel capacity increases for EOT-HoL and EOT-xHoL, 3) the average values increase as the channel capacity increases for EOT-APD although the PLR tends to decrease at the same time.

Figs C.12 to C.22 represent the repartition of the PD and provide the PLR for each link for BE traffic, for the different methods. The red bar indicates the mean PD for the corresponding buffer, while the PLR is shown to the right of each plot. Notice that, conversely to the DC traffic, the distribution support is theoretically $[0, +\infty[$, however to plot the histograms, we have set up the maximum value to fit the data, i.e. $[0, 650]$. One can observe that:

1. For the heuristics: 1) for links with the same channel capacity, the distributions and the PLR are very close, 2) the distributions are more concentrated and left-shifted as the channel capacity increases, 3) the average values decrease as the channel capacity increases, 4) except for KP, for channel capacity equal to 1, the average values are quite high whereas for channel capacities equal to 2 and 3, the average values are close to each other and close to 0. Notably, for a capacity equal to 3, about 20% of packets are sent with a PD of 0. This is because higher capacity allows both newly arrived and older packets to be transmitted simultaneously.
2. For the FC-based: 1) for links with the same channel capacity, the distributions slightly differ, resulting in slight difference in average. However, although the average are quite close, one can notice large difference in terms of PLR, 2) the average delays are approximately constant regardless of the channel capacity, 3) most of the packets are sent with a delay greater or equal to 100, indicating that these schedulers tend to delay serving BE buffers until they are more heavily loaded.
3. For the EOT-based: 1) for the links with the same channel capacity, the distributions and the PLR are very close, 2) the average delays are approximately constant across all links, regardless of channel capacity, 3) EOT-HoL appears to prioritize early packet transmission when the channel capacity is equal to 1 or 2, 4) the average delay for EOT-APD is lower than the others.

Conclusion of Section 2.6.2.3. We evaluated in this section the individual link metrics in terms of PLR and average delay. The use of DRL-based schedulers (particularly EOT-based) demonstrates lower variance in PLR and average delay compared to heuristics. Heuristic methods generally perform better on links with high channel capacities than on those with low channel capacities, whereas DRL-based schedulers are more effective at achieving balanced results across links with different capacities. The

FC-based schedulers struggle to deliver constant performance across two links with the same channel capacity, unlike the EOT-based schedulers, which do not face this issue.

2.6.2.4 Performance wrt n_L

We evaluate in this section the performance of different schedulers under the training conditions of the FC-based schedulers, i.e. using the values of $\mathcal{C}$ used for the training of these schedulers. The goal of this analysis is to determine whether the EOT-based schedulers can generalize across different values of n_L. Given that these conditions match those used during the training of the FC-based scheduler, we expect it to perform well.

Figure 2.14 shows the PLR as a function of the number of links.

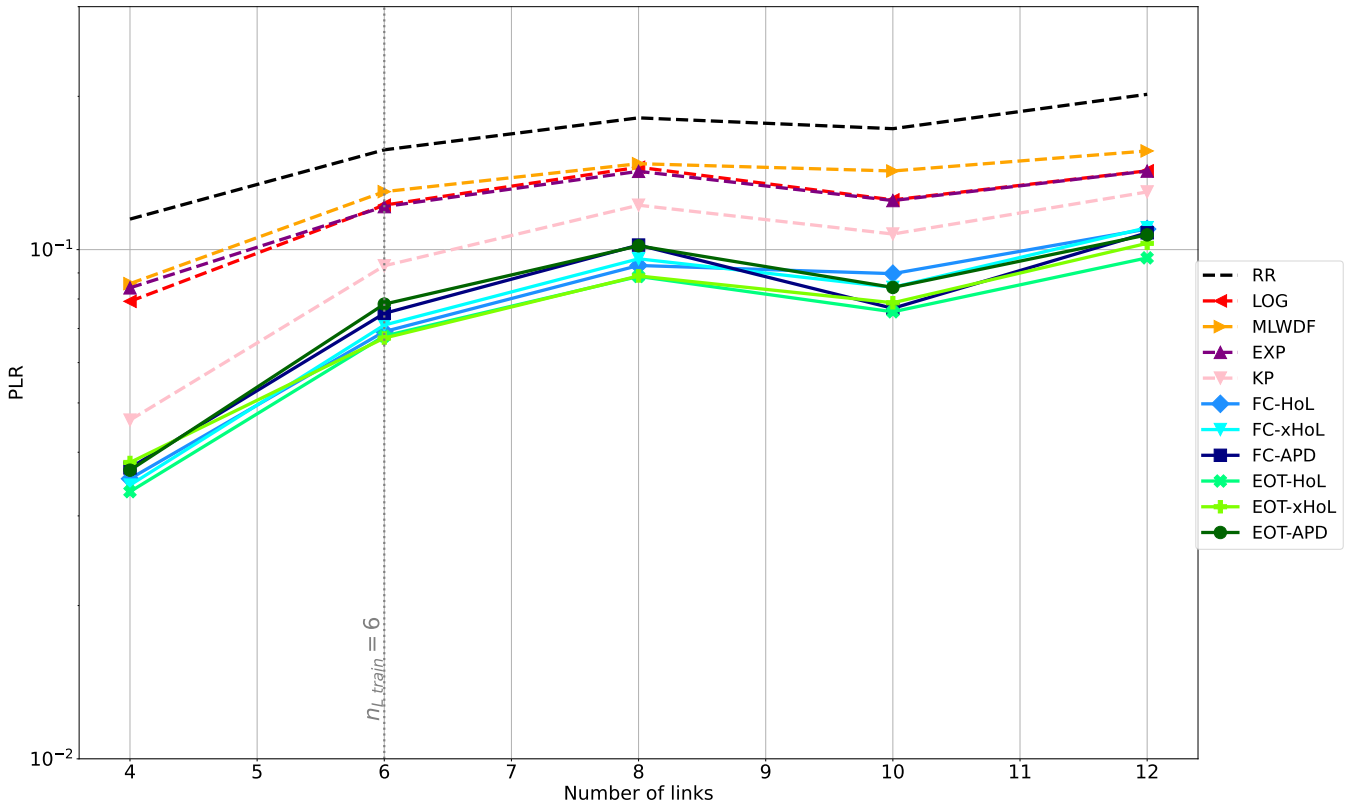
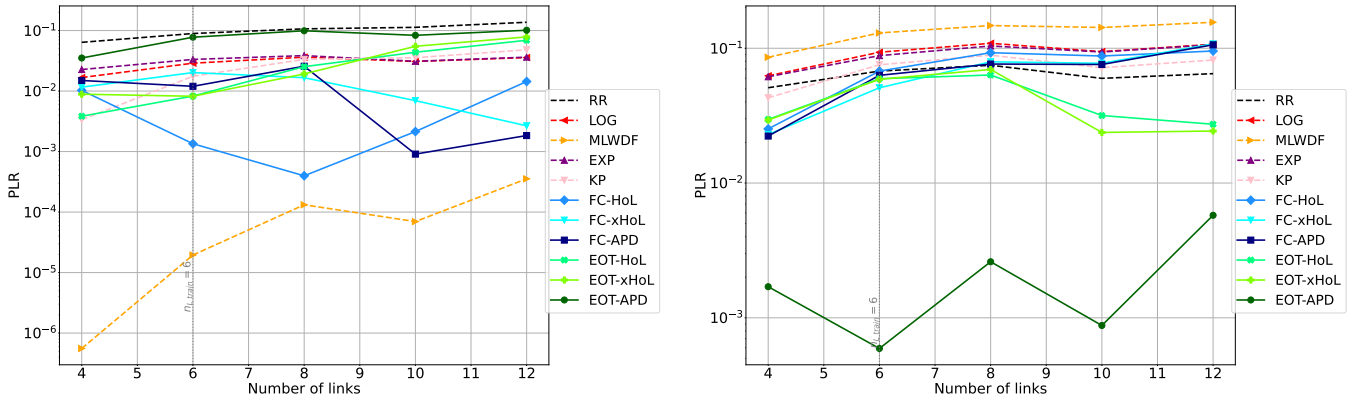


Figure 2.14: Evaluation of the PLR wrt n_L with the training channel capacities of the FC.

The following observations can be made:

- The DRL-based methods outperform the various heuristic approaches.
- The PLR for all methods remains above 3×10^{-2} across all values of n_L, indicating an overloaded regime.
- The curves for the DRL-based methods are nearly identical, suggesting that all the trained architectures result in a comparable policy in terms of total PL.
- Among the heuristics, the RR provides the worst performance, while the KP method performs the best. The MLWDF, LOG-rule, and EXP-rule methods show equivalent performance.

Figure 2.15 displays both the PLR due to DV (Figure 2.15a) and BO (Figure 2.15b).



(a) PLR DV.

(b) PLR BO.

Figure 2.15: PLR due to DV and BO wrt n_L in the training set up of the FC

The following observations can be made regarding PL due to DV:

1. PLR due to DV:
   - The MLWDF method results in the smallest PLR.
   - The RR method leads to the highest PL.
   - The KP, LOG-rule, and EXP-rule methods exhibit comparable performance.
   - The EOT-APD method loses more packets due to DV than all other methods, except for RR.
   - Other EOT-based schedulers begin to lose more packets than the FC-based scheduler when $n_L \geq 10$, with similar performance between the two methods for $n_L < 10$.
2. PLR due to BO:
   - The EOT-APD method incurs the least PL due to BO, compensating for its higher PL due to DV.
   - Other EOT-based methods outperform the rest for $n_L \geq 8$, while showing similar performance to the FC-based schedulers for lower values of n_L.
   - The FC-based schedulers outperform the heuristics for $n_L = 4$, achieving equivalent performance to them for $n_L \geq 6$.
   - Among the heuristics, MLWDF results in the highest PLR, as it favors DC buffers over BE buffers. The remaining heuristics yield comparable performance, with a PLR around 10^{-1}.

Figure 2.16 presents the throughput as a function of the number of links.

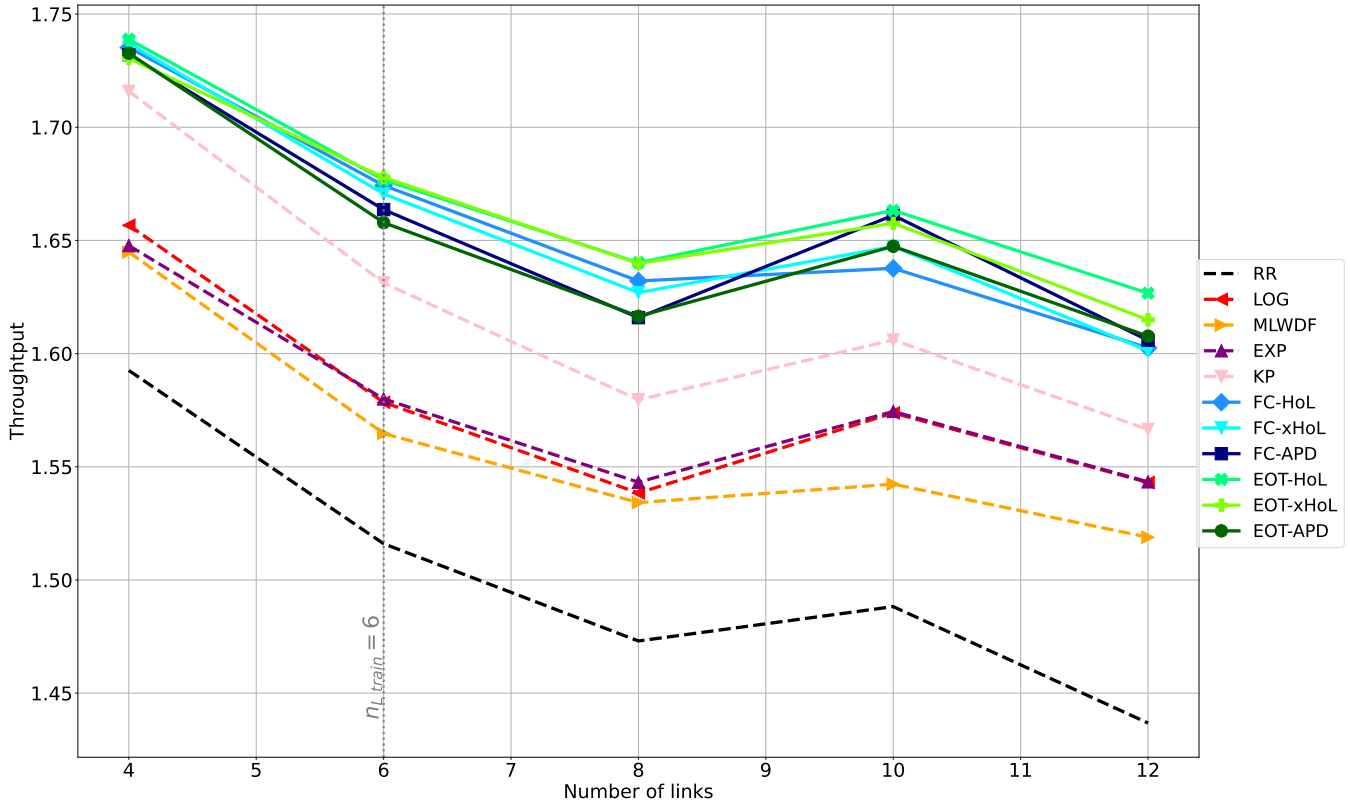


Figure 2.16: Evaluation of the throughput wrt n_L in the training set up of the FC.

The following observations can be made:

- Throughput generally decreases as the number of links increases across all methods.
- DRL-based methods achieve higher throughput compared to heuristic approaches, with no significant performance differences among the DRL methods.
- Among the heuristic methods, KP performs the best, while RR yields the lowest throughput.

Figure 2.17 illustrates throughput fairness as a function of the number of links.

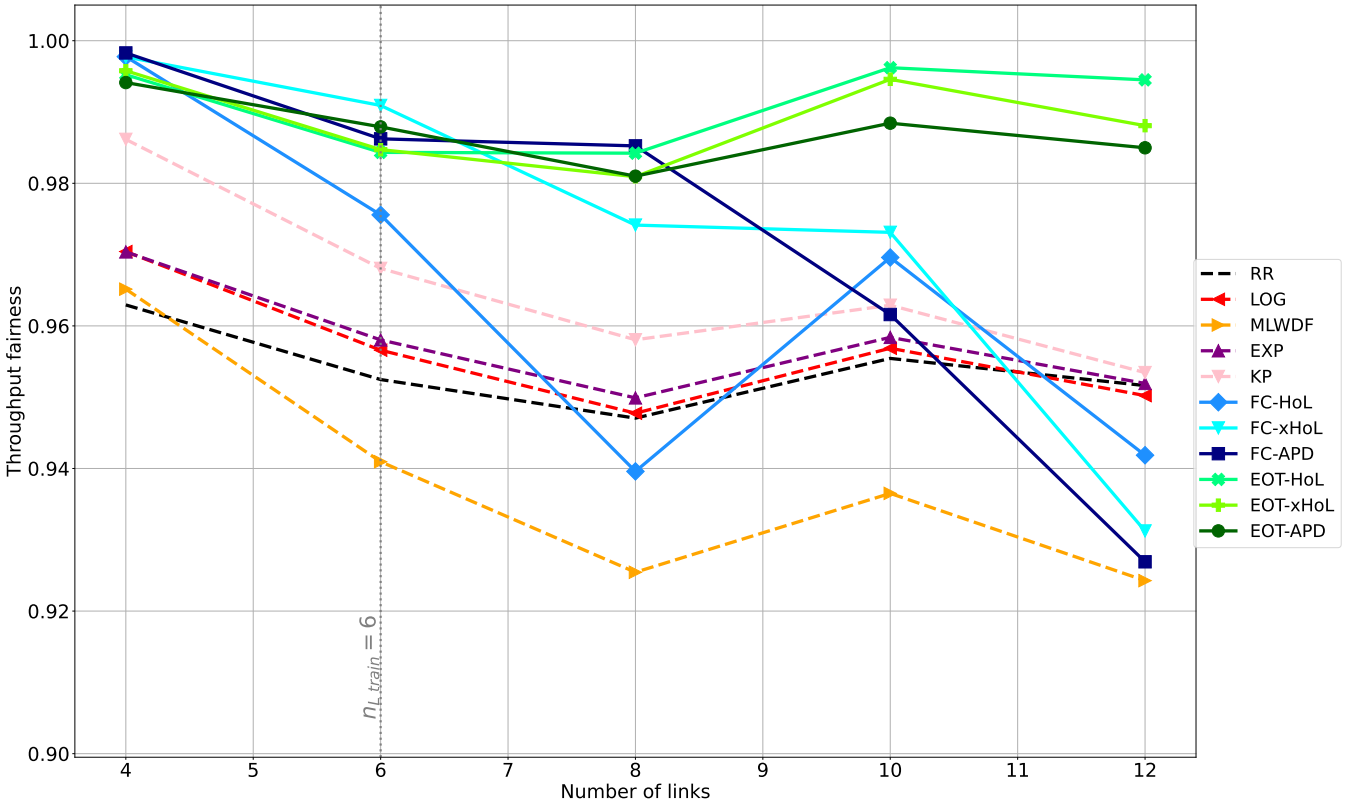


Figure 2.17: Evaluation of the fairness wrt n_L in the training set up of the FC.

The following observations can be made:

- The fairness of the EOT-based schedulers remains stable with respect to the number of links, constantly ranging between 0.98 and 1, outperforming all other methods.
- The fairness of the FC-based schedulers declines sharply as the number of links increases, ranking as the second worst when $n_L = 12$.
- Most heuristic methods, except MLWDF, converge to similar fairness levels as the number of links increases. The MLWDF may prioritize serving DC buffers while leaving BE buffers underserved.

Conclusion of Section 2.6.2.4. We compared the performance of the different schedulers under the training conditions of the FC-based schedulers, meaning identical arrival rates, channel capacities, and channel capacity ordering. The DRL methods achieve comparable performance, all outperforming the heuristics. Despite being trained with $n_L = 6$, the EOTs generalize well to other values of n_L.

To better distinguish the performance differences among the schedulers, we must conduct inference with different values of $\mathcal{C}$ for each n_L.. The results for this scenario are presented in the next section.

2.6.2.5 Generalization wrt n_L and $\mathcal{C}$

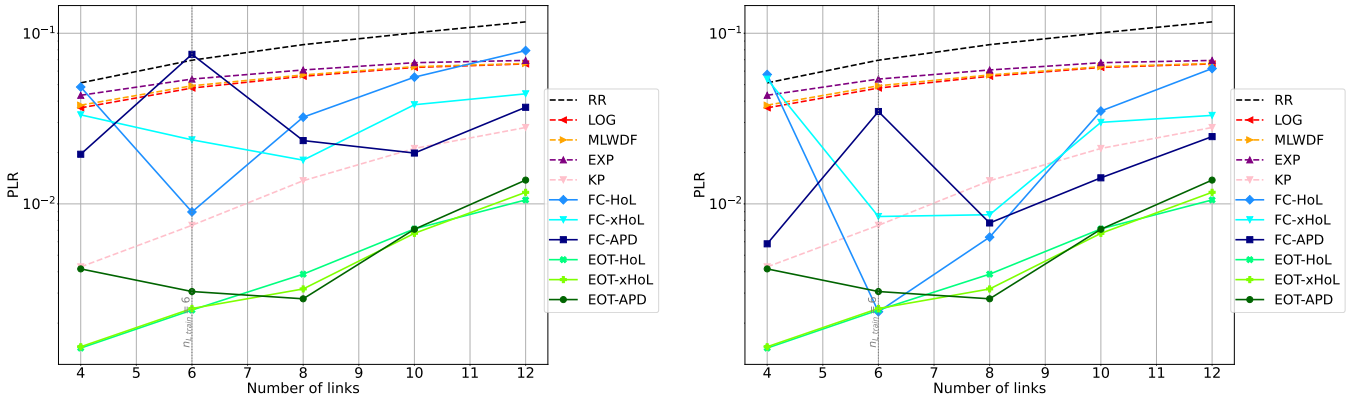
We assess the generalization capabilities of the EOT-based schedulers across both n_L and $\mathcal{C}$, while evaluating the FC-based schedulers across $\mathcal{C}$ only, as they are trained on a fixed n_L value. The analysis focuses on PLR, throughput, and fairness. All inferences are conducted with $\Lambda = 1.5$.

The channel capacities $\mathcal{C}$ are chosen such that each episode uses a distinct value. Additionally, an episode is conducted only if the median of $\mathcal{C}$ is at least 2, ensuring compliance with stability conditions

($\sum_{i=0}^{n_Q-1} \frac{\lambda_i}{n_i^c} < 1$[32]). Specifically, for $n_L = 4$, all 54 possible configurations of $\mathcal{C}$ are covered, while for other values of n_L, 100 distinct configurations are randomly sampled.

Additionally, we evaluate the FC-based schedulers both with and without sorting links by channel capacity. In the unsorted case, channel capacities are drawn at the beginning of each episode and remain unordered, meaning the states of different links appear in arbitrary order in the FC input. In contrast, with sorting, links are arranged in ascending order based on their respective channel capacities before being fed into the FC, to circumvent the absence of PE property of the FC.

Figure 2.18 presents the PLR as a function of the number of links, without channel ordering in Figure 2.18a and with channel ordering in Figure 2.18b.



(a) Without sorted link capacities.

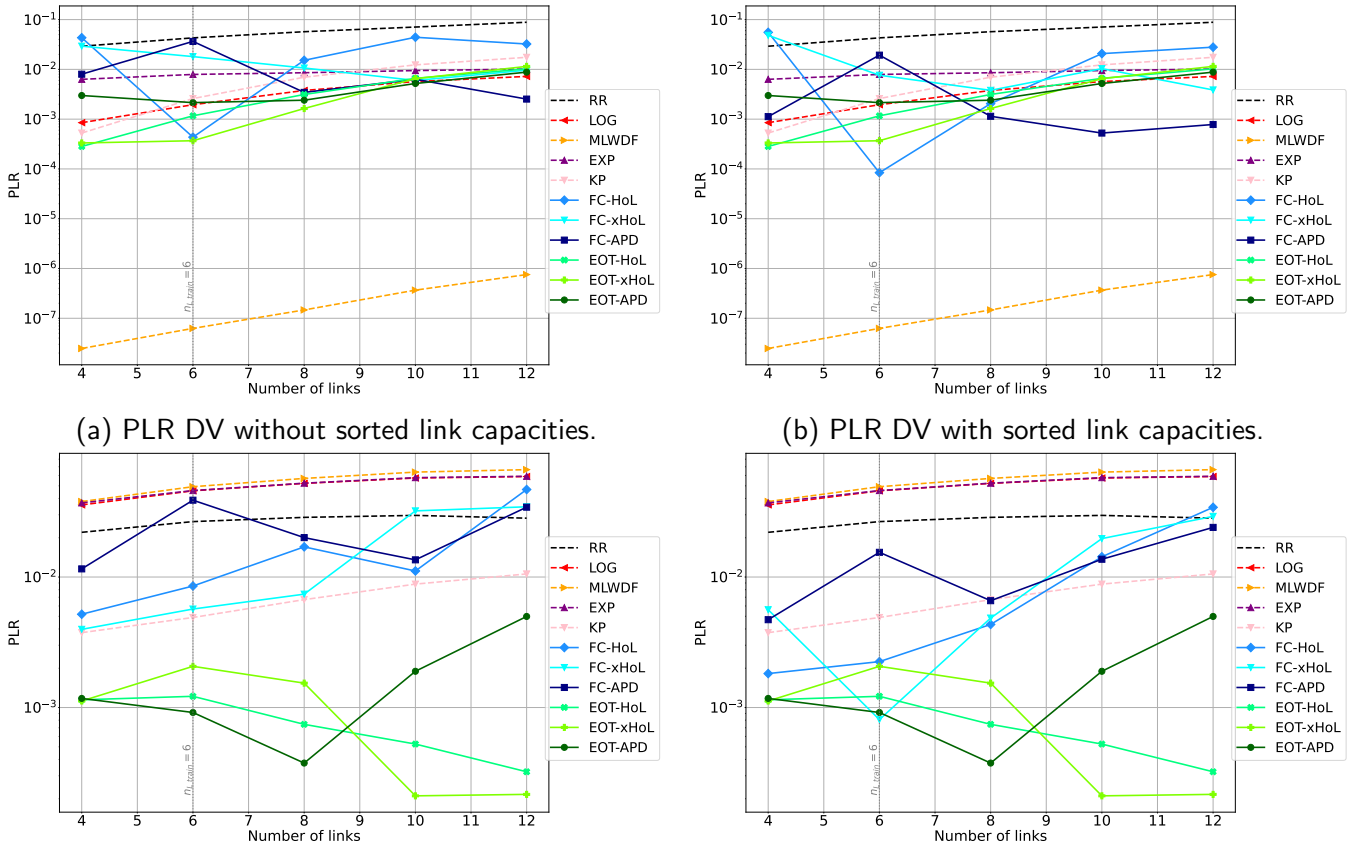
(b) With sorted link capacities.

Figure 2.18: Evaluation of the PLR wrt n_L.

The following observations can be made:

- Generally, as the number of links increases, the PLR also increases.
- The EOT-based schedulers constantly outperform all other methods. It is worth noting that the results with and without permutation are the same for EOT-schedulers since they are PE.
- Without sorting, the KP scheduler generally outperforms the FC-based schedulers. However, with channel ordering, the FC-based schedulers achieve better performance, indicating their sensitivity to the order of buffers in their VSI. Additionally, when inputs are not sorted, the performance of the FC-based schedulers is slightly better than the performance of the heuristics, except KP.

Figure 2.19 presents the PLR contributions from both DV and BO.



(a) PLR DV without sorted link capacities.

(b) PLR DV with sorted link capacities.

(c) PLR BO without sorted link capacities..

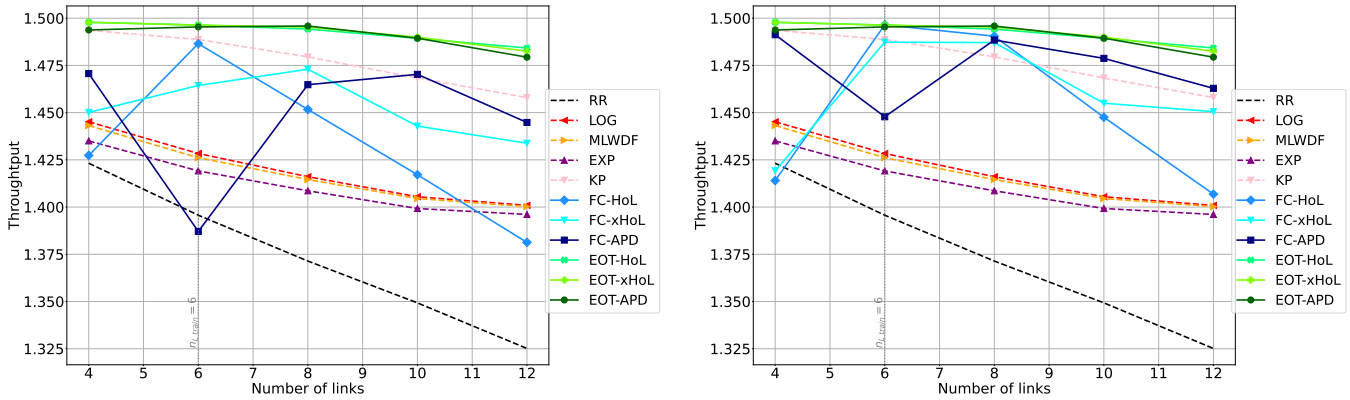
(d) PLR BO with sorted link capacities.

Figure 2.19: PLR due to DV and BO wrt n_L

The following observations can be made:

1. Regarding PLR due to DV:
   - The MLWDF outperforms all other methods.
   - The EOT-based schedulers generally surpass KP, except for EOT-APD when $n_L = 4$. Their performance approaches that of the LOG-rule as the number of links increases.
   - The FC-based schedulers exhibit erratic behavior, likely due to the use of distinct architectures for different numbers of links. Notably, their performance improves when the input is sorted in increasing order of link capacities.
2. Regarding PLR due to BO:
   - The EOT-based methods outperform all heuristic approaches, regardless of the number of links. They drop three times fewer packets than KP and achieve more than an order of magnitude improvement over other heuristics.
   - The FC-based methods benefit significantly from sorting inputs in increasing order of link capacities. In Figure 2.19c, they perform worse than KP and struggle to surpass RR. However, in Figure 2.19d, the FC-HoL achieves substantial gains over KP for $n_L \leq 8$, demonstrating improved performance in certain cases.

Figure 2.20 depicts throughput as a function of the number of links.



(a) Without sorted link capacities.

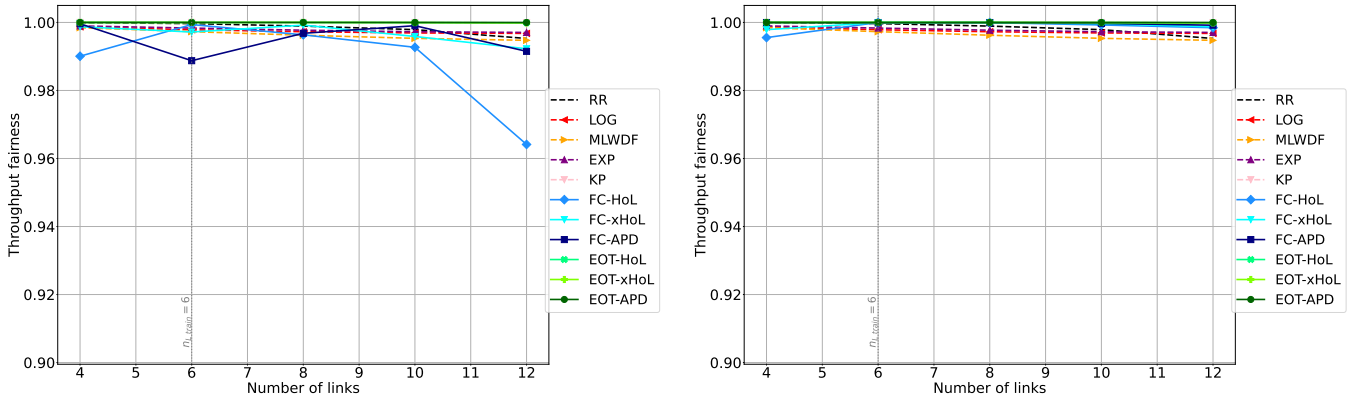
(b) With sorted link capacities.

Figure 2.20: Evaluation of the throughput wrt n_L.

The following observations can be made:

- The EOT-based schedulers outperform all other scheduling methods.
- The FC-based schedulers, when using unsorted input (i.e., link capacities not arranged in increasing order), perform worse than the KP scheduler. However, when the input is sorted, they achieve significant gains and even surpass the KP scheduler in certain cases.
- Unlike other methods, both sorted and unsorted FC-based schedulers exhibit erratic behavior. This may be attributed to the fact that each FC is trained for a specific number of links, leading to significantly different learned policies.

Figure 2.21 illustrates fairness as a function of the number of links.



(a) Without sorted link capacities.

(b) With sorted link capacities.

Figure 2.21: Evaluation of the fairness wrt n_L.

The following observations can be made:

- All methods achieve high fairness, exceeding 0.95, indicating that all buffers are served equitably, with none experiencing starvation. This is due to the moderate traffic load, which allows the schedulers to serve all buffers effectively.
- The FCs exhibit improved fairness when links are sorted compared to when they are not.

Conclusion of Section 2.6.2.5. We evaluated in this section various methods across different numbers of links and channel capacity values, with $\Lambda = 1.5$. The FC-based schedulers were assessed both with

and without sorting the links based on their channel capacities, emphasizing the significance of input order for this architecture.

Our results demonstrate that the proposed EOT-based scheduler outperforms all other methods in terms of PLR, throughput, and fairness. Notably, despite being trained on a fixed configuration (i.e., a specific number of links, channel capacities, and arrival rates), the EOT-based schedulers exhibit strong generalization capabilities, adapting effectively to varying conditions.

2.7 Conclusion

In this chapter, we addressed a slot-based scheduling problem with the objective of minimizing the PL that may come from DV and BO. We considered two types of traffic: packets with strict DC, and BE packets. The scheduling decision is performed at each slot where only one buffer can be served. The packet transmission is done on deterministic and error-free channels, with each link having different channel capacity.

As alternatives to the conventional HoL state space, we proposed the xHoL, which adds the multiplicity of the oldest packet in the buffer, and the APD, which takes into account the WT of all packets in the buffer. We proposed a DQL-based scheduler using an EOT architecture for the DNN. This architecture is NLI, meaning that it can handle variable number of links, GBM, meaning that it considers the entire buffer information, and is permutation equivariant.

Our simulations demonstrate that the proposed EOT scheduler surpasses both FC-based schedulers and heuristics from the SotA. This holds true even when the EOT is evaluated with a different number of links compared to those used in its training, alongside an FC specifically optimized for that particular number of links. This observation is promising, and suggests that the EOT could achieve even better performance if trained on varying numbers of links. Additionally, we observed that xHoL state space improves upon the standard HoL one, while getting close to the performance of the APD. In addition, the EOT-schedulers achieve better results on metrics other than the one used for training (throughput, fairness, and average delay). This suggests that these metrics are interrelated. For instance, minimizing the PL helps keep the average delay for all traffic types bounded, regardless of the channel conditions. Further investigation is required to fully understand the relationship between these two metrics.

Part of material of this chapter has been published in [69].

Chapter 3

Frame-based scheduling

3.1 Introduction

This chapter addresses the scheduling and resource allocation problem over a frame, i.e. over a set of RBs for a given slot as depicted in Section 1.2

We remind that two approaches can be used for the scheduling operation over a frame:

- RBs are assigned sequentially, one by one, which is refereed to as an approach SRS. This approach can be seen as a local optimization and one can use for instance the slot-based approach developed in Chapter 2.
- RBs are assigned all at once, which is refereed to as GRS. This approach can be seen as a global optimization.

The solution proposed in this chapter follows the GRS-I approach (Section 1.4.2), by introducing a novel DNN architecture trained with double deep Q-Learning (DDQL). The GRS approach suffers from the *curse of dimensionality*, as the number of possible actions grows exponentially with the number UEs, rendering this approach inapplicable with conventional DNN architectures. To mitigate this issue, we propose to use the AB module [77], described in Section A.4.3. The AB is composed of two different parts:

- The first part is a DNN producing a latent representation of the state.
- The second part is built of parallel branches which use the latent state representation as inputs. Each branch corresponds to a RB and allows selecting a UE for the corresponding RB.

The DNN producing a shared representation is an EOT because we have previously identified that it possesses the three properties described in Section 1.4.2: NLI, PE and GBM. The resulting architecture is called *EOT-AB*.

The EOT-AB solution is evaluated using Nokia's WS environment that is available online [6] and which implements a simplified version of the 5G downlink scheduling mechanism.

The main contributions of this chapter are:

1. The proposal of a GRS-DRL solution, based on an AB architecture, and using an EOT for the first DNN.
2. The development of a masking procedure adapted to the AB architecture.
3. The performance evaluation and comparison of the proposed architecture, along with three heuristics implemented, in the WS environment.

The rest of this chapter is organized as follows. Section 3.2 formulates the scheduling problem as an MDP. Section 3.3 describes the proposed solution. Section 3.4 presents the training and evaluation setup, in particular the WS environment and the proposed adaptations. Section 3.5 provides numerical

results and analyses them. Section 3.6 concludes the chapter.

3.2 Problem formulation

The GRS approach can be still cast into a MDP where the state space, the action space and the reward as described below.

3.2.1 State spaces

The features related to buffer i for RB j at time slot k is denoted by the vector $\mathbf{f}_{i,k}^{j}$ of dimensions $n_{\text{features}} \times 1$. Then the state $\mathbf{s}_k^j$ of all the buffers for RB j during time slot k is denoted by the $n_{\text{features}} \times n_L n_{\text{QoS}}$ matrix:

$$\mathbf{s}_k^j := [\mathbf{f}_{0,k}^j, \ldots, \mathbf{f}_{n_L n_{\text{QoS}}-1,k}^j]. \tag{3.1}$$

Since we propose an allocation for all the RBs for the GRS approach, the input state $\mathbf{s}_k$ provided to the DNN corresponds to $j = 1$, i.e.:

$$\mathbf{s}_k = \mathbf{s}_k^1. \tag{3.2}$$

3.2.2 Action space

For the GRS-I approach, the joint RB allocation is written as:

$$\boldsymbol{a}_k := \left(i_k^1, \ldots, i_k^{N_f}\right), \tag{3.3}$$

where i_k^j corresponds to the buffer to be scheduled during the jth RB of slot k. It is worth noting that the resulting action space has a dimension of $(n_L n_{\text{QoS}})^{N_f}$. For instance, if $N_f = 25$ and $n_L n_{\text{QoS}} = 32$, we obtain $(n_L n_{\text{QoS}})^{N_f} > 10^{37}$, illustrating the curse of dimensionality discussed previously, and thus justifying the need for the AB architecture to mitigate it.

3.2.3 Reward

In the GRS approach, a single reward r_k is received after all N_f RBs are allocated.

3.3 Problem solution

We first expose in what follows the proposed DNN architecture, and then detail the implementation of the masking mechanism.

3.3.1 Deep neural network architecture

The proposed architecture follows the AB architecture philosophy as described in Section A.4.3 along with the use of an EOT DNN, referred to as EOT-AB, and is illustrated in Figure 3.1. First, the state space, $\mathbf{s}_k$ undergoes a linear embedding using the $d_e \times n_{\text{features}}$ matrix $\boldsymbol{W}_e$. The result of this embedding is $\tilde{\mathbf{s}}_k := \boldsymbol{W}_e \mathbf{s}_k + \boldsymbol{b}_e \mathbf{1}_{n_L n_{\text{QoS}}}^T$, with dimensions $d_e \times n_L n_{\text{QoS}}$, where $\boldsymbol{b}_e$ is an additional bias vector of dimension $d_e \times 1$ and $\mathbf{1}_{n_L n_{\text{QoS}}}^T$ is a vector of one of dimension $n_L n_{\text{QoS}} \times 1$. Next, $\tilde{\mathbf{s}}_k$ is processed by an EOT module.

The output of the EOT is a $d_e \times n_L$ matrix, denoted by $\tilde{\boldsymbol{o}}_E$. To integrate the sequence of an EOT with the AB architecture, we propose to distribute the latent representation of the state across each branch. For the jth branch, the projection into advantages is performed by the $1 \times d_e$ vector $\boldsymbol{w}_u^j$, with the output denoted as: $\boldsymbol{\alpha}_j := \boldsymbol{w}_u^j \tilde{\boldsymbol{o}}_E + b_u^j \mathbf{1}^T_{n_L n_{\text{QoS}}}$, which has dimensions $1 \times n_L n_{\text{QoS}}$. b_u^j is the additional bias, which is here a scalar, on the jth branch. The ith entry of $\boldsymbol{\alpha}_j$ represents the advantage function of action i for RB j. Additionally, following [77], dueling is applied to estimate the value function. Here, $\tilde{\boldsymbol{o}}_E$ is multiplied by a $1 \times d_e$ vector, $\boldsymbol{g}$ and a vector $b\mathbf{1}^T_{n_L n_{\text{QoS}}}$ corresponding to the bias is added, yielding a $1 \times n_L n_{\text{QoS}}$ vector $\boldsymbol{v}$, where the ith entry of $\boldsymbol{v}$ represents the estimated state value of the ith buffer. This vector $\boldsymbol{v}$ is then added on each branch j to produce the vector $\boldsymbol{Q}_j$, where each entry is an estimate of the Q-value for each action. To prevent suboptimal decisions due to selection of an empty buffer, action masking [75] is applied on each branch. Section 3.3.2 provides the detailed implementation of the mask. Finally, the action a_k^j is selected by taking the $\arg\max$ on the masked vector $\boldsymbol{Q}_j$. As highlighted in gray in Figure 3.1, $\boldsymbol{W}_e$, EOT, $\boldsymbol{w}_u^j$, and $\boldsymbol{g}$ are learned during the training phase.

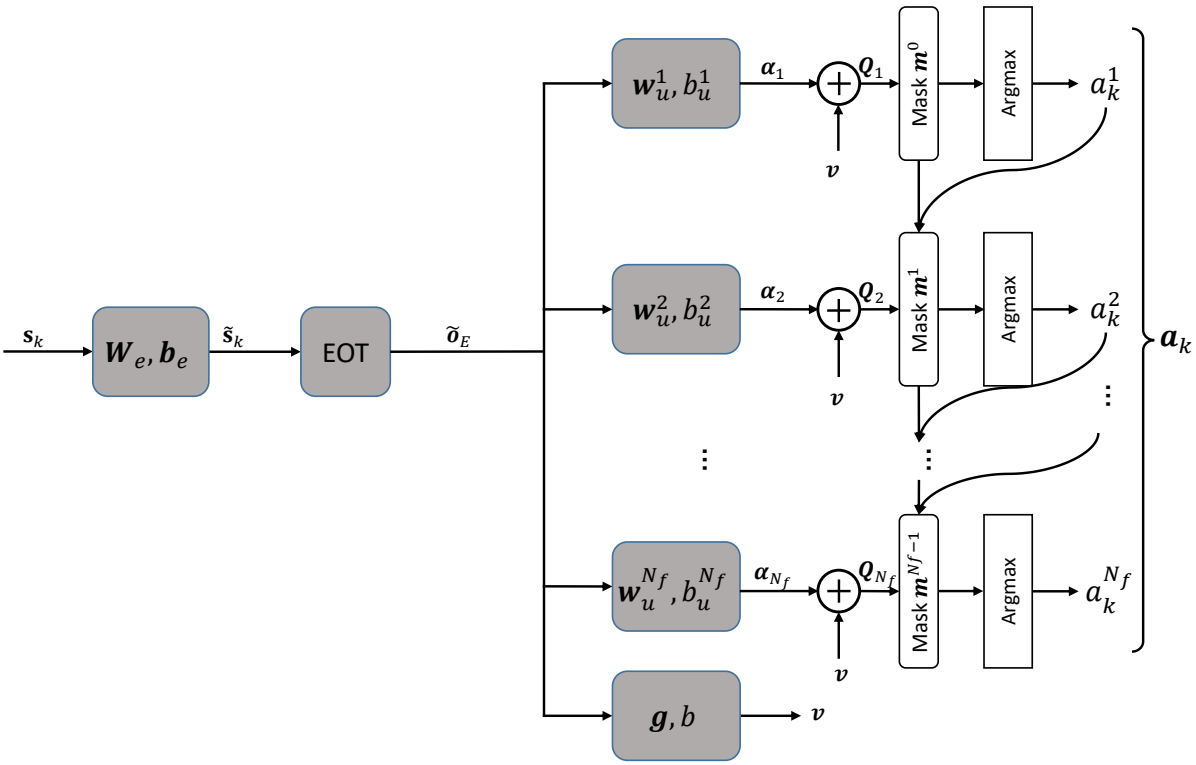


Figure 3.1: EOT-AB architecture combining an EOT and an AB structure. Gray blocks are learned.

Table 3.1 details the complexity of the proposed architecture.

Table 3.1: Number of FLOPs of the proposed architecture, EOT-AB.

Operation	Number of FLOPs
Embedding $\boldsymbol{W}_e$	$2 \times d_e \times n_{\text{features}} \times n_L n_{\text{QoS}}$
EOT (from Table A.1)	$2n_{\text{QoS}} n_L \left(d_e(4d_e + 2n_L n_{\text{QoS}} + 2d_{\text{mlp}} + 9) + 2Hn_L n_{\text{QoS}}\right)$ $-(n_L n_{\text{QoS}})^2 - n_L n_{\text{QoS}} d_e$
Dueling	$2 \times d_e \times n_L n_{\text{QoS}}$
Projection into advantages	$2 \times N_f \times d_e \times n_L n_{\text{QoS}}$
Q-values (advantage plus value)	$n_L n_{\text{QoS}} \times N_f$
Total	$n_L n_{\text{QoS}} \left[2d_e(n_{\text{features}} + 4d_e + 2n_L + 2d_{\text{mlp}} + 10 + N_f) + 4Hn_L n_{\text{QoS}}\right]$ $-(n_L n_{\text{QoS}})^2 - n_L n_{\text{QoS}} d_e$

3.3.2 Action masking procedure

The action masking procedure [75] aims to prevent the selection of "invalid" actions, i.e. actions that are not possible or allowed. In [75], authors:

1. Theoretically validate this approach.
2. Demonstrate that action masking improves training convergence and inference performance compared to more conventional methods, such as *invalid action penalty* that consists into attributing a bad reward to invalid action.

In the scheduling problem, there is no invalid actions, but rather suboptimal ones when an empty buffer is selected. Here, we propose using action masking for empty buffers.

Once the actions $\boldsymbol{a}_k$ have been determined, the RBs are filled with the bits taken from the selected buffers. Note that during this process, no new bit arrives, the buffer contents evolve only due to bits extraction. The RBs are filled sequentially, and for each RB, the bits are extracted from the buffer (the selection order of the RBs is not significant). As a result, a buffer may be selected multiple times, reflected by multiple occurrences of the same index in $\boldsymbol{a}_k$.

Two situations can lead to the selection of an empty buffer:

1. The first time a buffer is selected when it is already empty.
2. A buffer is selected multiple times, eventually emptying the buffer.

Therefore, we need to track each buffer's status (empty or not) after each RB allocation, and, when an empty buffer is detected, set its mask to empty. To do that, knowing the channel quality, with the CQI for instance, and therefore the quantity of data bits that can be transmitted per RB and the quantity of packets in the buffer, it is possible to determine the number of RBs, z_i^1, required to empty the buffer i. At each RB allocation, the remaining number of RBs required to empty the buffer is updated as follows:

$$z_i^j = \begin{cases} z_i^{j-1} - 1, & \text{if buffer } i \text{ is selected} \\ z_i^{j-1}, & \text{otherwise.} \end{cases} \tag{3.4}$$

The mask m_i^{j-1} associated with buffer i for the jth RB is:

$$m_i^{j-1} = \begin{cases} 1, & \text{if } z_i^j > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

The mask vector used for the jth RB is denoted as $\boldsymbol{m}^{j-1} = [m_0^{j-1}, \ldots, m_{n_L n_{\text{QoS}}-1}^{j-1}]$. This sequential adaptive masking procedure, to the best of our knowledge, is not covered in the literature and is illustrated in Figure 3.1 .

In this figure, the RBs are selected sequentially from RB $\#1$ to RB $\#N_f$ (top to bottom). The initial mask, $\boldsymbol{m}^0$, is set based on the buffer statuses at the beginning of the slot. For RB $\#1$, the buffer a_k^1 is selected among the non-empty buffers, thanks to the mask $\boldsymbol{m}^0$, which is applied to the Q-values in $\boldsymbol{Q}_1$. After bits are extracted from buffer a_k^1, we update the mask if the buffer becomes empty. This corresponds to the curved arrow pointing from a_k^1. We then update the mask, resulting into $\boldsymbol{m}^1$, apply $\boldsymbol{m}^1$ to the second branch, and repeat the process until the last branch.

3.4 Performance evaluation

We implement our architecture within the Nokia's WS environment that is suited for RL algorithms evaluation since its implementation follows the conventional *state, action, reward* of the RL scheme [80]. We focus in this chapter on the *TimeFreqResourceAllocation-v0* (TFRA) challenge presented in [6], and referred to as WS-TFRA in the sequel. In the following sections, we describe the WS-TFRA environment and the corresponding communication model, followed by the implemented RL model.

3.4.1 Wireless Suite environment

WS-TFRA implements a BS designed to send traffic to n_L UEs, which are uniformly distributed within a 1000-meter square around it. The BS transmits with effective isotropic radiated power (EIRP) $\mathcal{P}$ (in dB), using carrier frequency f_c. The total bandwidth W is divided into N_f RBs, which are allocated by the scheduler to UEs at each slot. The UEs move according to a random walk at constant speeds, which is independently sampled from a normal distribution, with mean 1.36 and standard deviation 0.19 in meter per second [81]. Free space propagation is assumed along with shadowing. The communication model is detailed in Section 3.4.2.

The data to be scheduled to the different UEs are stored in finite length FIFO buffers at the BS, each buffer containing at most B packets.

There are two kinds of QoS classes: GBR and non-GBR, and four types of traffic: voice, video, delay-critical, and web, each with a specific QoS. Voice, video, and delay-critical are GBR, while web is non-GBR. Let us note n_{QoS} the number of services and $\mathbb{Q} := \{1, \dots, n_{\text{QoS}}\}$. In the WS framework, we have $n_{\text{QoS}} = 4$ and the mapping between $q \in \mathbb{Q}$, services and QoS class is given in Table 3.2.

Table 3.2: Mapping between q, services and QoS class.

q	Service	QoS class
1	Voice	GBR
2	Video	GBR
3	Delay-critical	GBR
4	Web	non-GBR

WS-TFRA ensures that the traffic types are equally represented, which requires that the number of UEs is a multiple integer of four, i.e. $n_L = 4p$, with $p \in \mathbb{N}^*$. In the following, our developments are presented in a more general manner for possible future extensions keeping the constraint $n_L = pn_{\text{QoS}}$. The traffic characteristics are defined by the inter-arrival time τ_q, i.e. the time between the arrival of two consecutive packets in the buffer, and the incoming packet size $\mathcal{B}_q$, where $q \in \mathbb{Q}$.

The scheduling sequence is depicted as follows. At each slot k, the scheduler selects the different UEs to be served on the N_f RBs of the slot. To do so, the BS selects one UE per RB, with the possibility

of allocating multiple RBs to the same UE. The number of bits delivered by the BS to each selected UE depends on the CQI index $n_\ell^{\mathrm{CQI}} \in \{0, \ldots, n_{\mathsf{CQI}}\}$ with $n_{\mathsf{CQI}} = 15$, where 0 represents the worst channel quality, and 15 represents the best. Depending on the number of allocated RBs, the CQI and the packet sizes, a packet may not be completely transmitted, resulting in partial packet transmission. The age of a packet continues to increase as long as it is not fully transmitted. Once the bits have been extracted from the buffers, the age of the packets is incremented.

Remark: unlike in Chapter 2, the packets exceeding the packet delay budget (PDB) are not discarded. New packets arrive in the buffers according to τ_{q_ℓ} and $\mathcal{B}_{q_\ell}$, which depends on the traffic type of UE ℓ. If a new packet arrives while the corresponding buffer is full, a BO occurs, and the packet is lost. Finally, the UEs move, and the sequence is repeated at each slot.

Notice that in this chapter:

- A packet may not be completely transmitted in one slot, resulting in partial packet transmission, whereas in Chapter 2, the packets are always transmitted in one slot.
- The WT of the packets continues to increase even if it exceeds their PDB (corresponding to the DC), while in Chapter 2, these packets are deleted.
- The channel quality changes at each slot as users travel and shadowing is considered, whereas in Chapter 2, it is fixed.
- Packets arrive individually, with each arrival depends on an inter-arrival time in slot, whereas in Chapter 2 multiple packets may arrive simultaneously, according to a Poisson distribution.

3.4.2 Communication model

Let δ_ℓ represent the distance between UE ℓ and the BS. WS assumes free-space propagation and implements the path loss for UE ℓ in dB as:

$$\mathrm{PL}_\ell := 20 \log_{10} \left(\frac{4\pi \delta_\ell W}{c} \right) + \nu_\ell, \tag{3.6}$$

where c is the speed of light in vacuum and ν_ℓ corresponds to the shadowing, which is modeled as a gaussian random variable with zero mean and standard deviation σ which is set to 6 dB in WS.

The received power at UE ℓ in dB is:

$$\mathcal{P}_\ell^{\mathrm{Rx}} = \mathcal{P} - \mathrm{PL}_\ell. \tag{3.7}$$

The instantaneous signal-to-interference-plus-noise ratio (SINR) at UE ℓ is computed as:

$$\Gamma_\ell = \frac{10^{\frac{\mathcal{P}_\ell^{\mathrm{Rx}}}{10}}}{N_{\mathrm{th}} + I}, \tag{3.8}$$

where N_{th} represents the constant thermal noise, set to 2×10^{-11}, and I is a constant interference term, set to $10^{-\frac{105}{10}}$.

The spectral efficiency for UE k is determined as:

$$\mathrm{SE}_\ell = \log_2 \left(1 + \frac{\Gamma_\ell}{\Gamma_{\mathrm{map}}} \right), \tag{3.9}$$

where Γ_{map} is a mapping coefficient from SINR to spectral efficiency, as defined in [82].

Finally, the CQI is determined using the look-up table (LUT) provided in [83, Table 5.2.2.1-2] and drives the number of bits that are extracted from the buffers.

3.4.3 State spaces and reward of WS-TFRA and proposed adaptations

In this section, we describe the state and reward used in our architecture, which are derived from those of WS-TFRA. Several *adaptations* are required to convert the outputs of WS-TFRA to fit our architecture:

- Going from sequential to joint RB allocation.
- Adapting the state vector format to the MSI format.

Next, for the state and reward, we first review the outputs of WS-TFRA, followed by a description of the proposed adaptations.

3.4.3.1 State spaces

State space from WS-TFRA. The state space implemented in WS is the concatenation of the features related to each buffer and the RB index in the slot. The features related to buffer ℓ for RB j during time slot k is denoted by $\mathbf{f}_{\ell,k}^{j}$ and is defined as the vector of dimensions $(2B + n_{\text{QoS}} + 1) \times 1$:

$$\mathbf{f}_{\ell,k}^{j} := [\mathbf{b}_\ell, \mathbf{d}_\ell, n_\ell^{\text{CQI}}, \mathbf{h}_\ell]^T, \tag{3.10}$$

where $\mathbf{b}_\ell := [b_{1,\ell}, \ldots, b_{B,\ell}]$ and $\mathbf{d}_\ell := [d_{1,\ell}, \ldots, d_{B,\ell}]$ are the vectors containing the number of bits and the WT of each packet, respectively (the indexes of the slot and the RB are omitted), n_ℓ^{CQI} is the CQI, and $\mathbf{h}_\ell$ is the one-hot encoded vector of the QoS, where all entries equal to zero except the q_ℓth entry, which is equal to one, $q_\ell \in \mathbb{Q}$. The resulting state provided by the WS-TFRA at RB j of slot k can thus be written as the $(n_L(2B + n_{\text{QoS}} + 1) + 1) \times 1$ vector:

$$\mathbf{s}_{k,WS}^{j} := [{\mathbf{f}_{0,k}^{j}}^T, \ldots, {\mathbf{f}_{n_L-1,k}^{j}}^T, j]^T. \tag{3.11}$$

Naive adaptation. Since we propose a joint UE allocation for all the RBs in the slot, we only need the state at the beginning of the slot, and the RB index is thus useless in the state. Moreover, we propose an architecture that can handle a variable number of UEs that takes as an input a stacked version of the buffer information. The state space is therefore adapted as follows:

$$\tilde{\mathbf{f}}_{\ell,k}^{1} := \left[\frac{\mathbf{b}_\ell}{\mathcal{B}_{q_\ell,th}}, \frac{\mathbf{d}_\ell}{D_{q_\ell}}, \frac{n_\ell^{\text{CQI}}}{15}, \mathbf{h}_\ell \right]^T. \tag{3.12}$$

In (3.12), the data pertaining to the different types of traffic are normalized, by $\mathcal{B}_{q_\ell,th}$ for the number of bits of the different packets and by D_{q_ℓ} for the PDB, according to their type of traffic and the CQI n_ℓ^{CQI} is normalized by its maximum value 15. It is worth noting that the values of $\mathbf{d}_\ell / D_{q_\ell}$ may exceed 1 if the according packets exceed their PDB since they are not deleted. Then, the whole state is the matrix of dimension $(2B + n_{\text{QoS}} + 1) \times n_L$:

$$\mathbf{s}_{k,\text{Naive}} := [\tilde{\mathbf{f}}_{0,k}^{1}, \ldots, \tilde{\mathbf{f}}_{n_L-1,k}^{1}]. \tag{3.13}$$

The resulting scheduler is denoted by EOT-AB-N (EOT-AB-Naive).

Proposed adaptation. In the naive adaptation, an entry in (3.12) can correspond to different traffic types, while the DNN applies the same weight to this entry regardless of the traffic type. As a result, the network must rely heavily on the one-hot encoded vector $\mathbf{h}_\ell$ to differentiate between traffic types. To address this issue, we propose an alternative adaptation of the state space.

To allow the architecture to better distinguish the different traffic types, we propose a new state space by shifting traffic features in a larger dimension according to the traffic type by applying a Kronecker product between the one-hot vector $\mathbf{h}_\ell$ and the normalized features, except the CQI that is common to all services. The resulting state space can be written as a vector of dimensions $(2Bn_{\mathrm{QoS}}+1)\times 1$:

$$\mathbf{f}_{\ell,k}^{1,\mathrm{prop}} = \left[\mathbf{h}_\ell \otimes \left[\frac{\mathbf{b}_\ell}{\mathcal{B}_{q_\ell,th}}, \frac{\mathbf{d}_\ell}{D_{q_\ell}}\right], \frac{n_\ell^{\mathrm{CQI}}}{15}\right]^T, \tag{3.14}$$

where $\otimes$ represents the Kronecker product. For instance, with the four types of traffic of the WS environment, (3.14) yields:

$$\mathbf{f}_{\ell,k}^{1,\mathrm{prop}} = \begin{cases} [\frac{\mathbf{b}_\ell}{\mathcal{B}_{1,th}}, \frac{\mathbf{d}_\ell}{D_1}, \underbrace{0,\ldots,0}_{3\times 2B \text{ times}}, \frac{n_\ell^{\mathrm{CQI}}}{15}]^T, & \text{if } \mathbf{h}_\ell = [1,0,0,0] \\ [\underbrace{0,\ldots,0}_{1\times 2B \text{ times}}, \frac{\mathbf{b}_\ell}{\mathcal{B}_{2,th}}, \frac{\mathbf{d}_\ell}{D_2}, \underbrace{0,\ldots,0}_{2\times 2B \text{ times}}, \frac{n_\ell^{\mathrm{CQI}}}{15}]^T, & \text{if } \mathbf{h}_\ell = [0,1,0,0] \\ [\underbrace{0,\ldots,0}_{2\times 2B \text{ times}}, \frac{\mathbf{b}_\ell}{\mathcal{B}_{3,th}}, \frac{\mathbf{d}_\ell}{D_3}, \underbrace{0,\ldots,0}_{1\times 2B \text{ times}}, \frac{n_\ell^{\mathrm{CQI}}}{15}]^T, & \text{if } \mathbf{h}_\ell = [0,0,1,0] \\ [\underbrace{0,\ldots,0}_{3\times 2B \text{ times}}, \frac{\mathbf{b}_\ell}{\mathcal{B}_{4,th}}, \frac{\mathbf{d}_\ell}{D_4}, \frac{n_\ell^{\mathrm{CQI}}}{15}]^T, & \text{if } \mathbf{h}_\ell = [0,0,0,1]. \end{cases} \tag{3.15}$$

In the proposed architecture described in Section 3.3, the features from the various buffers are projected linearly using a weight matrix $\boldsymbol{W}_e$. Since the offset depends on the traffic type, only a specific parts of $\boldsymbol{W}_e$ is dedicated to a specific type of traffic, allowing the EOT to apply distinct weights accordingly. Indeed, for a given type of traffic $q \in \mathbb{Q}$, only the columns of $\boldsymbol{W}_e$ from $2B(q-1)+1$ to $2qB$ and the last one (for the CQI) are used. Unlike the other features, the CQI is not shifted, as it consistently represents the same amount of bits that can be transmitted, regardless of the QoS.

Subsequently, the overall state of the network $\mathbf{s}_{k,\mathrm{Prop}}$ is defined by staking the state of all the buffers defined in (3.14), which is a matrix with dimensions $(2Bn_{\mathrm{QoS}}+1)\times n_L$:

$$\mathbf{s}_{k,\mathrm{Prop}} := [\mathbf{f}_{0,k}^{1,\mathrm{prop}}, \ldots, \mathbf{f}_{n_L-1,k}^{1,\mathrm{prop}}]. \tag{3.16}$$

The resulting scheduler is denoted by EOT-AB-P (EOT-AB-Proposed).

3.4.3.2 Reward

The reward implemented in WS is given by:

$$r_{k,WS} = -n_{d,k} - n_{b,k}, \tag{3.17}$$

where $n_{d,t}$ and $n_{b,t}$ denote the total number of bits subject to DV in the buffers and the number of bits in the non-GBR buffers, respectively, during the frame duration. The reward is computed after the RB allocation and before new packets arrive. The objective of this reward is thus to minimize the number of bits present in the non-GBR buffers and the number of bits exceeding the PDB for both GBR and non-GBR ones.

Proposed adaptation. For the GRS approach, we propose to use the reward (3.17), mapped to the range $[0,1]$ using the exponential function:

$$r_{k,a} = \exp\left(r_{k,WS}\right). \tag{3.18}$$

3.4.4 Heuristics used in wireless suite

We use three heuristics from the WS environment as baselines for comparing the proposed DRL schedulers. Notice that the heuristics work in an SRS fashion, i.e., the state of the selected buffer for each RB is updated when it is chosen, while the WT of the packets in the different buffers remains constant within the frame duration.

Remark: In this section, to simplify the notation, the index k denotes the current combination of slot and RB, instead of using separate indices for slots and RBs.

3.4.4.1 Proportional fair

The implemented expression of the PF algorithm differs from that of [20], as it accounts for the HoL and the buffer load, but not for the average achieved rate. This expression of PF uses the spectral efficiency, which can be determined thanks to the CQI and tables of conversion [83]. Because the spectral efficiency is directly related to capacity, we continue to use the notation c_ℓ to denote spectral efficiency, rather than the transmission rate as in the previous chapters. The used PF algorithm is named *proportionalFairChannelAware* in WS, and its expression is:

$$h_{\mathrm{PF}}(\mathbf{x}_{\ell,k}) := \frac{1 + d_{0,\ell,k}}{D_{q\ell}} b_{\ell,k} c_{\ell,k} \tag{3.19}$$

where $b_{\ell,k}$ is the total number of bits in buffer of UE ℓ at time k.

3.4.4.2 Knapsack

The KP from the WS environment [6] is defined as the weighted sum of hyperbolic tangents. Its expression is given by (1.23), with $\alpha_j = 1$ for $j \in 1, 2, 3, 4$.

First, the expression in (1.24) remains unchanged.

Second, since no PLR targets are defined for the different traffic types, and because it may represent a channel metric as discussed in Section 1.3.7, (1.25) is adapted using the CQI, normalized by 15, the maximum CQI value:

$$v_2(\mathbf{x}_{\ell,k}) := \frac{n^{\mathsf{CQI}}_{\ell,k}}{15}. \tag{3.20}$$

Third, since bearer priority is not considered, a measure of allocation fairness is used instead, following the KP expression in [6]. Accordingly, (1.26) is adapted as:

$$v_3(\mathbf{x}_{\ell,k}) := \frac{1}{1 + z_{\ell,k}} \tag{3.21}$$

where $z_{i,k}$ is the number of times that UE ℓ is selected since the beginning of the simulation until k.

Finally, (1.27) takes into account a quantity of data in the buffer. It is adapted in [6] as follows:

$$v_4(\mathbf{x}_{i,k}) := \frac{b_{i,k}}{1 + \bar{b}_{i,k}} \tag{3.22}$$

where $b_{\ell,k}$ is the total number of bits in buffer of UE ℓ at k, $\bar{b}_{\ell,k}$ is the sliding window average number of bits in buffer of UE ℓ and $z_{\ell,k}$ is the number of times that the buffer ℓ is selected since the beginning of the simulation until k. Variable $\bar{b}_{\ell,k}$ is computed as follows:

$$\bar{b}_{\ell,k} = \begin{cases} (1 - \frac{N_f}{\tau_{\mathrm{win}}})\bar{b}_{\ell,k-1} + b_{\ell,k-1}\frac{N_f}{\tau_{\mathrm{win}}} & \text{if } k \bmod \tau_{\mathrm{win}} \neq 0 \\ 0 & \text{if } k \bmod \tau_{\mathrm{win}} = 0 \end{cases}, \tag{3.23}$$

where $\tau_{\text{win}} = 15N_f$ [6].

3.4.4.3 Bosch agent

The Bosch agent (BA) was specifically designed for the WS-TFRA environment and is therefore not presented in Section 1.3. It was introduced in [84], with the following expression:

$$h_{\text{BA}}(\mathbf{x}_{\ell,k}) := \sum_{j=1}^{4} \alpha_j v_j(\mathbf{x}_{\ell,k}), \tag{3.24}$$

where α_j, with $j \in \{1, 2, 3, 4\}$, are the weights of the different features, determined by the optimization methods of [84], v_1, v_2 and v_3 are given by (1.24), (3.20) and (3.21) respectively and:

$$v_4(\mathbf{x}_{\ell,k}) := \frac{b_{\ell,k}}{B\mathcal{B}_{\max}}, \tag{3.25}$$

where $\mathcal{B}_{\max} = \max_q \mathcal{B}_{q,th}$ is the maximum number of bits per packet.

Remark: (3.25) prioritizes buffers with more packets, such as (3.22).

According to the value of α_j in [6], the BA gives more importance to the number of bits in the buffers and to the CQI. In contrast, the WT and the fairness are less important according to their weights. As a result, it may prioritize non-GBR buffers and those with favorable channel conditions, potentially at the expense of GBR buffers whose WT values are approaching D_{q_ℓ}.

3.4.5 Communication setup

We use the default parameters of the WS-TFRA: a bandwidth of $W = 5$ MHz, $N_f = 25$ RBs, and $B = 8$. The parameters of the four types of traffic and QoS parameter D_q are reported in Table 3.3. For GBR traffic, both b_q and τ_q are fixed, while for the non-GBR traffic, they are drawn according to a geometric distribution $\mathcal{G}$. The probability mass function (pmf) of the geometric distribution $\mathcal{G}$, with mean $\frac{1}{p}$ where $\frac{1}{p} \in]0, 1[$, is defined as:

$$P(X = k) = \left(1 - \frac{1}{p}\right)^{k-1} \frac{1}{p}, \qquad \text{for} \quad k = 1, 2, 3, \ldots, \tag{3.26}$$

and,

$$b_4 = \min\left(\max\left(1, \mathcal{G}\left(\frac{1}{20\,000}\right)\right), \mathcal{B}_{th}\right), \tag{3.27}$$

$$\tau_4 \sim \mathcal{G}\left(\frac{1}{\beta}\right) \tag{3.28}$$

with $\mathcal{B}_{th} = 41\,250$ and $\beta = 10$, the default parameters in WS-TFRA.

Table 3.3: QoS and traffic parameters.

Service	QoS class	q	D_q	b_q	τ_q
Voice	GBR	1	100	584	20
Video	GBR	2	150	41 250	33
Delay-critical	GBR	3	30	200	20
Web	non-GBR	4	300	Eq. (3.27)	Eq. (3.28)

Table 3.4 details the training parameters of both EOT-AB.

An ϵ-greedy approach is used where the exploration parameter ϵ is set to $\epsilon_{\max} = 1$ at the beginning of the training and decayed by a factor ϵ_{decay} at each episode until reaching the value $\epsilon_{\min} = 0.01$.

Table 3.4: Training parameters.

	Parameter	Value
System model parameters	K	1000
	$\mathcal{P}$	13 dBm
	$f_{carrier}$	2655 MHz
	W	5 MHz
	N_f	25
	B	8
Both EOT-AB	Number of transformer layer	1
	Number of heads	4
DQL parameters	γ	0.99
	Learning rate	5×10^{-4}
	Batch size	64
	ϵ_{decay}	0.99
	Number of training episodes	2000
	Number of steps per episodes	$\lceil 65\,536/25 \rceil = 2622$
	Target network update period	20

3.4.6 Training and inference setup

The proposed scheduler is trained over 2000 episodes of $65\,536$ RBs (default value of the environment for one episode), which corresponds to 2622 steps, with a fixed $\beta = 10$. During the training, the number of UEs varies between each episode, with $n_L \in \mathcal{N} := \{32, 36, 40\}$. Our objective is to train a single DNN achieving good performance across different values of n_L, avoiding the need to implement a dedicated DNN for each possible n_L.

For the inferences, 1000 episodes are performed for each $n_L \in \mathcal{N}$, for each scheduler. Each episode consists on $K = 2622$ steps of 25 RBs.

Two kinds of inference are performed:

1. With the training setup, i.e. with $\beta = 10$ (all the environment parameters are the same).
2. With different values of β to evaluate the generalization capabilities of the EOT-AB-P.

We compare the different state spaces with three of the implemented heuristics in WS: the PF, BA and KP, such as described in Section 3.4.4.

3.4.7 Performance metrics

We compare the solutions in terms of the following performance metrics, collected at the end of the inference episodes:

- The average cumulative WS's reward, which is defined as:

$$R = \frac{1}{(K+1)N_f} \sum_{k=0}^{K} r_{k,WS}. \tag{3.29}$$

- The number of packets exceeding their PDB.
- The number of packets lost due to BO.
- The time spent by a packet in the buffer between arrival and departure in slot, referred to as PD in the sequel.

Since the age of a packet is determined by the age of the last transmitted bit:

- A packet is said to exceed its PDB as soon as, at least one bit of the packet exceeds the PDB.
- The PD is evaluated as the number of slots between the arrival of the packet in the buffer and the departure of the last bit of this packet from the buffer.

3.5 Performance analysis

This section is dedicated to the performance evaluation of both EOT-AB along with various other schedulers outlined in Section 3.4.4 that are already integrated into the WS-TFRA environment. We remind that the different heuristics follows the SRS approach. Notice that the less effective heuristics present in the environment are not included in this analysis.

We select the best values of d_e (assuming $d_{\text{mlp}} = d_e$) for the two state spaces. Table 3.5 reports the best values, noted $\tilde{d}_e$.

Table 3.5 gives the number of FLOPs of the different architectures for $n_L \in \{32, 36, 40\}$, using the expression of Table 3.1 for both EOT-AB.

Table 3.5 shows that the proposed architecture (EOT-AB-P) is about ten times less complex than the EOT-AB-N since the latter requires a higher-dimensional input.

Table 3.5: Number of FLOPs of the different architecture with $H = 4$ heads for MHA and to allocate $N_f = 25$ RBs.

	n_{features}	$\tilde{d}_e$	$n_L = 32$	$n_L = 36$	$n_L = 40$
EOT-AB-P	65	64	2 254 937	2 575 385	2 904 409
EOT-AB-N	21	256	27 136 281	30 677 465	34 251 801

3.5.1 Inference performance on the training setup

In this section, we set as during the training $\beta = 10$, and we evaluate the inference performance of the learned DNN for $n_L \in \mathcal{N}$.

Table 3.6 provides the average cumulative rewards, defined in (3.29), obtained by the different schedulers. The median, the standard deviation, the mean value, the minimum and the maximum of the reward obtained during the 1000 episodes are detailed in Tables 3.6a, 3.6b and 3.6c for 32, 36 and 40 UEs respectively. One can observe that:

- Both EOT-AB provide the best reward. The mean reward of both EOT-AB are at least twice better than the one of the other heuristics.
- The proposed architecture achieves a standard deviation twice less than the one of the other methods.

- Both EOT-AB achieve a minimum for 32 UEs that is superior to the maximum of the PF for the same number of UEs. In general, the minimum of the EOT-AB is approximately twice less than the minimum of the other methods.
- Concerning the maximum, both EOT-AB are approximately four times better than the other methods.

These results show that the EOT-AB architecture learns a policy that is more efficient than the heuristics in terms of the implemented reward.

Table 3.6: Reward for the different methods.

(a) Reward for the different methods for 32 UEs.

Method	median	mean	std	min	max
PF	-1797	-1814	243	-2852	-1057
Bosch	-1512	-1522	226	-2414	-754
Knapsack	-1412	-1427	224	-2362	-773
EOT-AB-P	$\mathbf{-446}$	$\mathbf{-457}$	$\mathbf{98}$	$\mathbf{-870}$	$\mathbf{-162}$
EOT-AB-N	$\mathbf{-432}$	$\mathbf{-441}$	$\mathbf{94}$	$\mathbf{-852}$	$\mathbf{-162}$

(b) Reward for the different methods for 36 UEs.

Method	median	mean	std	min	max
PF	-2829	-2886	451	-4761	-1640
Bosch	-2195	-2228	366	-3650	-1198
Knapsack	-2378	-2427	449	-4297	-1323
EOT-AB-P	$\mathbf{-722}$	$\mathbf{-747}$	$\mathbf{209}$	$\mathbf{-1692}$	$\mathbf{-284}$
EOT-AB-N	$\mathbf{-666}$	$\mathbf{-697}$	$\mathbf{195}$	$\mathbf{-1574}$	$\mathbf{-280}$

(c) Reward for the different methods for 40 UEs.

Method	median	mean	std	min	max
PF	-5032	-5209	1106	$-10\,596$	-2649
Bosch	-3709	-3916	1051	$-11\,888$	-1723
Knapsack	-4837	-5073	1343	-13009	-2041
EOT-AB-P	$\mathbf{-1788}$	$\mathbf{-1899}$	$\mathbf{689}$	$\mathbf{-5513}$	$\mathbf{-400}$
EOT-AB-N	$\mathbf{-1516}$	$\mathbf{-1815}$	2326	$-51\,588$	$\mathbf{-317}$

Table 3.7 compares the number of packets lost due to buffer overflow and PDB violations for $n_L \in \{32, 36, 40\}$ UEs. PF and BA show substantial degradation, particularly at higher loads. In contrast, EOT-AB-P and EOT-AB-N achieve significantly lower packet loss. Notably, EOT-AB-P achieves zero loss at 32 UEs and remains below 50 lost packets even at 40 UEs. EOT-AB-N shows similar trends, with an increase to 6983 packets at the highest load, still below the baseline methods, except KP.

Table 3.7: Number of lost packets due to BO and exceeding the PDB for 32, 36 et 40 UEs.

	32 UEs	36 UEs	40 UEs
PF	1813	18 650	143 048
Bosch	19 751	90 955	493 521
KP	3	5	200
EOT-AB-P	**0**	3	**42**
EOT-AB-N	**0**	**2**	6983

It is important to distinguish and analyze packet losses due to BO and PDB violations, as they reflect different network limitations (congestion handling and delay compliance) that directly impact QoS in 5G.

Table 3.8 provides the number of packets lost due to BO wrt n_L. The BA has the worst performance in terms of BO. The PF loses few packets due to BO. The KP loses more packets due to BO than the proposed solution. EOT-AB-P achieves the best performance in terms of BO. In contrast, while EOT-AB-N performs well for scenarios with 32 and 36 UEs, its performance significantly degrades for 40 UEs, with a loss exceeding 4000 in BO. All the packets lost by the KP is due to BO.

Table 3.8: Number of lost packets due to BO for 32, 36 et 40 UEs.

	32 UEs	36 UEs	40 UEs
PF	**0**	**0**	9
Bosch	**0**	275	50 796
KP	3	5	200
EOT-AB-P	**0**	**0**	**0**
EOT-AB-N	**0**	2	4107

Table 3.9 provides the number of packets exceeding the PDB. The BA has the worst performance in terms of both BO and PDB violations. The PF loses few packets due to BO but thousands exceed the PDB, and thus does not satisfy well the QoS. The KP has no packet exceeding the PDB. The proposed approach yields the best performance in terms of BO and is very close to the KP which has no packet lost in terms of PDB violation, outperforming the PF and KP.

Table 3.9: Number of transmitted packets exceeding the PDB for 32, 36 et 40 UEs.

	32 UEs	36 UEs	40 UEs
PF	1816	18 650	143 039
Bosch	19 751	90 680	442 725
KP	**0**	**0**	**0**
EOT-AB-P	**0**	3	42
EOT-AB-N	**0**	**0**	2876

We analyse in what follows the number of packets lost due to BO for each traffic type. Table 3.10 details the number of packets lost due to BO for each traffic type with the different values of n_L.

- Table 3.10a provides results with 32 UEs. One can observe that nearly all algorithms maintain zero packet loss across all traffic types.

- Table 3.10b provides results with 36 UEs. BA begins to show losses for voice and delay-critical traffic, algorithms such as PF, KP, and both EOT-AB variants still maintain a near loss-free operation.
- Table 3.10c provides results with 40 UEs. One can observe that only EOT-AB-P continues to have zero loss across all traffic types. BA suffers from considerable losses, particularly in voice and delay-critical traffic. EOT-AB-N obtains few losses for GBR traffic at the cost of the web traffic.
- One can remark that the KP loses packets only for web traffic, whatever the number of tested UE.

Table 3.10: Number of packets lost due to BO for each traffic for 32, 36 et 40 UEs.

(a) 32 UEs.

	Voice	Video	Delay-critical	Web
PF	**0**	**0**	**0**	**0**
Bosch	**0**	**0**	**0**	**0**
KP	**0**	**0**	**0**	3
EOT-AB-P	**0**	**0**	**0**	**0**
EOT-AB-N	**0**	**0**	**0**	**0**

(b) 36 UEs.

	Voice	Video	Delay-critical	Web
PF	**0**	**0**	**0**	**0**
Bosch	248	**0**	27	**0**
KP	**0**	**0**	**0**	5
EOT-AB-P	**0**	**0**	**0**	**0**
EOT-AB-N	**0**	**0**	**0**	2

(c) 40 UEs.

	Voice	Video	Delay-critical	Web
PF	1	**0**	**0**	8
Bosch	37 182	**0**	13 522	92
KP	**0**	**0**	**0**	200
EOT-AB-P	**0**	**0**	**0**	**0**
EOT-AB-N	40	104	**0**	3963

In Table 3.11, the number of packets exceeding the PDB for each traffic as well as the corresponding proportion, i.e. the number of transmitted packets exceeding the PDB over the total number of transmitted packets for the corresponding traffic, are provided for 32, 36 and 40 UEs in Tables 3.11a, 3.11b and 3.11c respectively. One can observe that:

- All schedulers have no packets exceeding the PDB for all the tested number of UEs for video and web traffic, except the EOT-AB-N for 40 UEs.
- The PF exceeds the PDB principally for the delay-critical traffic. It also exceeds the PDB for voice traffic with 40 UEs.
- The BA is the worst one in terms of number of packets exceeding the PDB.

- The EOT-AB-P has packets exceeding the PDB for voice and delay-critical traffic for 36 and 40 UEs.
- As seen in Table 3.9, the KP has no packets exceeding the PDB whatever the traffic.

The 5G specifications [38, Section 5.7.3.4] states that 98% of the packets from the GBR traffic shall not experience a delay exceeding its PDB. Therefore, only the KP and EOT-AB respects this QoS for all the tested number of UEs.

Table 3.11: Number of transmitted packets exceeding the PDB for each traffic for 32, 36 et 40 UE.

(a) 32 UEs.

	Voice		Video		Delay-critical		Web	
	Count	Prop.	Count	Prop.	Count	Prop.	Count	Prop.
PF	**0**	**0**	**0**	**0**	1816	1.7×10^{-3}	**0**	**0**
Bosch	44	4.2×10^{-5}	**0**	**0**	19 707	1.9×10^{-2}	**0**	**0**
KP	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
EOT-AB-P	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
EOT-AB-N	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

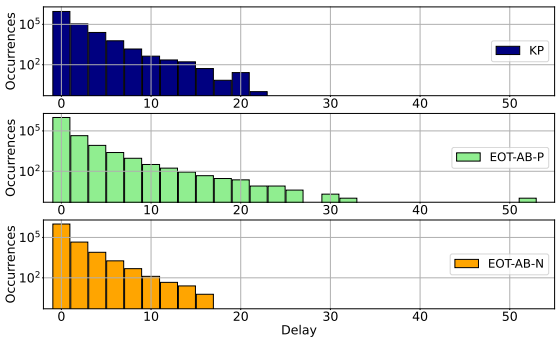
(b) 36 UEs.

	Voice		Video		Delay-critical		Web	
	Count	Prop.	Count	Prop.	Count	Prop.	Count	Prop.
PF	**0**	**0**	**0**	**0**	18 650	1.6×10^{-2}	**0**	**0**
Bosch	3268	2.8×10^{-3}	**0**	**0**	87 412	7.4×10^{-2}	**0**	**0**
KP	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
EOT-AB-P	2	1.7×10^{-6}	**0**	**0**	1	8.5×10^{-7}	**0**	**0**
EOT-AB-N	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

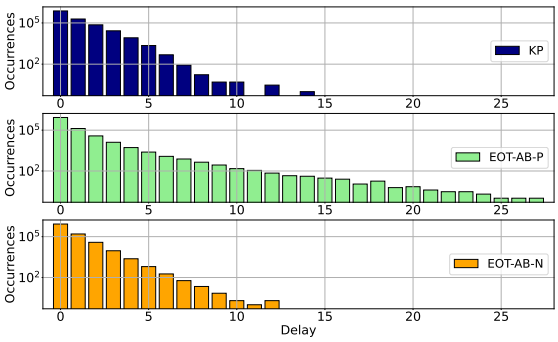
(c) 40 UEs.

	Voice		Video		Delay-critical		Web	
	Count	Prop.	Count	Prop.	Count	Prop.	Count	Prop.
PF	1029	8×10^{-4}	**0**	**0**	142 010	1.1×10^{-1}	**0**	**0**
Bosch	94 109	7.5×10^{-2}	**0**	**0**	348 616	2.7×10^{-1}	**0**	**0**
KP	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
EOT-AB-P	11	8.4×10^{-6}	**0**	**0**	31	2.4×10^{-5}	**0**	**0**
EOT-AB-N	148	1.1×10^{-1}	1595	2.0×10^{-3}	915	7.0×10^{-4}	218	8.3×10^{-5}

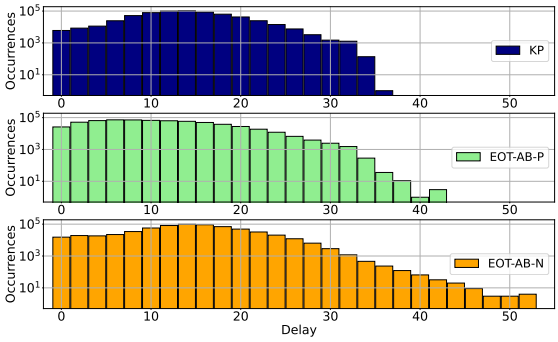
The PD is measured for each transmitted packets during all the inference episodes. Figures 3.2 and 3.3 depict the histograms and the cumulative distribution function (cdf) respectively of the PD for the voice, delay-critical, video and web traffic, for KP and both EOT-AB with 32 UEs. The red line represents the PDB and thus the bars at the right of this line represent the packets exceeding their PDB. The histograms are plotted in the $\log_{10}$ scale because of the high dynamic range of the PD values. The other heuristics are not displayed since they yield significantly worse performance.



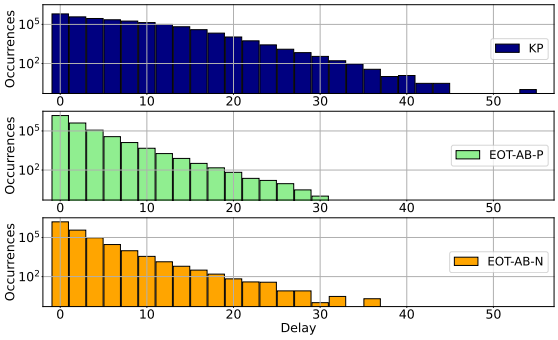
(a) Voice.



(b) Delay critical.

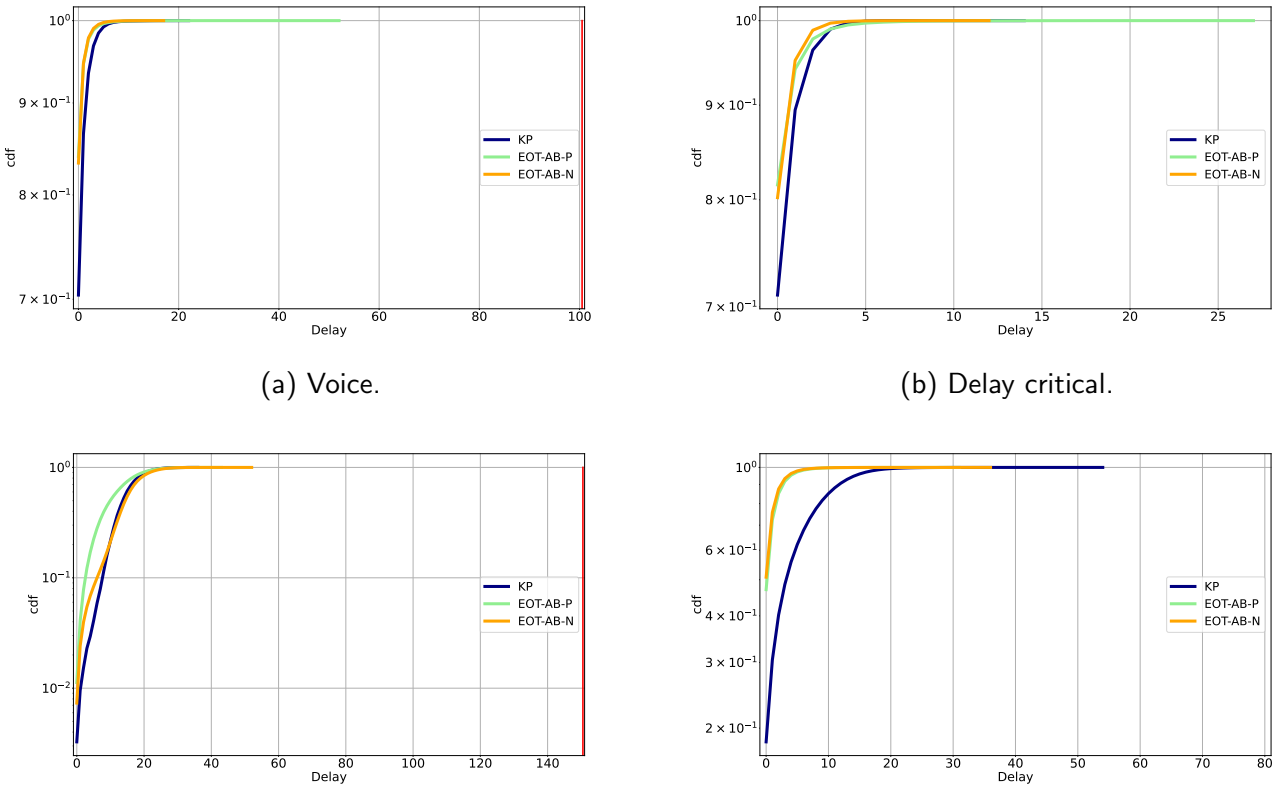


(c) Video.



(d) Web.

Figure 3.2: Histograms of the packet delays for 32 UEs.



(a) Voice.

(b) Delay critical.

(c) Video.

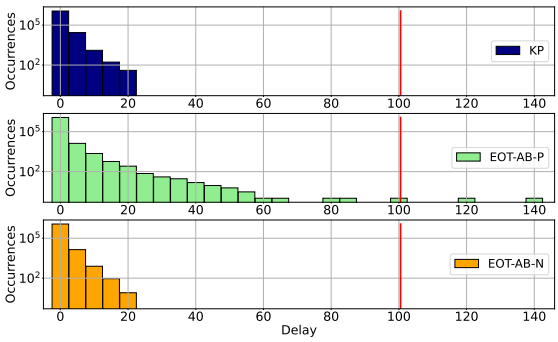
(d) Web.

Figure 3.3: CDF of the packet delays for 32 UEs.

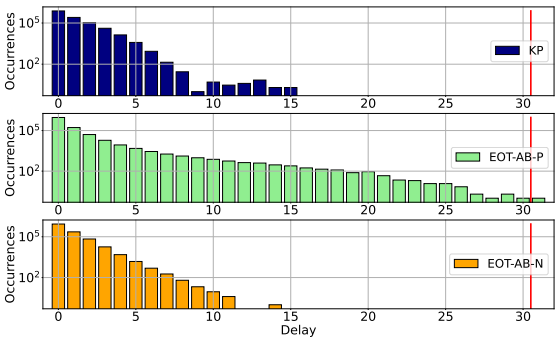
One can observe that:

1. The EOT-AB-P spreads more the packet transmissions for voice and delay-critical traffic than the KP and tends to serve the web packets faster. This behavior can be explained by the reward expression (3.18), which encourages the scheduler to prioritize non-GBR buffers by penalizing any residual bits left in them.
2. The majority of the packets for voice and delay-critical traffic are sent within few slots for both KP and EOT-AB-N. The voice and delay-critical packets are short and thus can be transmitted within few RBs.
3. To send more than 80% of video packets, both schedulers needs at least 15 slots. This can be explain by the size of the video packets which are more than 20 times larger than the delay-critical packets. Therefore they need more RBs to be fully sent, leading to larger transmission delay.
4. Concerning the web traffic, both EOT-AB send 90% of packets within less than 5 slots whereas the KP scheduler needs more than 10 slots for the same proportion. For the same number of slots, the KP sends 50% of the packets. This can be explained by the reward formulation that encourages both EOT-AB to serve the non-GBR buffers.

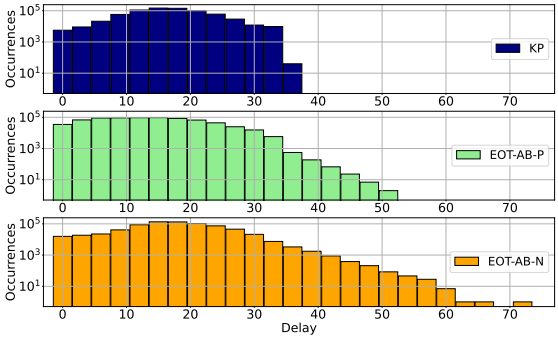
Figures. 3.4 and 3.5 depict the histograms and the cdf respectively of the PD for the voice, delay-critical, video and web traffic, for KP and both EOT-AB for a scenario with 36 UEs.



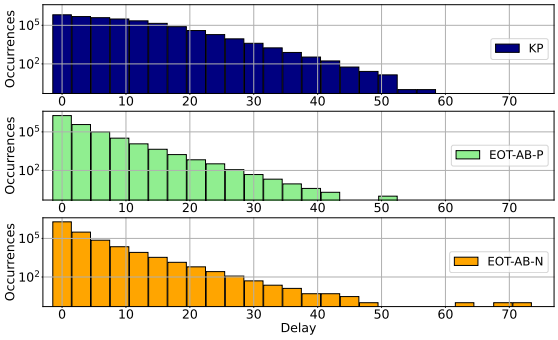
(a) Voice.



(b) Delay critical.

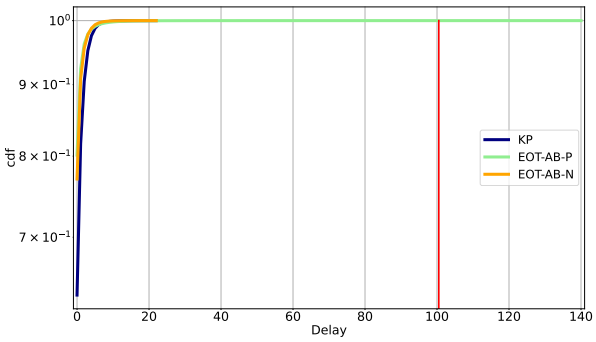


(c) Video.

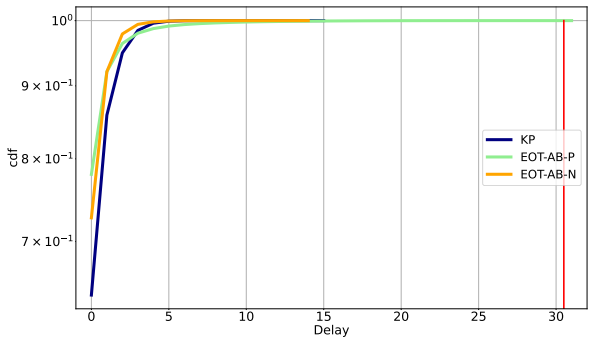


(d) Web.

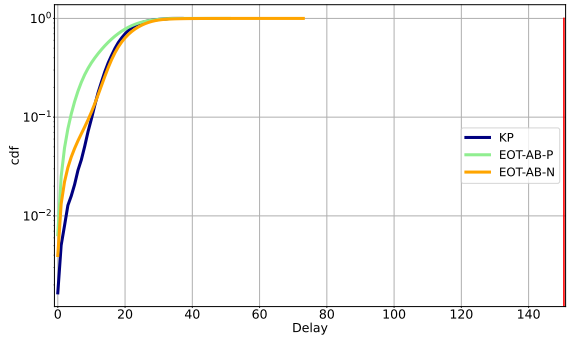
Figure 3.4: Histograms of the packet delays for 36 UEs.



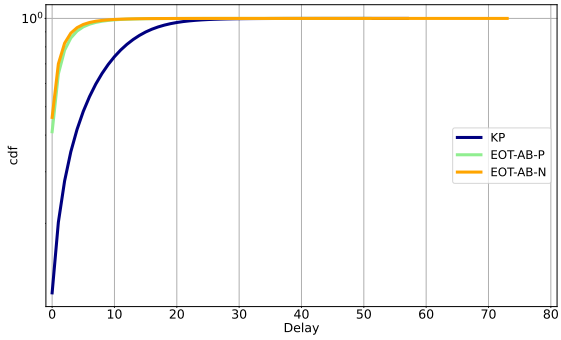
(a) Voice.



(b) Delay critical.



(c) Video.



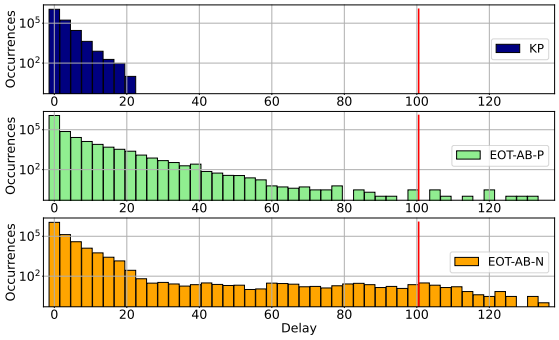
(d) Web.

Figure 3.5: CDF of the packet delays for 36 UEs.

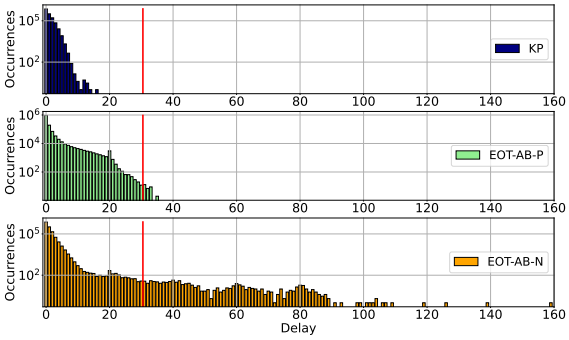
The same observations can be made as for 32 UEs, with the following differences:

1. The EOT-AB-P spreads more the transmission of packets for voice and delay-critical traffic than the KP and EOT-AB-N. However, this time, the EOT-AB-P has packets exceeding the PDB for these two traffic types.
2. The EOT-AB-P sends 90% of web packets within 5 slots whereas the KP does it within 15 slots.

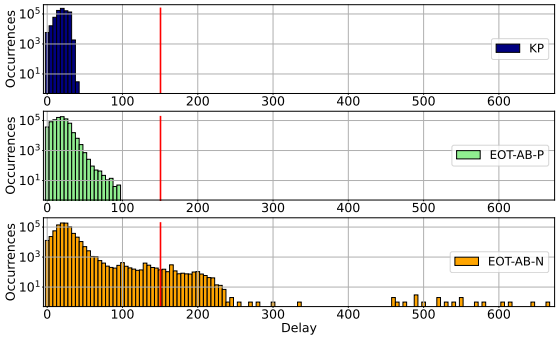
Figures. 3.6 and 3.7 depict the histograms and the cdf respectively of the PD for the voice, delay-critical, video and web traffic, for KP and both EOT-AB for a scenario with 40 UEs.



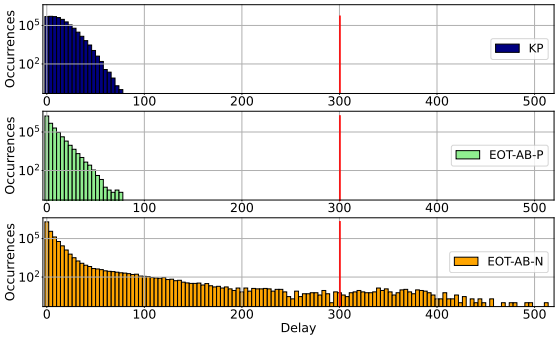
(a) Voice.



(b) Delay critical.

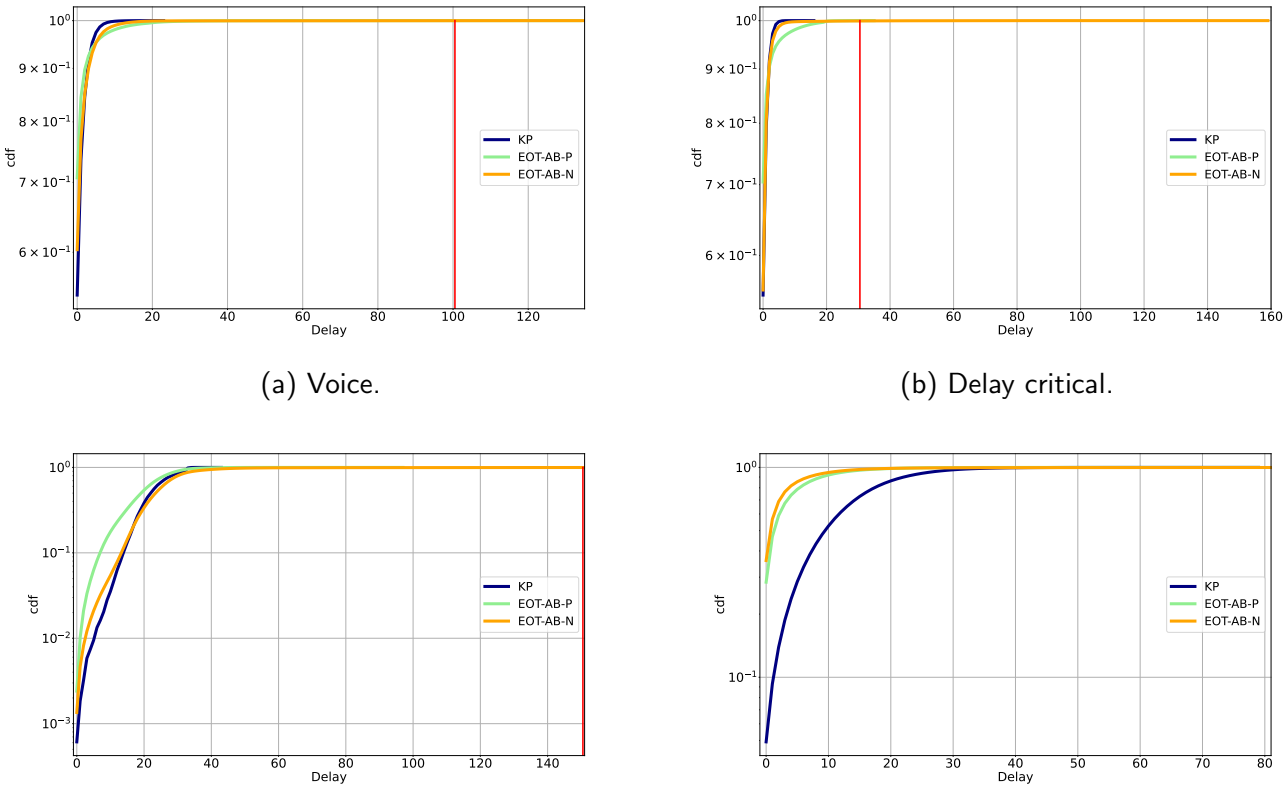


(c) Video.



(d) Web.

Figure 3.6: Histograms of the packet delays for 40 UEs.



(a) Voice.

(b) Delay critical.

(c) Video.

(d) Web.

Figure 3.7: CDF of the packet delays for 40 UEs.

The same observations can be made as for 32 and 36 UEs, with the following differences:

1. The EOT-AB spreads more the PD for video traffic compared to 32 and 36 UEs. The maximum PD has doubled.
2. The EOT-AB sends 90% of web packets within 10 slots whereas the KP does it within 25 slots.

Synthesis of the inference on the training setup. We have observed through simulations that the proposed solution outperforms heuristic schedulers in terms of reward and achieves lower total packet loss. Additionally, it enhances PD for non-GBR traffic at the cost of a degradation of those of delay-critical GBR and voice traffic, due to the reward formulation (3.18).

Furthermore, we have shown that the chosen state representation improves both generalization and computational efficiency compared to the "naive" representation described in (3.12).

We will focus in the next section on the EOT-AB-P architecture using the state representation defined in (3.14).

3.5.2 On the importance of the mask for the proposed architecture

The training and inference in Section 3.5.1 were performed using adaptive action masking. In this section, we assess the benefits of adaptive masking during the inference phase by comparing performance with and without adaptive action masking.

The number (resp. the proportion of packets) exceeding the PDB for each traffic obtained during the inference of the EOT-AB-P with and without mask is reported in Table 3.12a (resp. 3.12b).

One can remark that, without the action masking, there is a lot of packets exceeding the PDB for all the GBR traffic. No packets have exceeded the PDB for the web traffic. This can be explained by the reward that encourages the agent to serve the non-GBR traffic and by the PDB of this traffic which is much larger than the other PDB.

Table 3.12: Number of packets exceeding the PDB for the EOT-AB-P with and without mask.

(a) Number of packets.

	32 UEs	36 UEs	40 UEs
Voice without mask	5398	120 276	357 499
with mask	0	2	11
Video without mask	8762	56 768	116 794
with mask	0	0	0
Delay-critical without mask	45 359	249 102	231 963
with mask	0	1	31
Web without mask	10	0	0
with mask	0	0	0
Total without mask	59 529	426 146	706 256
with mask	0	3	42

(b) Proportion of packets.

	32 UEs	36 UEs	40 UEs
Voice without mask	5.1×10^{-3}	1.0×10^{-1}	2.7×10^{-1}
with mask	0	1.7×10^{-6}	8.4×10^{-6}
Video without mask	1.4×10^{-2}	7.9×10^{-2}	1.5×10^{-1}
with mask	0	0	0
Delay-critical without mask	4.3×10^{-2}	2.1×10^{-1}	1.8×10^{-1}
with mask	0	8.5×10^{-7}	2.4×10^{-5}
Web without mask	4.8×10^{-6}	0	0
with mask	0	0	0
Total without mask	1.2×10^{-2}	7.8×10^{-2}	1.2×10^{-1}
with mask	0	5.5×10^{-7}	6.9×10^{-6}

Table 3.13 shows the number of packets packets lost due to BO. Even if the number of packets exceeding the PDB for the delay critical is important, the trained EOT-AB-P does not loss packets for this QoS due to BO.

Table 3.13: Number of lost packet due to BO for the EOT-AB with and without mask.

	32 UEs	36 UEs	40 UEs
Voice without mask	1277	50 448	229 758
with mask	0	0	0
Video without mask	3	51	531
with mask	0	0	0
Delay-critical without mask	0	0	0
with mask	0	0	0
Web without mask	9511	112 472	394 330
with mask	0	0	0
Total without mask	10 791	162 971	624 619
with mask	0	0	0

Table 3.14 shows the number and proportion of RBs where an empty buffer is selected while there were packets in other buffers. For more than a quarter (resp. more than ten percent) of the RB, the EOT-AB-P selects an empty buffer for the setup with 32 UEs (resp. 36 and 40 UEs). When the number of UEs increases, it is easier to select a UE with packets, leading to a lower proportion for 36 and 40 UEs. Without adaptive mask, the EOT-AB-P selects therefore non-optimal action, causing packet loss and packet exceeding the PDB. This shows the importance of the adaptive masking during the inference in addition of the training.

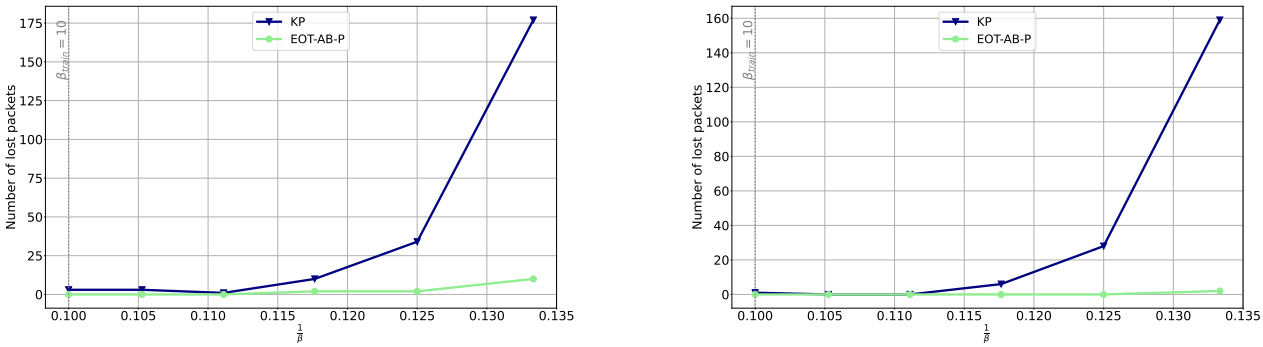
Table 3.14: Empty buffer selection for the EOT-AB-P without mask.

	Count	Proportion
32 UEs	17 027 151	26%
36 UEs	7 507 960	11.4%
40 UEs	7 347 526	11.2%

3.5.3 Generalization vs. β

The proposed architecture was trained with the mean inter-arrival time for non-GBR $\beta = 10$. We evaluate here the generalization capability of the proposed architecture vs. $\frac{1}{\beta}$. To do so, we saturate the traffic by reducing β (or increasing $\frac{1}{\beta}$). The metric considered here is the sum of the number of packets lost due to BO and the ones exceeding the PDB for the DC. These packets are included in the packet error rate (PER) in the 5G specifications [38, Section 5.7.3.5] and are thus referred to as "lost packets" in the sequel. 1000 inference episodes are performed for each value of β. The range of β values is adjusted based on the number of UEs to prevent over-saturation. After running all inference episodes, we observed that the trained architecture performed poorly in few cases, leading to unsatisfactory overall results. Notably, two particular runs significantly degraded the overall performance of the proposed architecture. We suspect that during these episodes the architecture encountered states that were far from those seen during training, causing it to deviate from its expected behavior. Further investigation should focus on detecting such deviations as they occur and therefore switch on a heuristic such as KP. The two least effective episodes for each scheduler are eliminated (a total of four episodes) to prevent the substantial distortions in the overall results. This approach retains the majority of the results while ensuring that no single scheduler's performance is unduly favored.

Figure 3.8 shows the lost packets as a function of $\frac{1}{\beta}$ for the KP and EOT-AB-P schedulers for $n_L = 32$. Figure 3.8a provides the results over the 1000 inference episodes. Figure 3.8b represents the results over the 996 inference episodes where the four episodes are withdrawn, corresponding to the two worst for EOT-AB-P and for KP.



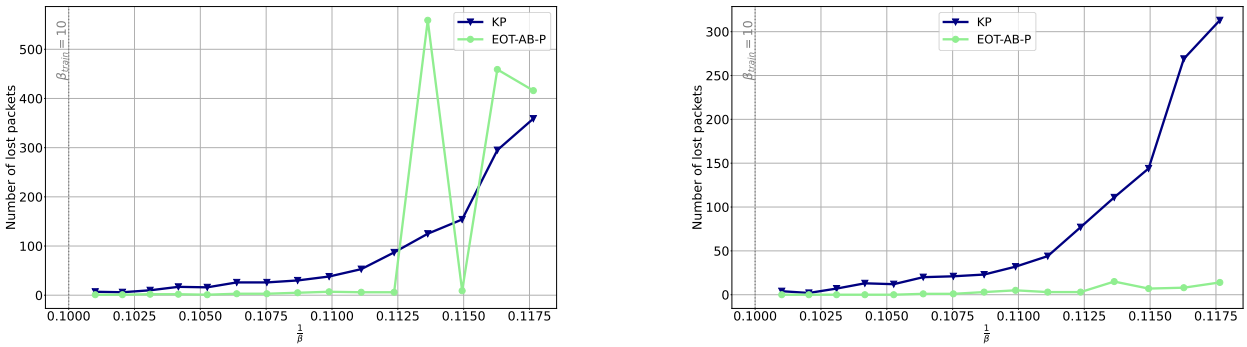
(a) With all the 1000 episodes.

(b) Without the worst episodes.

Figure 3.8: Number of lost packets vs. $\frac{1}{\beta}$ for $n_L = 32$.

One can observe a good generalization capability of the proposed approach, which loses less packets than the KP for all the simulated values of β. The two figures show no significant differences, indicating that packet loss is not concentrated in just four episodes. Therefore, there is no instance in which the trained scheduler exhibits underperformance, demonstrating robust generalization capabilities across all inference episodes.

Figure 3.9 plots the lost packets as a function of $\frac{1}{\beta}$ for the KP and EOT-AB-P schedulers for $n_L = 36$.



(a) With all the 1000 episodes.

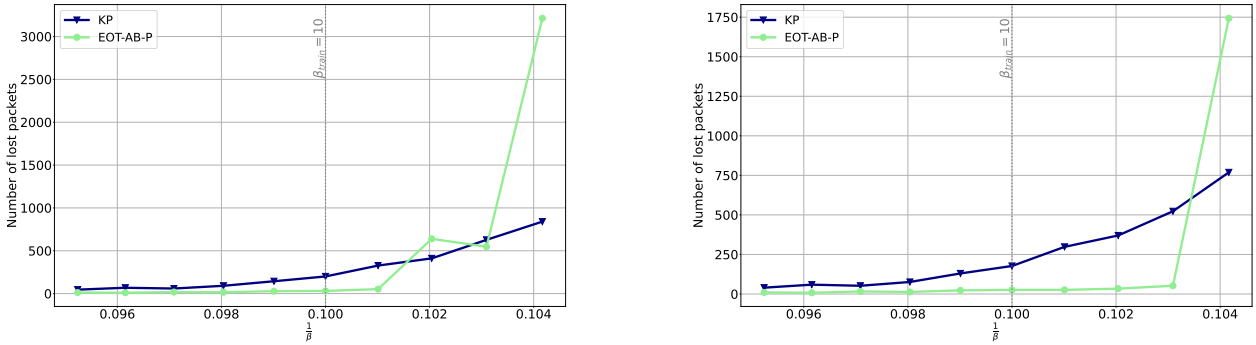
(b) Without the worst episodes.

Figure 3.9: Number of lost packets vs. $\frac{1}{\beta}$ for $n_L = 36$.

One can remark that for all the 1000 episodes, in Figure 3.9a, the EOT-AB-P performs poorly for $\beta = 8.5$, 8.6 and 8.8, corresponding to $\frac{1}{\beta} \approx 0.118$, 0.116 and 0.114, with more than 400 lost packets. When the worst episodes are withdrawn, in Figure 3.9b, the EOT-AB-P outperforms the KP scheduler for all the values of β, with less than 50 lost packets. Moreover, KP's performance varies little with the removal of its worst episodes. This suggests that:

- Losses of the KP schedulers appear evenly distributed among between episodes. Consequently, eliminating the least effective episodes has no apparent effect on its overall performance.
- Packet loss is mainly observed in a limited number of episodes for the EOT-AB-P, indicating that it has made poor decisions only in those instances. Further investigation is required to determine the factors that may have contributed to these choices.

Figure 3.10 plots the lost packets as a function of β for the KP and EOT-AB-P schedulers for $n_L = 40$.



(a) With all the 1000 episodes.

(b) Without the worst episodes.

Figure 3.10: Number of lost packets vs. $\frac{1}{\beta}$ for $n_L = 40$.

One can remark that for all the 1000 episodes, in Figure 3.10a, the EOT-AB-P performs poorly for $\beta = 9.6$, 9.7 and 9.8, corresponding to $\frac{1}{beta} \approx 0.104$, 0.103 and 0.102. When the worst episodes are withdrawn, in Figure 3.10b, the EOT-AB-P outperforms the KP scheduler for all the values of β except $\beta = 9.6$. One can still remark that KP's performance varies little.

Synthesis of the inference over different values of β. The EOT-AB-P sometimes struggles to generalize under heavy traffic conditions, i.e. when β decreases. This can be explained by the current reward formulation, which is ineffective for reducing packet loss during periods of high traffic load. As outlined in (3.17), the agent tends to prioritize scheduling non-GBR traffic. Consequently, when the packet inter-arrival rate decreases, more packets accumulate in the non-GBR buffers, inciting the trained scheduler to focus on emptying these buffers at the expense of other traffic types, leading to their starvation. Therefore, the delay-critical traffic may be starving, leading to delay-critical packets exceeding their PDB. In future work, it would be useful to detect when the method begins to diverge and switch to a heuristic like KP, which offers more consistent performance.

3.6 Conclusion

We addressed in this chapter the problem of the packet scheduling over a frame composed of RBs, as in the 5G context. We proposed solutions based on an architecture which is a combination of EOT and AB. This architecture is NLI thanks to the EOT and is capable to handle the very large action space induced by the joint RB allocation thanks to the AB.

We observed through simulations using the WS-TFRA environment that the proposed scheduler outperforms the heuristics implemented in WS in terms of number of lost packets, although the reward provided in WS is not perfectly suited to this task.

We have also shown the importance of the adaptive action masking during the inference phase to prevent the scheduler from selecting empty buffers and thus non-optimal action.

Finally we have observed that the trained architecture generalizes over different inter arrival rates for non-GBR traffic.

There are few issues concerning the generalization over the inter-arrival rate, where the proposed EOT-AB scheduler performs poorly for some rare episodes. Additionally, the reward design in WS-TFRA is questionable, as similar average rewards can correspond to significantly different results in communication metrics like BO and PDB violations. A high reward does not necessarily reflect better performance for these metrics.

In future work, it could be of interest to improve the proposed architecture, for instance, by replacing the linear embedding and projections of Q-values with a DNN. Moreover, it would be valuable to train with variable arrival rates to enhance the architecture's reliability. In addition, detecting when the architecture begins to deviate and switching to an heuristic could further improve robustness.

Part of the material developed in this chapter had been patented in [85] and published in [42].

Chapter 4

Joint scheduling and MCS selection

4.1 Introduction

This chapter addresses the problem of joint frame-based scheduling and MCS selection. Specifically, we assume that scheduling decisions are made at the beginning of each frame, and we aim to jointly allocate each RB of the frame to a transmission buffer, and to select an appropriate MCS for packet transmission.

In Chapter 2 and in Chapter 3, we assumed an ideal error-free channel for which the capacity is perfectly known, which does not reflect real-world conditions. This chapter considers a more realistic setup where transmission errors at the receiver may occur and where only statistical CSI is known by the scheduler. We assume that the packets are transmitted over a Rayleigh flat fading channel.

The main contributions of this chapter are the proposal and comparison of three different solutions elaborated along two approaches: 1) one involving joint selection of the buffer and the MCS, referred to as joint scheduling and MCS (JSM), 2) the other one involving separately selections of the buffer and the MCS referred to as disjoint scheduling and MCS (DSM).

The three solutions can be synthesized as follows:

- The JSM solution, based on DRL, adapting the EOT-AB proposed in Chapter 3.
- A fully heuristic-based DSM solution, where scheduling and MCS selection are performed using heuristics. Introducing the acronym heuristic for the buffer selection (HBS), this approach is denoted as DSM-HBS.
- A hybrid DSM solution, using the EOT-AB proposed in Chapter 3 for buffer selection and a heuristic for the MCS selection. Introducing the acronym learning for buffer selection (LBS), this approach is denoted as DSM-LBS.

This chapter is organized as follows. Section 4.2 introduces the system model. Section 4.3 provides the MDP formulation of this problem. Section 4.4 presents the proposed solutions to solve this MDP. Section 4.5 details the training and evaluation setups. Section 4.6 provides the simulation results. Finally, Section 4.7 draws concluding remarks.

4.2 System model

We consider the same communication model as in Section 2.2, with n_L active links and two types of traffic, a DC and a BE traffic.

In this chapter, each link is characterized by its average SNR denoted by $\bar{\Gamma}$. We assume that the average SNR in dB is uniformly distributed among the links, in the range $[\bar{\Gamma}^{\mathrm{dB}}_{\mathrm{min}}, \bar{\Gamma}^{\mathrm{dB}}_{\mathrm{max}}]$. The channel

may induce transmission errors depending on $\bar{\Gamma}$ and the selected MCS. The available MCSs are noted MCS-μ, where $\mu \in \{1, \dots \mathcal{M}\}$ and $\mathcal{M}$ is the total number of available MCSs. Each MCS enables the extraction of a different integer number of packets, noted ρ_μ, from the buffer. We assume that $\rho_1 < \rho_2 < \cdots < \rho_\mathcal{M}$. The channel model is detailed in Section 4.2.2.

The joint scheduling and MCS selection is performed at each frame over a total bandwidth W divided into N_f RBs. Each RB and each link experiences different channel realizations.

4.2.1 Buffer model

The buffer model is described in Section 1.2 for the general model and in Section 2.2 for the DC and BE buffer model. We remind that $n_{i,k,1}$ is the number of packets in buffer i at the beginning of frame k i.e. the number of entries $d_{u,i}$ (index of the frame is omitted) whose value is greater than -1.

For each frame k, N_f buffers are selected for transmission and a MCS is assigned to the selected buffers. Let $\boldsymbol{i}_k = \{i_k^1, \dots, i_k^{N_f}\}$ be the selected buffers and $\boldsymbol{\mu}_k = \{\mu_k^1, \dots, \mu_k^{N_f}\}$ the corresponding MCS assignment. Note that a given buffer may be selected multiple times (i.e., $i_k^{j_1} = i_k^{j_2}$ with $j_1 \neq j_2$), and that different MCSs may be applied to the same buffer if selected multiple times. The scheduling and MCS selection at frame k yields a set of pairs $\{(i_k^1, \mu_k^1), \dots, (i_k^{N_f}, \mu_k^{N_f})\}$, where MCS-$\mu_k^j$ determines the number of packets $p_k^j = \rho_{\mu_k^j}$ that can be extracted from buffer i_k^j for the jth RB.

Then, the $n_{i,k,j}^t := \min(p_k^j, n_{i,k,j})$ oldest packets are extracted from this buffer and transmitted through the channel, using MCS-μ_k^j, where $n_{i,k,j}$ is the number of packets in the ith buffer before the transmission of the jth RB. The value of $n_{i,k,j}$ is updated at each RB as $n_{i,k,j+1} = n_{i,k,j} - n_{i,k,j}^t$. The bits of the $n_{i,k,j}^t$ packets are jointly interleaved, encoded and modulated producing a codeword (CW). This CW is then modulated using MCS-μ_k^j, producing a constant number of modulated symbols, denoted by $\mathcal{B}_\mathcal{S}$, which are transmitted over the channel. The channel may induce errors at the receiver side. Let $n_{i,k,j}^{\mathrm{Tx}}$ be the number of packets lost due to transmission errors for RB j of frame k from buffer i, which is equal to $n_{i,k,j}^t$ in case of decoding failure and 0 otherwise.

We remind that at the end of the frame (i.e. $j = N_f$) the WT of all the packets is incremented and may lead to DV. Then, new packets arrive in the buffers according to a Poisson distribution and BO may occur.

Note that all the packets can be sent without using all the RBs. Let n_k^u be the number of unused RBs for the frame k.

4.2.2 Channel model

We assume that the transmitted modulated CW $\boldsymbol{x}$ experiments a flat-fading Rayleigh channel $h_\ell(k,j) \sim \mathcal{CN}(0, 2\sigma_\ell^2)$.

The channel varies between each RB, and the received signal is corrupted by an additive white Gaussian noise (AWGN) with variance $2\sigma_n^2$ at the receiver level. The received signal on the ℓth link at the jth RB of frame k can be expressed as:

$$\boldsymbol{y}_\ell(k,j) = h_\ell(k,j)\boldsymbol{x} + \boldsymbol{\nu}_\ell(k,j), \tag{4.1}$$

where $\boldsymbol{\nu}_\ell(k,j)$ is the realization of the complex AWGN. We assume that the channel varies independently between successive RBs and frames.

Let $\Gamma_\ell(k,j)$ be the instantaneous SNR experimented for link ℓ at frame k on RB j which can be written as:

$$\Gamma_\ell(k,j) = \frac{|h_\ell(k,j)|^2}{2\sigma_n^2}. \tag{4.2}$$

The corresponding instantaneous PER is derived from a LUT, which maps each selected MCS and instantaneous SNR value to a PER estimate, based on AWGN performance. This mapping is denoted by $q_\mu(\Gamma_\ell(k,j))$, where μ is the index of the selected MCS.

We assume in this Chapter that Γ_ℓ is not available at the scheduler, instead, MCS selection is performed based on the average SNR which is defined as:

$$\bar{\Gamma}_\ell = \mathbb{E}\left[\Gamma_\ell(k,j)\right]. \tag{4.3}$$

In this context, computing the average PER is essential to enable a suitable MCS selection.

4.2.2.1 Procedure for the average PER

The average PER can be written as:

$$\bar{q}_\mu(\bar{\Gamma}_\ell) := \mathbb{E}_h[q_\mu(\Gamma_\ell)], \tag{4.4}$$

where the expectation is taken over the channel realizations.

Eq. (4.4) can be rewritten as:

$$\bar{q}_\mu(\bar{\Gamma}_\ell) = \int_0^{+\infty} q_\mu(u) p_{\Gamma_\ell}(u) du, \tag{4.5}$$

where p_{Γ_ℓ} corresponds to the probability density function (pdf) of the SNR on link ℓ. The LUT curve is often provided in a dB scale rather than in a linear one. In this case, we have $q_\mu^{\mathrm{dB}}(v)$ where $u = 10^{v/10}$ instead of $q_\mu(u)$. Therefore, we aim to rewrite (4.5) using $q_\mu^{\mathrm{dB}}(v)$. Since $du = \frac{\log(10)}{10} 10^{v/10} dv$, (4.5) becomes:

$$\bar{q}_\mu(\bar{\Gamma}_\ell) = \frac{\log(10)}{10} \int_{-\infty}^{+\infty} q_\mu^{\mathrm{dB}}(v) p_{\Gamma_\ell}(10^{v/10}) 10^{v/10} dv. \tag{4.6}$$

As noticed in [86], previous expression is computed over a restricted range of SNR, corresponding to the SNR of the LUT, denoted by $[\mathrm{SNR}_{\min}^{\mathrm{dB,LUT}}; \mathrm{SNR}_{\max}^{\mathrm{dB,LUT}}]$ and (4.6) is approached by:

$$\bar{q}_\mu(\bar{\Gamma}_\ell) \approx \frac{\log(10)}{10} \int_{\mathrm{SNR}_{\min}^{\mathrm{dB,LUT}}}^{\mathrm{SNR}_{\max}^{\mathrm{dB,LUT}}} q_\mu^{\mathrm{dB}}(v) p_{\Gamma_\ell}(10^{v/10}) 10^{v/10} dv. \tag{4.7}$$

Since the LUT is discrete, the integral in (4.6) is approximated by a sum, as in [86]. Specifically, the SNR values are uniformly sampled in dB as $v = [v_0, v_1, \ldots, v_{N_{\mathrm{sample}}-1}]$, with a constant step size Δ_v between consecutive samples, where $v_0 = \mathrm{SNR}_{\min}^{\mathrm{dB}}$ and $v_{N_{\mathrm{sample}}-1} = \mathrm{SNR}_{\max}^{\mathrm{dB,LUT}}$.

$$\bar{q}_\mu(\bar{\Gamma}_\ell) \approx \frac{\log(10)}{10} \Delta_v \sum_{i=0}^{N_{\mathrm{sample}}-1} q_\mu^{\mathrm{dB}}(v_i) 10^{v_i/10} p_{\Gamma_\ell}(10^{v_i/10}), \tag{4.8}$$

where Δ_v is the discretization step in the dB domain. Equation (4.8) holds for any channel model, i.e., for any distribution p_{Γ_ℓ}.

For Rayleigh flat-fading channel, the probability to have a given instantaneous SNR $\Gamma_\ell(k)$ is:

$$p_{\Gamma_\ell}(u) = \frac{1}{\bar{\Gamma}_\ell} \exp\left(-u/\bar{\Gamma}_\ell\right). \tag{4.9}$$

Inserting (4.9) in (4.8) gives us:

$$\bar{q}_\mu(\bar{\Gamma}_\ell) \approx \frac{\log(10)}{10} \frac{1}{\bar{\Gamma}_\ell} \Delta_v \sum_{i=0}^{N_{\text{sample}}-1} q_\mu^{\text{dB}}(v_i) 10^{v_i/10} \exp(-10^{v_i/10}/\bar{\Gamma}_\ell). \tag{4.10}$$

This approach can be extended to frequency selective channel by using effective SNR mapping (ESM) approach such as in [86].

4.3 Problem formulation for JSM and DSM approaches

Three types of packet loss can occur in the system described in Section 4.2: 1) BO losses, 2) DV losses for DC traffic, and 3) losses due to the propagation channel.

In Section 2.3, we showed that under system model with only losses due to BO and DV, the problem can be formulated as an MDP. In Section 3.2, we showed that frame-based scheduling can be also cast as an MDP, where the action is the simultaneous allocation of all the RBs. In this chapter, the losses due to channel and the selection of a proper MCS are also considered. One can prove that the Markov property is preserved in both cases: the selected MCS influences the number of packets extracted, and losses due to the propagation channel depend on the average SNR and the chosen MCS, i.e. the current state and action.

We propose an MDP formulation of the packet loss minimization problem for such system, introducing the state spaces, the action spaces and the reward.

4.3.1 State spaces

As for Section 2.3.2, let us note $\mathbb{S}_\ell$ the set of all the possible states for link ℓ. Since the state of each link ℓ is composed of the state of the DC buffer $\mathbb{S}_\ell^{\text{DC}}$, the state of the BE buffer $\mathbb{S}_\ell^{\text{BE}}$ and the state related to the channel information $\mathbb{S}_\ell^{\text{PER}}$, then $\mathbb{S}_\ell = \mathbb{S}_\ell^{\text{DC}} \times \mathbb{S}_\ell^{\text{BE}} \times \mathbb{S}_\ell^{\text{PER}}$. The considered sets $\mathbb{S}_\ell^{\text{DC}}$ and $\mathbb{S}_\ell^{\text{BE}}$ are the same as in Section 2.3.2, whereas the set $\mathbb{S}_\ell^{\text{PER}}$ is different from the set $\mathbb{S}_\ell^{\text{capa}}$ defined in Section 2.3.2.

The state vector $\mathbf{f}_\ell \in \mathbb{S}_\ell$ can be expressed as:

$$\mathbf{f}_\ell = \left[\mathbf{f}_\ell^{\text{DC-}x}, \mathbf{f}_\ell^{\text{BE}}, \mathbf{f}_\ell^{\text{PER}}\right], \tag{4.11}$$

where:

- $\mathbf{f}_\ell^{\text{DC-}x} \in \mathbb{S}_\ell^{\text{DC}}$ can be represented with the three different representations (HoL, xHoL and APD) defined in Sections 2.3.2.1, 2.3.2.2 and 2.3.2.3 respectively.
- $\mathbf{f}_\ell^{\text{BE}} \in \mathbb{S}_\ell^{\text{BE}}$ is the number of packets in the BE buffer, normalized by the buffer size such as in Section 2.3.2.
- $\mathbf{f}_\ell^{\text{PER}} \in \mathbb{S}_\ell^{\text{PER}}$ is defined as:

$$\mathbf{f}_\ell^{\text{PER}} = \left[\frac{\log\left(\bar{q}_1(\bar{\Gamma}_\ell)\right)}{\log\left(\bar{q}_{\min}\right)}, \ldots, \frac{\log(\bar{q}_{\mathcal{M}}(\bar{\Gamma}_\ell))}{\log(\bar{q}_{\min})}\right], \tag{4.12}$$

where $\bar{q}_{\min}$ is the lowest average PER achievable in the system, according to the available MCS and the SNR range. $\bar{q}_{\min}$ is thus defined as:

$$\bar{q}_{\min} := \min_{\mu \in \{1,\dots,\mathcal{M}\}} \bar{q}_{\mu}(\bar{\Gamma}_{\max}), \tag{4.13}$$

where $\bar{\Gamma}_{\max} = 10^{\bar{\Gamma}^{\mathrm{dB}}_{\max}/10}$ is the highest average SNR in linear considered in the system model in Section 4.2. The logarithm in (4.12) helps the DNN to differentiate between orders of magnitude in the PER. The entries of $\mathbf{f}_{\ell}^{\mathrm{PER}}$ are in $[0, 1]$. We consider that $\mathbf{f}_{\ell}^{\mathrm{PER}}$ remains constant over the time. Hence, the considered state spaces follow the Markovian property.

Let us note n_{feature} the number of features characterizing the different state spaces, where:

- $n_{\mathrm{feature}} = 3 + \mathcal{M}$ for HoL representation.
- $n_{\mathrm{feature}} = 4 + \mathcal{M}$ for xHoL representation.
- $n_{\mathrm{feature}} = D_0 + 3 + \mathcal{M}$ for APD representation.

Notice that the PER information per link is required at the input of both JSM and DSM-LBS architectures. Therefore, one might guess that:

- In the JSM solution, the PER information should be fully used by the architecture to jointly select a buffer along with a suitable MCS.
- In the DSM-LBS solution, although the MCS is selected by an heuristic, the architecture may use this information to guess the selected MCS, thanks to the training, and thus adapt the buffer selection accordingly.

It is worth noting that other state representations could be considered instead of $\mathbb{S}_{\ell}^{\mathrm{PER}}$ for the DSM approach. For example, one may include the MCS chosen by the heuristic for the MCS selection (HMS) along with its associated PER.

4.3.2 Action spaces for both JSM and DSM approaches

The proposed scheduler aims to jointly select the buffer to be served and the associated MCS for each RB. For the JSM approach, the action can be written as $\boldsymbol{a}_k := \{a_k^1, \dots, a_k^{N_f}\}$, where a_k^j corresponds to a pair (i_k^j, μ_k^j). For the DSM-LBS the action can be written as $\boldsymbol{a}_k := \{i_k^1, \dots, i_k^{N_f}\}$.

4.3.3 Reward

The proposed scheduler aims to mitigate the total number of lost packets. The reward is computed at the end of the frame and is defined by:

$$r_k := e^{\omega\left(\alpha_1 \mathfrak{R}_{d,k} + \alpha_2 \mathfrak{R}_{o,k} + \alpha_3 \mathfrak{R}_{c,k}\right)}, \tag{4.14}$$

where

$$\mathfrak{R}_{d,k} := -\sum_{\ell=0}^{n_L-1} n_{2\ell,k}^{d} \tag{4.15}$$

$$\mathfrak{R}_{o,k} := -\sum_{i=0}^{n_Q-1} n_{i,k}^{o} \tag{4.16}$$

$$\mathfrak{R}_{c,k} := -\sum_{j=1}^{N_f} \sum_{i=0}^{n_Q-1} n_{i,k,j}^{\mathrm{Tx}}. \tag{4.17}$$

correspond to the opposite of the number of lost packets due to DV, to BO and to the propagation channel, respectively, for frame k. $\omega > 0$ is a hyperparameter controlling the behavior of the exponential function, and α_1, α_2 and α_3 are positive hyperparameters controlling the importance of the different losses, with $\sum_{j=1}^{3} \alpha_j = 1$. The exponential maps the reward in $]0, 1]$.

4.4 Problem solutions

4.4.1 Solution approaches for MCS and buffer selections

The three different approaches proposed in this chapter are the following ones:

1. The JSM solution, performing joint scheduling and MCS selection with statistical CSI, leveraging also the EOT-AB architecture.
2. The DSM-HBS solution, fully based on heuristics. Both the MCSs and the buffers are selected by two different heuristics.
3. The DSM-LBS solution, where the MCSs are selected by a heuristic and the buffers are selected according to a DRL-scheduler, leveraging the EOT-AB architecture proposed in Section 3.3.

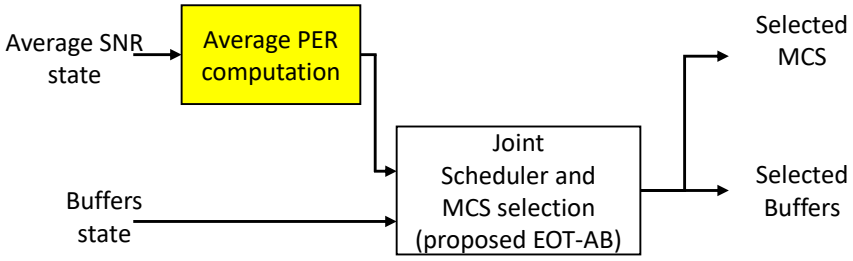
Table 4.1 classifies the different scheduling and MCS selection solutions considered in this thesis.

Table 4.1: Considered scheduling and MCS selection solutions.

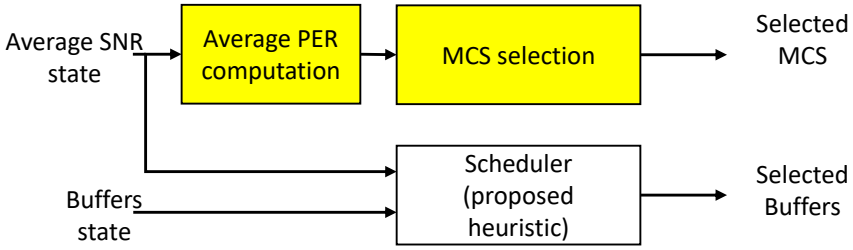
		Scheduling	
		Heuristic	DRL
MCS selection	Heuristic	DSM-HBS	DSM-LBS
	DRL	Not addressed	JSM

One can notice that in Table 4.1, the combination of HBS with DRL-based MCS selection has not been addressed. We were unable to evaluate this configuration within the timeframe available for this thesis.

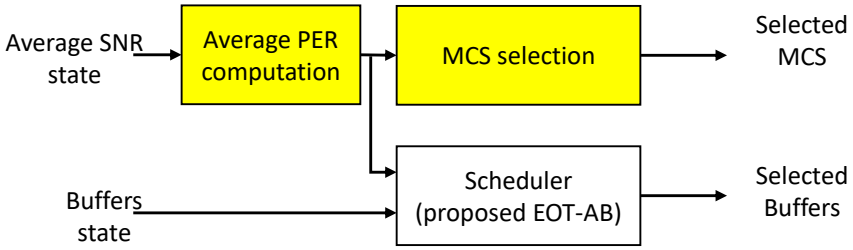
The different solutions listed in Table 4.1 are illustrated in Figure 4.1.



(a) Joint scheduling and MCS selection (JSM).



(b) Disjoint scheduling and MCS selection (DSM-HBS).



(c) Disjoint scheduling and MCS selection (DSM-LBS).

Figure 4.1: Scheduling and MCS selection with different methods. Yellow boxes represent operations that are done once per average SNR trial.

These different solutions can be explained as follows:

- The method in Figure 4.1a illustrates the JSM solution, where DRL is employed for both scheduling and MCS selection. The average PER is computed and is fed into the EOT-AB that outputs the Q-value for each buffer-MCS pairs.
- The method in Figure 4.1b illustrates the DSM-HBS solution, where the average SNR is used for both the average PER computation and the scheduler.
- The method in Figure 4.1c illustrates the DSM-LBS solution, where the average SNR is used to compute the average PER only. The average PER is used as state space, such as in Section 4.3.1, for DRL scheduler which is in our case an EOT-AB. The EOT-AB outputs the Q-value for each buffer.

One can note that the EOT-AB used in JSM and DSM-LBS are slightly different since they do not have the same number of outputs.

The HMS used for the DSM solutions is as follows: 1) we compute the average PER (4.4) $\bar{q}_\mu(\bar{\Gamma}_\ell)$ for $\mu \in \{1, \ldots, \mathcal{M}\}$. 2) we select $\mu^{\mathrm{HMS}} = \arg\max_{\mu \in \{1,\ldots,\mathcal{M}\}} \{\rho_\mu \mid \bar{q}_\mu(\bar{\Gamma}_\ell) \leq \bar{q}_{\mathrm{th}}\}$. This corresponds to the MCS that allows to transmit the largest number of packets while satisfying a PER threshold condition.

4.4.2 Adaptation of the heuristics for buffer selection

In this section, we present the proposed adaptations for the different HBSs, introduced in Section 1.3 in order to account for the frame-based and MCS selection context:

1. PF, MLWDF, and LOG-rule require the instantaneous rate. Since this information is not available, for the sake of simplicity, we propose to replace it using this expression:

$$c_i = \log_2\left(1 + \bar{\Gamma}_{\ell_i}\right). \tag{4.18}$$

2. According to Sections 1.3.4 and 1.3.5, we must distinguish between DC and BE traffic; therefore, MLWDF and LOG-rule should be modified accordingly.
3. The KP presented in Section 1.3.7 requires a PLR target and a bearer priority, which are not defined in this chapter. Consequently, this method also requires adaptation.

Remark: In this section, as for Section 3.4.4, to simplify the notation, the index k denotes the current combination of slot/RB, instead of using separate indices for slots and RBs.

4.4.2.1 Round-robin

The RR does not depend on the channel and thus does not require adaptation.

4.4.2.2 Proportional fair

The PF uses the instantaneous rate information for each buffer i, noted c_i (for link ℓ_i).

The PF expression is thus given by (1.13):

$$h_{\mathrm{PF}}(\mathbf{x}_i) = \frac{c_i}{\bar{c}_i} = \frac{\log_2\left(1 + \bar{\Gamma}_{\ell_i}\right)}{\bar{c}_i}, \tag{4.19}$$

where $\mathbf{x}_i$ is the feature vectors of buffer i, and as a reminder

$$\bar{c}_i = \frac{1}{KN_f}\sum_{k=1}^{K}\sum_{j=1}^{N_f} c_i \delta_{j,k}(i), \tag{4.20}$$

with

$$\delta_{j,k}(i) = \begin{cases} 1 & \text{if } i \text{ is selected for RB } j \text{ of frame } k \\ 0 & \text{otherwise} \end{cases}. \tag{4.21}$$

4.4.2.3 MLWDF

The MLWDF uses also c_i.Thus, we propose the same approach as for PF and to use (4.18) for c_i. The MLWDF expression given by (2.14) is transformed as:

$$h_{\mathrm{MLWDF}}(\mathbf{x}_i) := \begin{cases} \alpha_i c_i d_{0,i} & \text{if } i \mod 2 = 0 \text{ (if } i \text{ is a DC buffer)} \\ h_{\mathrm{PF}}(\mathbf{x}_i) & \text{if } i \mod 2 = 1 \text{ (} i \text{ is a BE buffer)} \end{cases}, \tag{4.22}$$

where α_i is set as in Section 1.3.

4.4.2.4 LOG-rule

The LOG-rule uses also c_i.Thus, we propose the same approach as for PF and to use (4.18) for c_i. The LOG-rule expression given by (2.15) is transformed as:

$$h_{\mathrm{LOG-rule}}(\mathbf{x}_i) := \begin{cases} \alpha_i c_i \log\left(\beta_i + \eta_i d_{0,i}\right) & \text{if } i \mod 2 = 0 \text{ (if } i \text{ is a DC buffer)} \\ h_{\mathrm{PF}}(\mathbf{x}_i) & \text{if } i \mod 2 = 1 \text{ (} i \text{ is a BE buffer)} \end{cases}, \tag{4.23}$$

where α_i, β_i and η_i are set as in Section 1.3.

4.4.2.5 Knapsack

Since a PLR target is not defined for the different traffic types and the CQI is unavailable, and because it may represent a channel metric as discussed in Section 1.3.7, we adapt (1.25) using the average SNR, normalized by $\bar{\Gamma}^{\mathrm{dB}}_{\mathrm{max}}$

$$v_2(\mathbf{x}_{i,k}) := \frac{\bar{\Gamma}^{\mathrm{dB}}_i}{\bar{\Gamma}^{\mathrm{dB}}_{\mathrm{max}}}. \tag{4.24}$$

The new expression of KP adapted to the statistical CSI context is given by (1.23) along with (1.24), (4.24), (2.18), and (1.27).

It is worth noting that for BE buffers, (1.24) is equal to 0. Because KP expression is a sum (1.23), it does not imply that $h_{\mathrm{KP}}(\mathbf{x}_{i,k}) = 0$, conversely to the MLWDF expression (1.19) when $\eta_i = 0$. Thus, there is no need to differentiate between DC and BE traffic, as is the case with MLWDF.

4.4.3 Adaptation of the EOT-AB for JSM and DSM-LBS approaches

As in Chapter 3, we use the EOT-AB architecture that is adapted to the JSM and DSM-LBS approaches. It is illustrated in Figure 4.2.

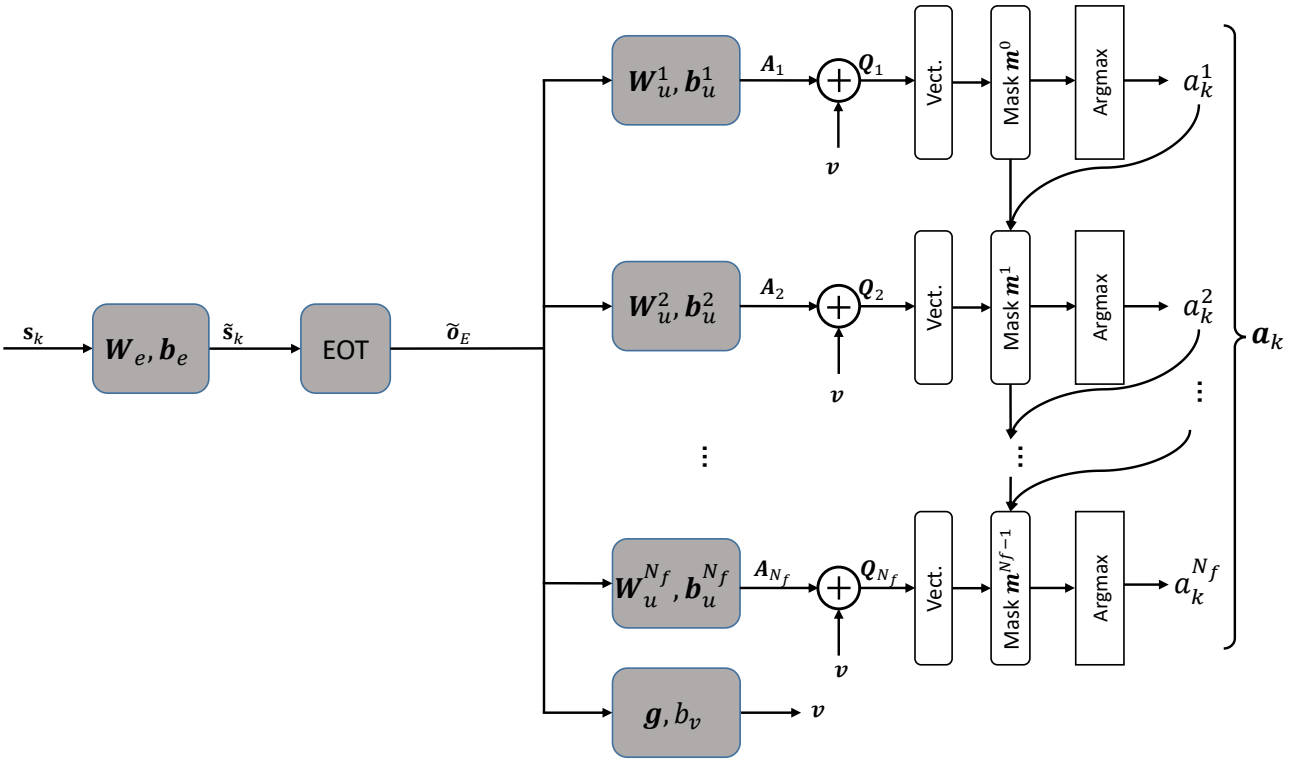


Figure 4.2: EOT-AB architecture adapted to JSM and DSM-LBS approaches. Gray boxes are learn.

The EOT-AB works as follows:

1. First, the state space, $\mathbf{s}_k$ undergoes a linear embedding using the $d_e \times n_{\text{feature}}$ matrix $\boldsymbol{W}_e$. The resulting embedded state is $\tilde{\mathbf{s}}_k := \boldsymbol{W}_e \mathbf{s}_k + \boldsymbol{b}_e \mathbf{1}_{n_L}^T$, with dimensions $d_e \times n_L$, where $\boldsymbol{b}_e$ is the additional bias vector of dimension $d_e \times 1$, and $\mathbf{1}_{n_L}$ is a vector of ones of dimensions $n_L \times 1$.
2. Next, $\tilde{\mathbf{s}}_k$ is processed by an EOT module which outputs a $d_e \times n_L$ matrix, denoted by $\tilde{\boldsymbol{o}}_E$.
3. Then $\tilde{\boldsymbol{o}}_E$ is projected into advantages on all the branches and into value for the dueling.
   - The $1 \times n_L$ value vector is obtained by $\boldsymbol{v} := \boldsymbol{g}\tilde{\boldsymbol{o}}_E + b\mathbf{1}_{d_e}^T$ where the ℓth entry of $\boldsymbol{v}$ represents the estimated state value of the ℓth link, $\boldsymbol{g}$ is a $1 \times d_e$ weights vector and b_v is a scalar corresponding to the bias.
   - For the jth branch, the projection into advantages is performed:
     - For JSM approach with the $2\mathcal{M} \times d_e$ matrix $\boldsymbol{W}_u^j$, where the output is denoted as: $\boldsymbol{A}_j := \boldsymbol{W}_u^j \tilde{\boldsymbol{o}}_E + \boldsymbol{b}_u^j \mathbf{1}_{n_L}^T$, which has dimensions $2\mathcal{M} \times n_L$. b_u^j is the additional bias on the jth branch.
     - For DSM approach with the $2 \times d_e$ matrix $\boldsymbol{W}_u^j$,where the output is denoted as: $\boldsymbol{A}_j := \boldsymbol{W}_u^j \tilde{\boldsymbol{o}}_E + \boldsymbol{b}_u^j \mathbf{1}_{n_L}^T$, which has dimensions $2 \times n_L$. b_u^j is the additional bias on the jth branch.
4. The vector $\boldsymbol{v}$ is then added to the advantage on each branch j to produce the matrix $\boldsymbol{Q}_j$, i.e. $\boldsymbol{Q}_j := \boldsymbol{A}_j + \mathbf{1}_{2\mathcal{M}} \boldsymbol{v}$, where the ℓth column is an estimate of the Q-value for the different possible actions for the ℓth link.
   - For JSM approach, the ℓth column contains $2\mathcal{M}$ entries where the first (resp. last) $\mathcal{M}$ entries are the estimated Q-values corresponding to each MCS for the DC (resp. BE) buffer. The corresponding MCS is given by the modulo.
   - For DSM approach, the ℓth column contains 2 entries where the first (resp. second) entry is the estimated Q-values for the DC (resp. BE) buffer.

To prevent from selecting an empty buffer, adaptive action masking is applied on each branch.

5. Finally, the action a_k^j is selected by taking the $\arg\max$ on the masked flattened matrix $\boldsymbol{Q}_j$.
   - For JSM approach, the selected buffer and associated MCS are retrieved as $i_k^j = \left\lfloor \frac{a_k^j}{\mathcal{M}} \right\rfloor$ and $\mu_k^j = a_k^j \bmod \mathcal{M}$ respectively.
   - For the DSM-LBS approach, the action corresponds to the scheduling, i.e. $i_k^j = a_k^j$ and the MCS is chosen according to the procedure discussed in Section 4.3.2.

The differences between this architecture and the one from Section 3.3 reside in the outputs. In Section 3.3, only buffer selection is performed whereas in this chapter, a MCS is also selected.

Notice that there exist cases where a selected buffer i_k^j and its corresponding MCS, MCS-μ_k^j, verify $n_{i_k^j} < \rho_{\mu_k^j}$. Hence, the MCS is over-dimensioned compared to the number of packets to be transmitted, and the RB is not fully used. In that case, it would be more convenient to use a MCS-$\hat{\mu}$, with $\hat{\mu} < \mu_k^j$, verifying $\rho_{\hat{\mu}} = n_{i_k^j}$, ensuring that the average PER is smaller than the one achieved using MCS-μ_k^j. One may consider such a trick in future works to improve performance.

Table 4.2 gives the complexity of this architecture.

Table 4.2: Number of FLOPs of the EOT-AB for JSM.

Operation	Number of FLOPs
Embedding $\boldsymbol{W}_e$	$2 \times d_e \times n_{\text{feature}} \times n_L$
EOT (from Table A.1)	$n_L(2d_e(4d_e + 2n_L + 2d_{\text{mlp}} + 11) + 4Hn_L + 2d_{\text{mlp}})$
Dueling	$2 \times d_e \times n_L$
Projection into advantages	$2 \times 2\mathcal{M} \times d_e \times n_L \times N_f$
Q-values (advantage plus value)	$2\mathcal{M}n_L \times N_f$
Total	$2n_L\left[d_e\left(n_{\text{feature}} + 4d_e + 2n_L + 2d_{\text{mlp}} + 10 + 2\mathcal{M}N_f\right)\right.$ $\left. 2Hn_L + \mathcal{M}N_f\right] - n_L^2 - n_L d_e$

4.5 Performance evaluation

In this section, we compare the performance of the different solutions referenced in Table 4.1.

For the DSM-HBS solutions, we use the heuristics presented in Section 4.4.2. We note "DSM-HBS h" the DSM-HBS solution using heuristic "h"$\in \{\text{RR}, \text{MLWDF}, \text{LOG-rule}, \text{KP}\}$.

For the DRL solutions (JSM and DSM-LBS), we use the three state space representations depicted in Section 4.3.1: HoL, xHoL and APD. We note "JSM s" and "DSM-LBS s" the JSM and DSM-LBS solutions using state space representation "s"$\in \{\text{HoL}, \text{xHoL}, \text{APD}\}$.

The MCS used for the DSM methods is chosen according to the procedure described in Section 4.4.1.

4.5.1 Simulation settings

At the beginning of each episode, in both training and inference, the average SNR of each link is drawn uniformly at random from the interval $[\bar{\Gamma}_{\min}^{\text{dB}}, \bar{\Gamma}_{\max}^{\text{dB}}]$, and is considered to be constant for the episode duration. Then the average PER is computed for each link thanks to (4.8). Table 4.3 details the simulation parameters.

Table 4.3: Simulation system model parameters.

D	20
B	40
N_f	5
q_{th}	10^{-2}
$\bar{\Gamma}^{\text{dB}}_{\text{min}}$	20 dB
$\bar{\Gamma}^{\text{dB}}_{\text{max}}$	40 dB
$\mathcal{B}$	1000 data bits per packet
$\mathcal{B}_{\mathcal{S}}$	1000 symbols per RB

Table 4.4 details the available MCSs which are implemented along with a low-density parity check (LDPC) code.

Table 4.4: Available MCSs.

MCS index μ	1	2	3
Modulation	QPSK	16QAM	64QAM
Coding rate	$1/2$	$1/2$	$2/3$
Number of encoded bits	1000	2000	3000
Number of transmitted packets per RB (ρ_μ)	1	2	3

Figure 4.3 illustrates the average PER over a Rayleigh flat fading channel for the considered MCSs. For $\bar{q}_{\text{th}} = 10^{-2}$, which corresponds to the threshold for both HBS and DSM-LBS methods, the following MCSs are applied based on the average SNR:

- QPSK $1/2$ is selected from 20 dB up to approximately 27 dB.
- 16QAM $1/2$ is then used from 27 dB to around 34 dB.
- 64QAM $2/3$ is applied from 34 dB to 40 dB.

For $\bar{q}_{\text{th}} = 10^{-1}$, the MCS selection changes:

- QPSK $1/2$ is never used.
- 16QAM $1/2$ is used between 20 dB and about 23.5 dB.
- 64QAM $2/3$ is employed from 23.5 dB to 40 dB.

As stated in Table 4.3, all the DSM-HBS solutions have been evaluated with $q_{\text{th}} = 10^{-2}$. We observed that DSM-HBS KP outperforms all other heuristics. Therefore, we tried the DSM-HBS KP with $q_{\text{th}} = 10^{-1}$. Indeed, in that case, we select more often MCSs with higher ρ_μ, thus draining faster the buffers. Thus, we can expect better performance in terms of total PLR for high value of Λ. Let us note KP1 (resp. KP2) the DSM-HBS KP using the PER threshold equal to 10^{-1} (resp. 10^{-2}).

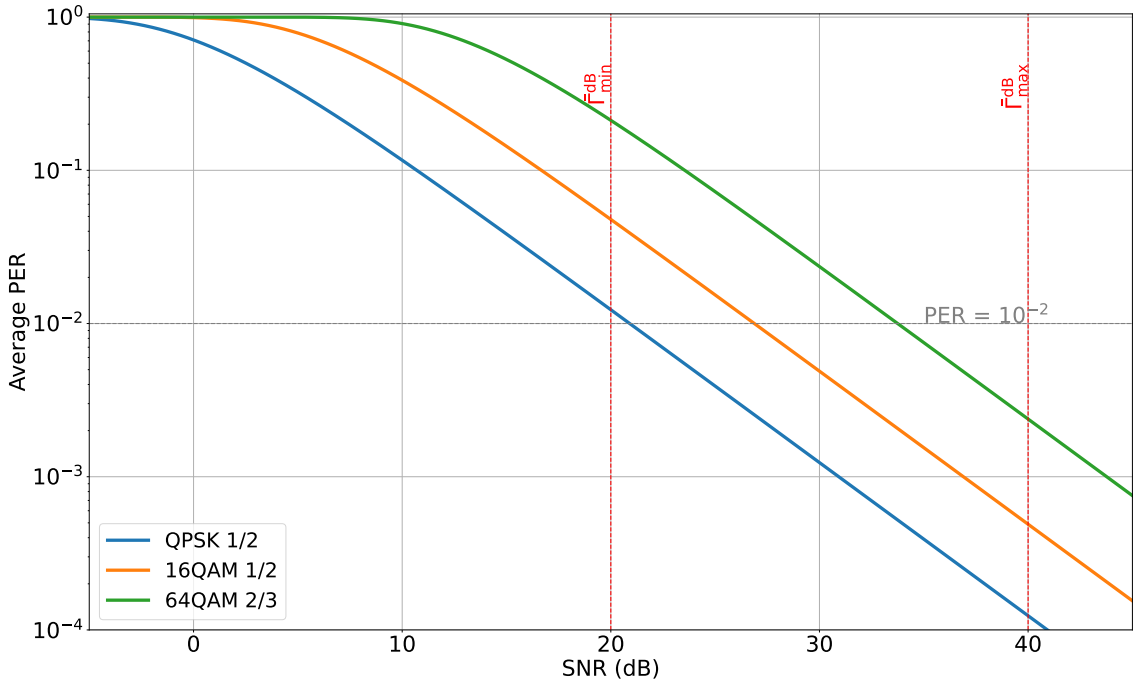


Figure 4.3: Average PER for the considered MCSs on Rayleigh flat-fading channel (red vertical bars represent the lower and upper bounds for the average SNR trials).

Table 4.5 details the EOT-AB hyperparameters.

Table 4.5: EOT-AB hyperparameters.

Parameter	Numerical value
γ	0.95
τ (soft update parameter)	0.005
Target network update period	20 steps
Batch size	512
Learning rate	5×10^{-4}
ϵ_{start}	1
ϵ_{end}	0.01
ϵ_{decay}	0.99
Number of heads H	4
Number of EOT layer	1
$d_e = d_{\text{mlp}} = Hd_{\text{attn}}$	256

Table 4.6 shows the complexity in FLOPs computed for the JSM and DSM-LBS solutions along with the EOT used in Chapter 2.

Table 4.6: EOT-AB complexity for $n_L = 6$.

State space	FLOPs JSM	FLOPs DSM-LBS	FLOPs EOT (Chapter 2)
HoL	4 895 925	4 834,485	4 800 688
xHoL	4 898 997	4 837 557	4 803 760
APD	4 957 365	4 895 925	4 862 128

One can see that the complexity of the JSM solution is approximately 1% higher than the DSM-LBS solution, even if it also performs MCS selection. It is also less than 2% higher than the architecture used for the slot-based scheduling in Chapter 2, even if the architecture used for the JSM is used for a frame with N_f RBs instead of a single slot.

4.5.1.1 Training setup

The different architectures are trained during 4000 episodes of 7000 steps. The arrival rate for each buffer is set to $\lambda = \frac{\Lambda}{n_Q} N_f = \frac{1.6}{12} \times 5$. Every 100 episodes, a validation episode of 70 000 steps is performed. A validation episode is also performed when the current average training reward is higher than the previous average training rewards. This may lead to different number of validation episodes among the trained architectures. The weights used for the inference are the ones for which the highest validation reward has been obtained. This may lead to a selection bias since only one validation episode is performed, i.e. only one seed, and by extension one average SNR configuration is used for the weights selection. The used average SNR in dB for the validation are: $[\bar{\Gamma}_0^{\mathrm{dB}}, \ldots, \bar{\Gamma}_5^{\mathrm{dB}}] = [27.49, 39.01, 34.64, 31.97, 23.12, 23.12]$. In future works, multiple validation episodes, i.e. multiple seeds with distinct average SNR configuration, should be used to reduce the selection bias.

We also implemented the following trick for the training: when the total number of packet in the buffers is lower than N_f at the beginning of the frame, we fill the RBs using the most robust MCS. This trick enhances transmission reliability since it implements deterministic optimal actions according to our problem, thus minimizing the number of lost packets for the current frame. We suppose that such transitions are not valuable for training since the actions can be directly performed without using a DNN and therefore exclude them from the replay buffer.

4.5.1.2 Inference setup

We perform $n_{\mathrm{episode}} = 25$ inference episodes. Each episode, corresponding to different average SNR draws, is composed of $K = 1\,000\,000$ steps, each step corresponding to a frame of $N_f = 5$ RBs. We consider two types of inferences:

1. Inferences with the implementation trick used during the training: if there are less than N_f packets in total at the beginning of the frame, we fill directly the RB using the most robust MCS. This may decrease the energy efficiency (EE) but decrease the PER.
2. Inferences without the implementation trick used during the training: if there are fewer than N_f packets in total at the beginning of the frame, the selected MCS still depends on the scheduler and may not be the most robust one.

4.5.2 Performance Metrics

We evaluate the following metrics:

- The training and validation rewards.

- The PLR defined as the total number of lost packets divided by the total number of arrived packets in the buffers. The loss may come from DV, BO and channel error transmission i.e.:

$$\xi := \frac{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} \left(n_{i,k}^{o}(n_{\mathsf{ep}}) + n_{i,k}^{d}(n_{\mathsf{ep}}) + \sum_{j=1}^{N_f} n_{i,k,j}^{\mathrm{Tx}}(n_{\mathsf{ep}}) \right)}{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^{r}(n_{\mathsf{ep}})}. \tag{4.25}$$

- The PLR due to DV and BO only defined as the total number of lost packets due to DV and BO divided by the total number of arrived packets:

$$\xi_{\mathrm{DV+BO}} := \frac{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} \left(n_{i,k}^{o}(n_{\mathsf{ep}}) + n_{i,k}^{d}(n_{\mathsf{ep}}) \right)}{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^{r}(n_{\mathsf{ep}})}. \tag{4.26}$$

- The PLR due to DV only defined as the total number of lost packets due to DV divided by the total number of arrived packets:

$$\xi_{\mathrm{DV}} := \frac{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^{d}(n_{\mathsf{ep}})}{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^{r}(n_{\mathsf{ep}})}. \tag{4.27}$$

- The PLR due to BO only defined as the total number of lost packets due to BO divided by the total number of arrived packets:

$$\xi_{\mathrm{BO}} := \frac{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^{o}(n_{\mathsf{ep}})}{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} n_{i,k}^{r}(n_{\mathsf{ep}})}. \tag{4.28}$$

- The PLR due to channel which is here defined as the number of lost packets due to channel error transmission divided by the total number of transmitted packets, i.e.:

$$\zeta_{\mathrm{CH}} := \frac{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} \sum_{j=1}^{N_f} n_{i,k,j}^{\mathrm{Tx}}(n_{\mathrm{ep}})}{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} \sum_{j=1}^{N_f} n_{i,k,j}^{t}(n_{\mathsf{ep}})}. \tag{4.29}$$

- The throughput, defined as the number of correctly transmitted packets divided by the total number of frame, over all the episodes:

$$\eta := \frac{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} \sum_{j=1}^{N_f} \left(n_{i,k,j}^{t}(n_{\mathsf{ep}}) - n_{i,k,j}^{\mathrm{Tx}}(n_{\mathsf{ep}}) \right)}{n_{\mathsf{episode}} K}. \tag{4.30}$$

  It is worth noting that the number of packets sent cannot exceed the limit imposed by the highest MCS, which allows a maximum of 3 packets per RB. As a result, the throughput is capped at $3N_f = 15$.
- The proportion of unused RB in total, which might occur w when the selected MCSs empty all the buffers with a number of RBs strickly less than N_f, and which is defined by:

$$\mathcal{U} := \frac{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} n_{k}^{u}(n_{\mathsf{ep}})}{n_{\mathsf{episode}} K N_f}. \tag{4.31}$$

- The EE, which is defined as the number of correctly transmitted packets divided by the number of effective RBs:

$$\mathcal{E} := \frac{\sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} \sum_{i=0}^{n_Q-1} \sum_{j=1}^{N_f} \left(n_{i_k^j}^{t}(n_{\mathsf{ep}}) - n_{i,k,j}^{\mathrm{Tx}}(n_{\mathsf{ep}}) \right)}{\mathcal{P}\left(n_{\mathsf{episode}} K N_f - \sum_{n_{\mathsf{ep}}=1}^{n_{\mathsf{episode}}} \sum_{k=1}^{K} n_{k}^{u}(n_{\mathsf{ep}}) \right)}, \tag{4.32}$$

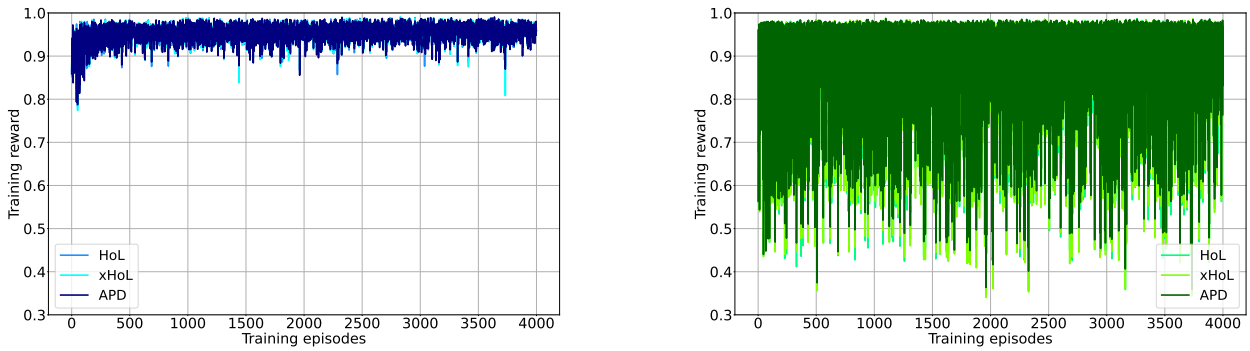
where $\mathcal{P}$ is the transmit power, which is assumed to be equal to one. It is important to note that, in general, EE formulations include circuit power consumption, particularly in EE maximization problems [1]. However, this is not considered here. In our case, excluding the power of circuitry allows for simpler interpretation: if $\mathcal{E} > \Lambda$, then not all available RBs are utilized, resulting in power saving and nearly all packets are transmitted. Otherwise, if $\mathcal{E} \leq \Lambda$, either nearly all RBs are used, not all packets are transmitted correctly, or both. If all the RBs are used for all the episodes, then the EE is proportional to the throughput.

4.6 Performance analysis

4.6.1 Training analysis

In this section, we analyze the evolution of the reward during the training and validation reward.

Figure 4.4 plots the training rewards for JSM and DSM architectures.



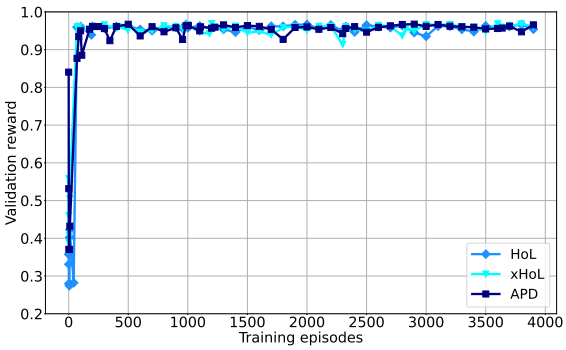
(a) Training rewards for JSM architectures.

(b) Training rewards for DSM architectures.

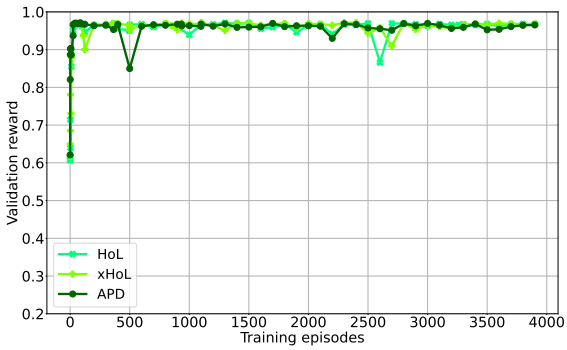
Figure 4.4: Training rewards for JSM and DSM architectures.

One can observe in Figure 4.4a that the JSM architectures have converged, and the training rewards have a low variance. In Figure 4.4b, one can observes that the DSM architectures have a training reward with higher variance than the JSM architectures. Due to the high variance of the DSM architectures, it seems difficult to conclude if they have converged or not. The difference in variance can be attributed to the following: when the average SNR is high, both the JSM and DSM approaches succeed in emptying the buffers with low packet loss, resulting in high reward values for both. However, when the average SNR is low, the JSM approach adapts the MCS for each RB to optimize buffer draining, thereby reducing the overall packet loss, yielding high reward values. In contrast, the DSM approach, which uses a fixed MCS for all the RBs, struggles to efficiently empty the buffers, leading to high packet losses due to DV and BO, and consequently lower reward values.

The validation reward is plotted in Figure 4.5 for both methods, as a function of the training episodes.A validation episode is performed every 100 training episodes, or whenever the average reward obtained at the current episode surpasses all previously observed training rewards. The markers indicate the episodes at which validation occurs.



(a) Validation rewards for JSM architectures.



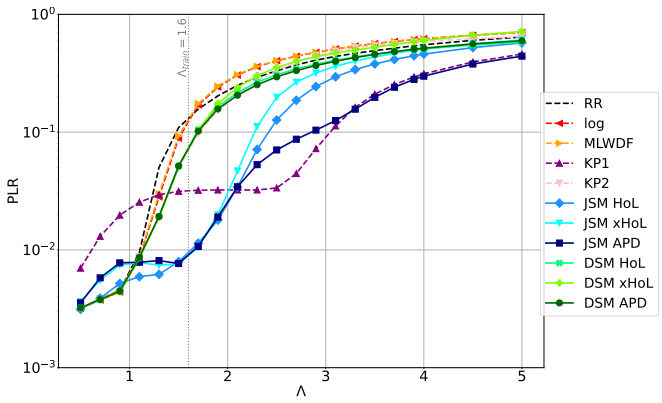
(b) Validation rewards for DSM architectures.

Figure 4.5: Validation rewards for JSM and DSM architectures.

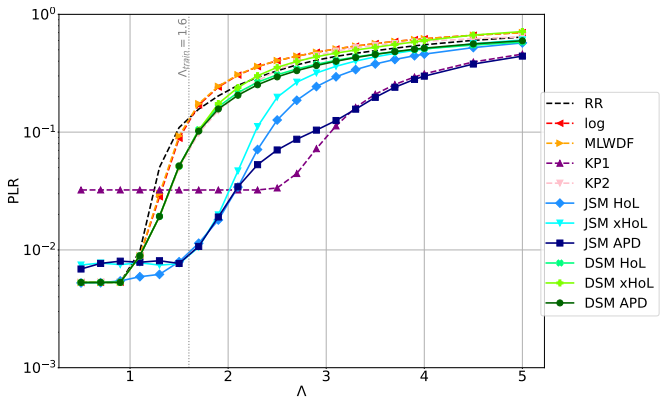
One can observe that for both methods, the validation rewards converge in less than 500 training episodes, to the average value of about 0.96. The two different methods seem to converge at the same speed despite the fact that the action space in Fig. 4.5a is larger than in Fig. 4.5b.

4.6.2 Inference performance analysis

Figure 4.6 plots the PLR (noted ξ) vs. Λ. Figure 4.6a plots the PLR from the first inference setup, which includes the implementation trick, while Figure 4.6b plots the PLR from the second setup, without the implementation trick.



(a) With the implementation trick.



(b) Without the implementation trick.

Figure 4.6: PLR vs. Λ.

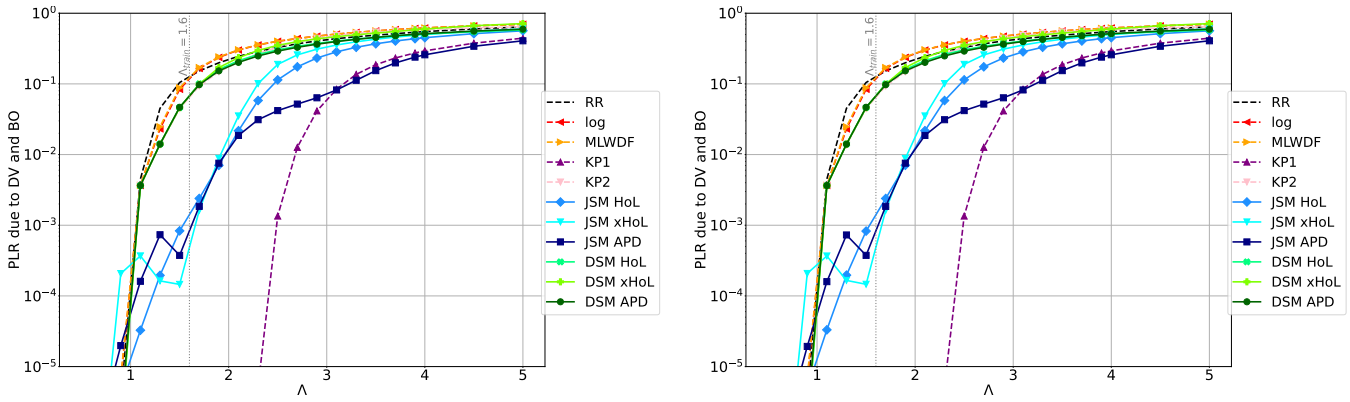
One can observe that:

- The two figures are nearly identical, except for $\Lambda < 1.2$, where the setup including the implementation trick achieves better PLR performance.
- Regardless of the value of arrival rates, the KP2 yields similar results as the DSM-LBS.
- When $\Lambda < 1$, the JSM architectures have a higher PLR than the DSM methods, which obtain approximately the same PLR. In this regime, the PLR is mostly due to channel transmission errors, see e.g. Figure 4.9 and Figure 4.7.

- For $1 \leq \Lambda \leq 2$, the proposed JSM solutions yield significant lower PLR compared to the DSM-HBS and the DSM-LBS solutions. It is noteworthy that a gain of approximately one order of magnitude is achieved compared to RR, MLWDF, and LOG-rule from $\Lambda = 1.5$, and relative to KP2 and the DSM-LBS from $\Lambda = 1.7$. This highlights the importance of performing MCS selection to adapt to varying traffic loads.
- KP1 has the higher PLR for $\Lambda \leq 1.2$. For $2 \leq \Lambda \leq 3.1$, it obtains better performance than the JSM architecture. A possible explanation is that this range of arrival rate is far from the training arrival rate, producing generalization issues. For $\Lambda \geq 3.1$, KP1 and JSM APD obtain similar results and beat the other methods. This confirms what was anticipated when defining KP1 in Section 4.5.1.
- For $\Lambda \geq 3.1$, all the methods have a PLR greater than 10^{-1}. For these arrival rates, the average number of arriving packets exceeds the maximum that can be transmitted using the highest MCS.

The above figures aggregate the losses due to DV, BO and channel providing a global view of the performance of the different methods. To better understand the underlying strategy of the different methods, let us study the loss from the different sources separately, that is, $\xi_{\mathrm{DV+BO}}$, ξ_{DV}, ξ_{BO}, and ζ_{CH}.

Figure 4.7 represents the PLR due to DV plus BO ($\xi_{\mathrm{DV+BO}}$) vs. Λ.



(a) With the implementation trick.

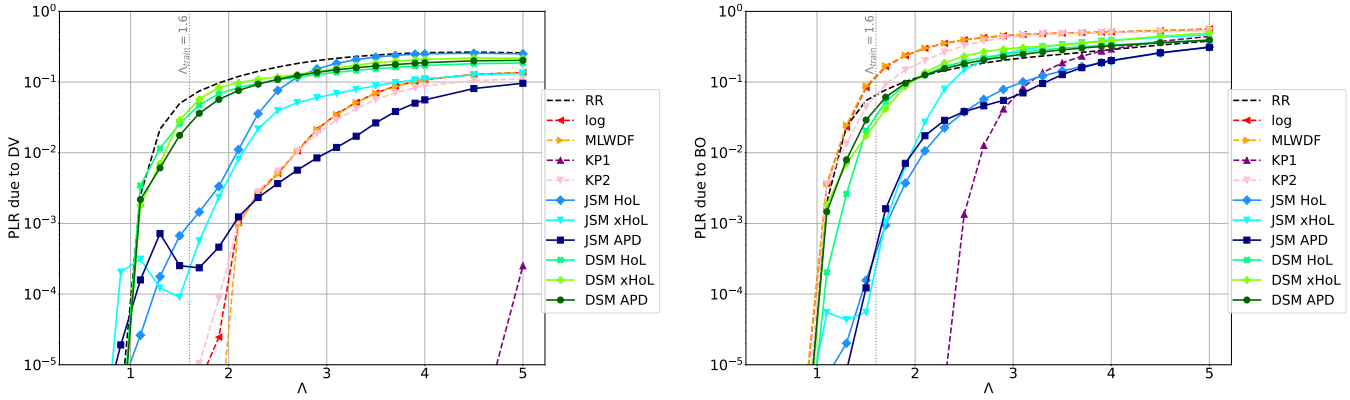
(b) Without the implementation trick.

Figure 4.7: PLR due to DV plus BO vs. Λ.

One can observe that:

- There is almost no difference between the two figures, with and without the implementation trick, meaning that it prevents only from packet loss due to the channel.
- For low arrival rates, i.e. $\Lambda < 1$, there is few packet loss due to DV and BO.
- For $\Lambda \geq 1.1$, the proposed architecture obtains lower PLR due to DV and BO, with a factor 10.
- Concerning the different state space input for the JSM approach, for $\Lambda \leq 0.7$, the different methods are closed. Concerning the JSM xHoL, it has a higher PLR than the other methods for $\Lambda \in [0.8, 1.1]$, is better than JSM APD for $\Lambda \geq 1.2$ and better than JSM HoL for $\Lambda \geq 1.3$. The JSM HoL has a lower PLR than the JSM APD for $\Lambda \in [0.5, 1.4]$.
- KP1 achieves low PLR due to DV and BO for $\Lambda < 2.5$, then its PLR increases drastically until reaches the PLR of JSM APD for $\Lambda \geq 3$. This implies that for $\Lambda < 3$, the majority of the packet loss of KP1 comes from the channel.

Figures 4.8a and 4.8b plot the PLR due to DV (ξ_{DV}) and BO (ξ_{BO}) vs. Λ respectively. Since there is no great difference for the sum of the PLR due to DV and BO, only the results without the implementation tricks are plotted.



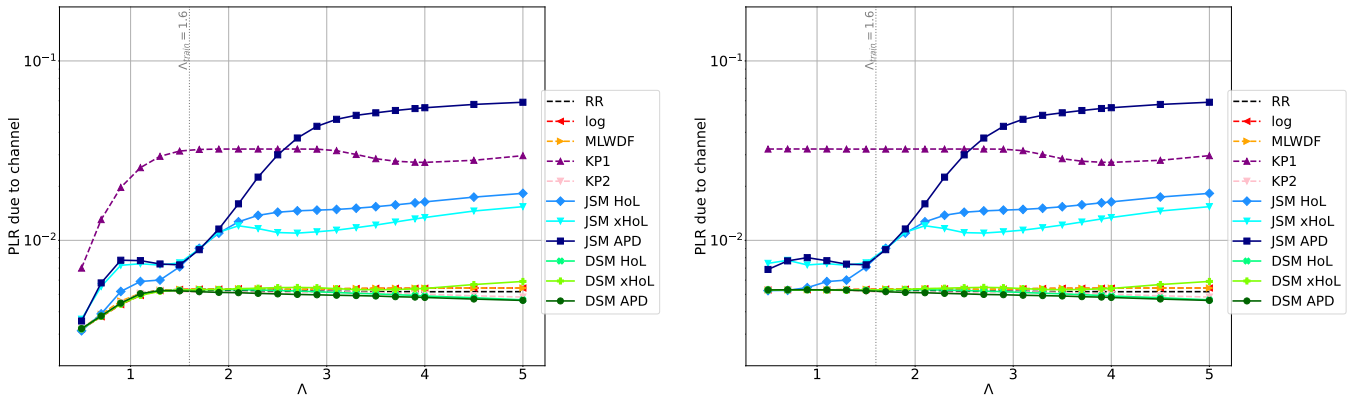
(a) PLR due to DV without the implementation trick. (b) PLR due to BO without the implementation trick.

Figure 4.8: PLR due to DV (a) and PLR due to BO (b) vs. Λ.

One can observe that:

- There is little packet loss due to DV and BO for all methods when $\Lambda < 1$.
- The KP1 obtains little packet loss due to DV regardless the value of Λ and little packet loss due to BO for $\Lambda < 2.7$.
- Except for RR, DSM-HBS methods lose fewer packets due to DV than DRL-based methods. However, they lose more packets due to BO.
- There is at least one order of magnitude difference between one of the JSM methods and all DSM-LBS methods for $1.1 < \Lambda < 2.1$, across the different types of loss. For $2.1 < \Lambda < 2.9$, both JSM-APD and JSM-xHoL exhibit an order of magnitude difference in packet loss due to BO, whereas only JSM-APD shows such a difference in packet loss due to DV.
- There is a magnitude order of difference between at least one JSM and all the DSM-LBS ones, for $1.1 < \Lambda < 2.1$ for the different types of loss. For $2.1 < \Lambda < 2.9$, JSM APD and JSM xHoL have one order of magnitude for packet loss due to BO, whereas only the JSM APD has one order of magnitude for packet loss due to DV.

Figure 4.9 plots the PLR due to channel (ζ_{CH}) vs. Λ.



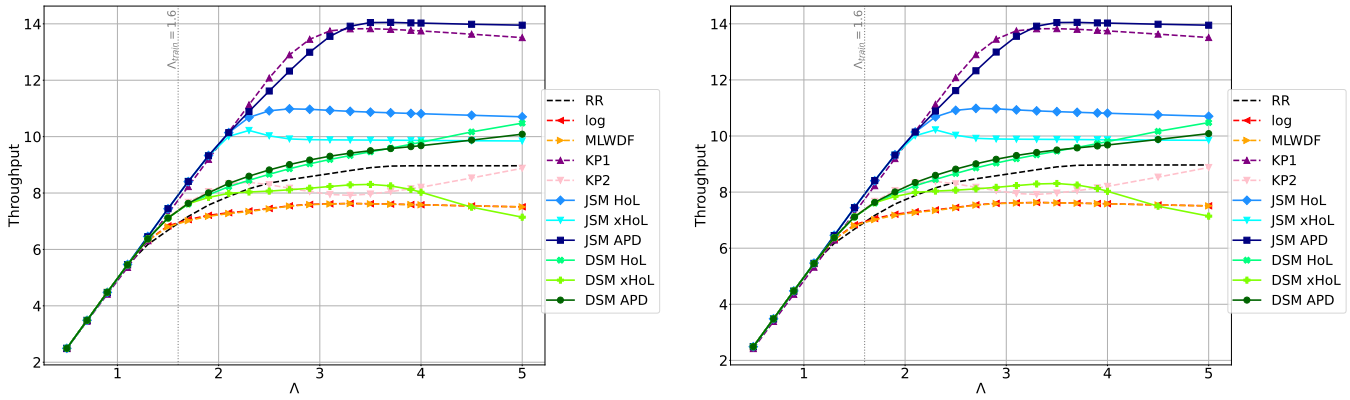
(a) With the implementation trick. (b) Without the implementation trick.

Figure 4.9: PLR due to channel vs. Λ.

One can observe that:

- The first setup with the implementation trick offers better performance for $\Lambda < 1.2$. For $\Lambda \geq 1.2$, the two setups are nearly identical.
- For $\Lambda \geq 1.2$, the PLR due to channel of the DSM methods is almost constant for both setups.
- The PLR due to channel of DSM methods remains below the PER target, as expected, since the chosen MCS guarantees a PLR due to channel under the specified threshold.
- The PLR due to channel for the JSM architectures increases as Λ increases. Additionally, the PLR due to channel for these architectures is higher than the PLR due to channel of the DSM methods, except for the JSM HoL which have the same PER than the DSM methods for $\Lambda \leq 0.6$.
- The PLR due to channel of the JSM APD exceeds the one of the KP1 for $\Lambda \geq 2.7$. This signifies that this architecture selects less robust MCS to empty faster the buffer for high arrival rates.

Figure 4.10 plots the throughput (η) vs. Λ.



(a) With the implementation trick.

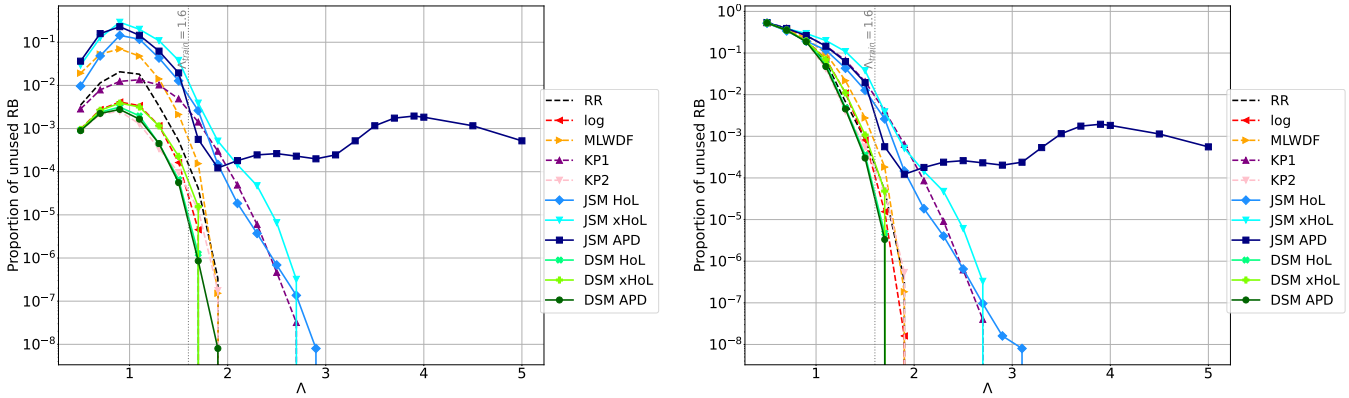
(b) Without the implementation trick.

Figure 4.10: Throughput vs. Λ.

One can observe that:

- All the curves are superimposed for $\Lambda \leq 1.2$. It is worth noting that the implementation trick does not affect the performance of the throughput.
- For $1.2 \leq \Lambda \leq 2.2$, the JSM methods and KP1 offer the same throughput, surpassing the other DSM-LBS methods with $\bar{q}_{\text{th}} = 10^{-2}$.
- For $\Lambda \geq 2.2$, JSM APD and KP1 remain close in performance and outperform the other methods, with 14 packets per frame, which is close to the bound of 15 packets. JSM HoL saturates at 11 packets per frame, while JSM xHoL reaches 10. DSM-LBS HoL and DSM-LBS APD also achieve 10 packets per frame, outperforming DSM-HBS using HMS with $\bar{q}_{\text{th}} = 10^{-2}$, which are limited to 9 packets per frame.

Figure 4.11 plots the proportion of unused RBs ($\mathcal{U}$) vs. Λ.



(a) With the implementation trick.

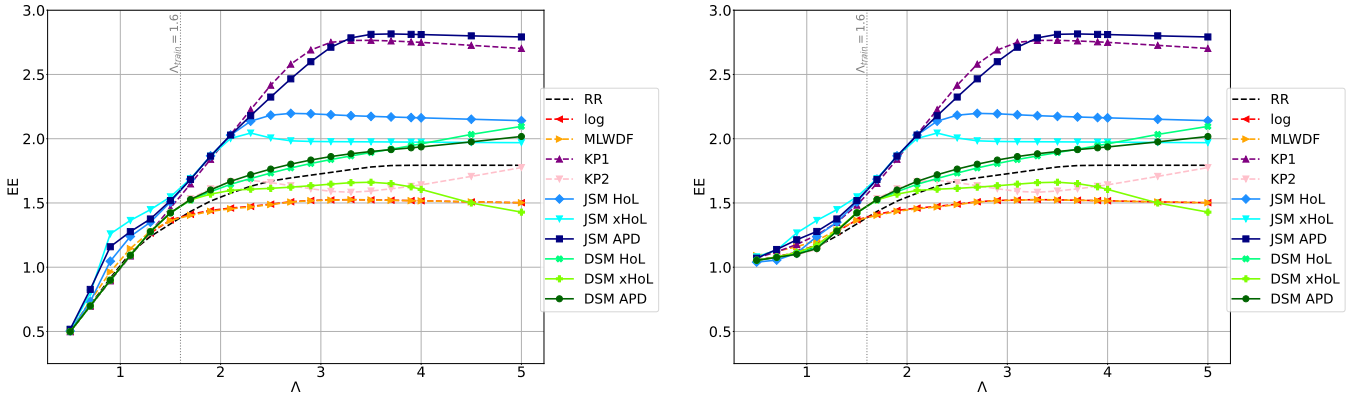
(b) Without the implementation trick.

Figure 4.11: Proportion of unused RB vs. Λ.

One can observe that:

- In the first setup, the implementation trick leads the methods to use more RBs when arrival rates are low (i.e., $\Lambda < 1$). Without this trick, nearly 50% of the RBs remain unused at $\Lambda = 0.5$, with usage gradually decreasing as Λ rises, except for JSM APD. As Λ increases beyond 1, the proportion of used RBs decreases, since the buffers are filling up and more RBs are needed to empty them .
- The JSM methods lead to higher unused RBs than the other methods, whereas the DSM-LBS methods and KP2 lead to lower unused RBs than the other methods.
- The JSM APD continues to leave some RBs unused regardless of the value of Λ. This may be due to its frequent selection of the highest MCS to empty the buffers when Λ is high.

Figure 4.12 plots the EE ($\mathcal{E}$), where the power is considered unitary, vs. Λ.



(a) With the implementation trick.

(b) Without the implementation trick.

Figure 4.12: Energy efficiency vs. Λ.

One can observe that:

- The setup with the implementation trick results in lower EE for small values of Λ, while for $\Lambda > 1.5$, both setups yield nearly the same EE.
- The proposed JSM methods, in particular APD, lead to better EE in general, than the other methods. This highlights the gain to use a joint optimization.

4.7 Conclusion

We addressed in this chapter the joint scheduling and MCS selection problem over a frame, considering statistical CSI. The objective was to minimize the packet loss due to DV, BO and channel effect.

We have proposed and evaluated two different approaches for the DRL-based methods: the first one, referred to as JSM, uses an EOT-AB architecture that jointly selects the buffer and the MCS. The second one, referred to as DSM-LBS, uses an EOT-AB architecture that selects only the buffers for each RB of the frame, while relying on an heuristic for MCS selection. One of the main difference between both approaches is that for the DSM, the MCS is selected once for all transmissions based on the average SNR value, whereas for the JSM, the MCS is selected for each RB. These approaches are compared along with DSM-HBS, where heuristics are used for both MCS and buffer selections.

Simulation results show that JSM solutions (in particular JSM APD) outperforms the DSM ones. This can explained as follows:

- Both DSM solution have difficulties to maintain low PLR under different traffic loads due to its lack of flexibility. Indeed, these methods are constrained by the fixed MCS selection, which may be either insufficiently robust, resulting in a high PLR due to channel errors, or not efficient enough to empty the buffer, leading to high PLR due by DV and BO.
- The JSM solution offers better performance in terms of total PLR, throughput and EE at the expense of a higher PLR due to the channel. Specifically, it demonstrates an adaptation in the MCS selection strategy based on the traffic load, accepting a higher PLR due to the channel conditions in order to mitigate the total PLR. In contrast, the DSM approach yields lower PLR due to the channel conditions but results in a higher total PLR, reduced throughput, and lower EE. This is due to its reliance on heuristic-based MCS selection, which lacks the flexibility to adjust to varying traffic loads. These observations highlight the benefit of jointly performing scheduling and MCS selection to dynamically adapt decisions based on the state of the buffers.

Since the average PER is used in the state space, it would be interesting in future works to study if the proposed architecture succeeds to generalize over different channel conditions (e.g. multipath Rayleigh channel, Rician channel). Moreover, the average SNR remained constant throughout each episode, which differs from real-world scenarios where link mobility causes variations in average SNR. Future works could explore integrating mobile links to better reflect these dynamics such as in Chapter 3. In addition, they could integrate repetition mechanism to mitigate the PER.

Part of the material presented in this chapter has been published in [87] and patented in [85].

Conclusions and perspectives

The main objective of this thesis was to propose DRL solutions for a central joint optimization of scheduling and resource allocation in multi-user wireless communication systems.

In Chapter 1, we presented the general system model along with a SotA of both heuristics and DRL-based solutions for scheduling and resource allocation. We recommended that an effective DNN architecture for scheduling should satisfy the following three key properties: NLI, PE and GBM. We then proposed an original classification of the DRL schedulers based on these three properties. We identified that the EOT architecture possesses these three properties thanks to its attention mechanism, and we thus proposed to use it as a basis for the scheduling solutions developed throughout this thesis.

In Chapter 2, we considered a slot-based scheduling problem with two types of traffic, DC and BE, and assuming error-free propagation channel. We trained the EOT using DQL for a specific number of links and for a specific value for the traffic arrival rate, and we compared its performance against heuristics, and a conventional FC DNN scheduler. We evaluated the generalization capability of the EOT scheduler with respect to traffic arrival rate and the number of links in the networks not seen during the training. The performance are evaluated in terms of packet loss, throughput, fairness, and packet delay. Our results showed that 1) EOT outperforms the heuristics, which are NLI and 2) EOT also outperforms the FC-based schedulers, although the latter were trained specifically for a given number of links, i.e., one FC-based scheduler per trained/tested number of links. Remarkably, EOT maintains robust performance even when tested under link configurations it had not encountered during training, confirming its scalability, adaptability, and good generalization capability.

In Chapter 3, we considered a frame-based scheduling problem with four types of traffic and assuming error free propagation channel. We introduced the EOT-AB architecture trained with DDQL to jointly perform RB allocation over a frame. This model combines the benefits of the EOT, which is NLI, PE, and GBM, with the AB architecture, which enables managing large discrete action space through a structured action decomposition, where each branch handles the selection of a specific RB. To prevent the allocation of RBs to empty buffers, we incorporated adaptive action masking. Experimental results showed that jointly allocating RBs across the frame outperforms heuristics that operate on an RB-by-RB basis. Our solution shows also better results than the heuristics. Additionally, we demonstrated the effectiveness of adaptive action masking during inference, highlighting its importance for improved performance during inference phases.

In Chapter 4, we also considered a frame-based scheduling problem with two types of traffic, DC and BE, and assuming Rayleigh flat fading propagation channel. We extended the scheduling problem by incorporating MCS selection alongside buffer scheduling over a frame. We introduced the packet loss due to the channel. The probability of correctly transmitting the packets depends on the selected MCS. Two approaches were evaluated: 1) a joint solution, where both buffer and MCS selections

are made simultaneously by a single EOT-AB architecture; and 2) a disjoint solution, where the EOT-AB architecture performs buffer selection, and the MCS is determined heuristically based on average PER conditions. Experimental results demonstrated that the joint solution consistently outperforms the disjoint one in terms of PLR across various traffic arrival rates, highlighting the benefit of learning both tasks in a unified framework.

Perspectives

The following challenges are identified as perspectives for future research.

System design

- In this thesis, we assumed either perfect packet transmission or transmission with errors. In the latter case, retransmission mechanisms such as HARQ, which are commonly used in practical communication systems, were not considered. Future work should therefore incorporate HARQ alongside the scheduling process to better reflect real-world conditions and improve performance.
- We assumed fixed transmit power in this thesis. It should be of interest to allow dynamic transmit power adaptation to improve the EE of the system [1].
- We assumed frequency flat Rayleigh fading channel. It should be of interest to consider more realistic multipath channel models such as the ones from [88].

DRL improvement

- We trained the EOT-AB using DDQL. It should be of interest to implement other DRL algorithms such as soft actor critic (SAC) [89] or proximal policy optimization (PPO) [90] for performance comparisons.
- As the system model evolves, the associated DRL methods must also be adapted accordingly. For example, incorporating transmit power control, which can be modeled as a continuous action, requires modifying the AB architecture to handle a mixed discrete-continuous action space. One possible approach is to leverage methods such as those proposed in [91], which are specifically designed to address this type of hybrid action setting.
- Both Chapters 2 and 4 focus on packet loss minimization. However, in practical situations, it is often important to manage multiple objectives simultaneously, such as fairness, delay, or EE. Addressing these potentially conflicting goals with a single architecture, without retraining for each new set of objective weights, would be highly beneficial. One promising approach is the method proposed in [53], which enables multi-objective optimization within a unified learning framework.
- Throughout this thesis, training was performed with a specific GAR. It would be interesting to investigate whether training over a range of arrival rates could improve performance. Additionally, enriching the state representation, by including features such as the current arrival rate or historical buffer states, could help the model better adapt to dynamic traffic conditions and improve generalization.

Appendix A

Background on machine learning

A.1 Introduction

This Appendix introduces the concept of ML as applied throughout this thesis. It begins with an overview of RL in Section A.2, which first explores fundamental concepts like Markov chains and MDPs. Next, we discuss various methods for solving an MDP, typically by computing the optimal policy that specifies the best action to take in each state to maximize the expected return. Specifically, Section A.2.6 focuses on the VI algorithm and Section A.2.7 on the Q-Learning algorithm. Section A.3 addresses the DRL approach, that leverages DNNs to solve an MDP with high-dimensional state spaces, and more particularly the DQL algorithm. Finally, Section A.4 presents the specific architectures utilized in this thesis.

A.2 Reinforcement learning

A.2.1 Introduction

In the RL paradigm [80], an agent interacts with an environment such as illustrated in Figure A.1.

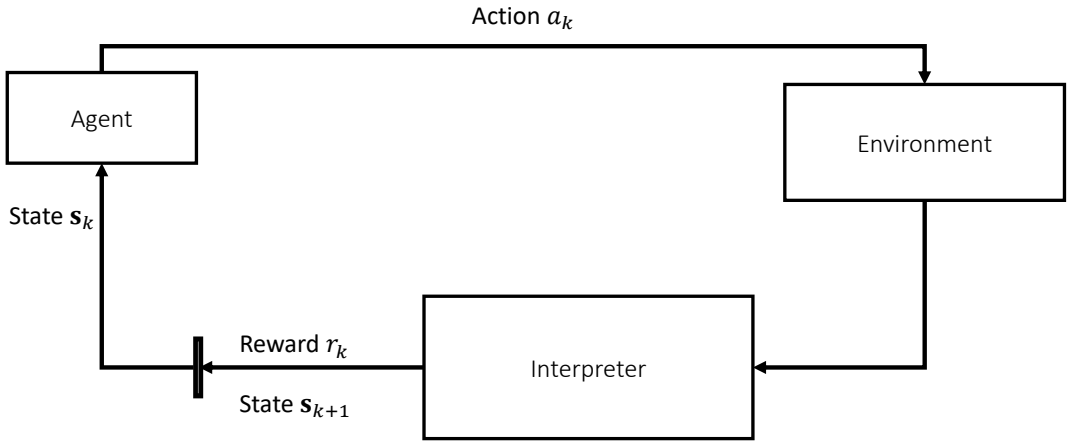


Figure A.1: RL interaction with the environment.

The agent selects at each step k an action A_k, based on the state S_k, which is an observation of the

environment. After taking action A_k, the environment is abstracted by an *interpreter*, which outputs the reward R_{k+1} and the next state S_{k+1}.

We think it is of interest to incorporate an interpreter in Sutton's RL schemes [80, Figure 3.1]. This interpreter can be seen as a way to elaborate an observation state S_k from the environment from which it may be difficult to define the *true* state. The way the observation state is built is left to the solution designer who makes technical choices to take into account some constraints, such as metrics availability or agent internal status. For instance, let us assume a wireless packet scheduler acting as the agent and interacting with an environment made up of several links, each characterized by a transmission quality. The transmission quality can, for instance, be represented by a SNR, a CQI or a PER. The state can then be complemented by the agent's internal status information, such as buffer occupancy.

A common assumption in RL is that the environment is Markovian. When 1) the underlying Markovian model is known i.e. when the transition probabilities are known, and 2) both the action and the state space are of relatively low dimension, the MDP can be solved using dynamic programming algorithm such as VI, detailed in Section A.2.6. This approach is referred to as *model-based* since the transition probabilities are assumed to be known. Otherwise, RL algorithms need to be used. If 2) is fulfilled but 1) is not, then the Q-Learning detailed in Section A.2.7 can be applied. If neither 1) nor 2) are fulfilled, then one can resort to DQL, as detailed in Section A.3. The RL approach does not rely on prior knowledge of the model. Instead, the RL algorithm learns the unknown transition probabilities and is therefore referred to as *model-free*.

Note that in the remainder of Section A.2, we consider two complementary approaches to represent MDPs:

1. The *conventional* approach, which represents the MDP using transition probabilities and rewards, as in the book by Sutton and Barton [80].
2. The *optimal control* approach, which incorporates a stochastic perturbation model, as in the book by Bertsekas [92]. This approach provides a theoretical framework for the scheduling problem, where perturbations represent the packet arrivals.

We present both approaches and show that they are equivalent in the sense that they lead to the same MDP model. All the material presented hereafter can be found in either [80] or [92].

A.2.2 Finite Markov chain

A discrete-time stochastic process $\{S_k\}_{k\geq 0}$ with finite state space $\mathbb{S} = \{\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^n\}$, i.e. $|\mathbb{S}| < +\infty$, is called a *finite Markov chain* if and only if it satisfies, for all integers $k \geq 0$ and all states $\mathbf{s}^{i_{k+1}}, \mathbf{s}^{i_k}, \ldots, \mathbf{s}^{i_0}$ belonging to $\mathbb{S}$:

$$\Pr(S_{k+1} = \mathbf{s}^{i_{k+1}} \mid S_k = \mathbf{s}^{i_k}, S_{k-1} = \mathbf{s}^{i_{k-1}}, \ldots, S_0 = \mathbf{s}^{i_0}) = \Pr(S_{k+1} = \mathbf{s}^{i_{k+1}} \mid S_k = \mathbf{s}^{i_k}). \quad \text{(A.1)}$$

We assume that all Markov chains considered in this thesis are finite and *homogeneous*, meaning that the right-hand side of (A.1) is independent of k.

A Markov chain can be thus defined by the set of *transition probabilities* $p_{i_{k+1},i_k} := \Pr(S_{k+1} = \mathbf{s}^{i_{k+1}} \mid S_k = \mathbf{s}^{i_k})$ with $\mathbf{s}^{i_k} \in \mathbb{S}$ and $\mathbf{s}^{i_{k+1}} \in \mathbb{S}$. Let us take the example of a Markov chain with three states, i.e. $\mathbb{S} = \{\mathbf{s}^1, \mathbf{s}^2, \mathbf{s}^3\}$. It can be represented by the graph in Figure A.2 which shows the states and the associated transition probabilities.

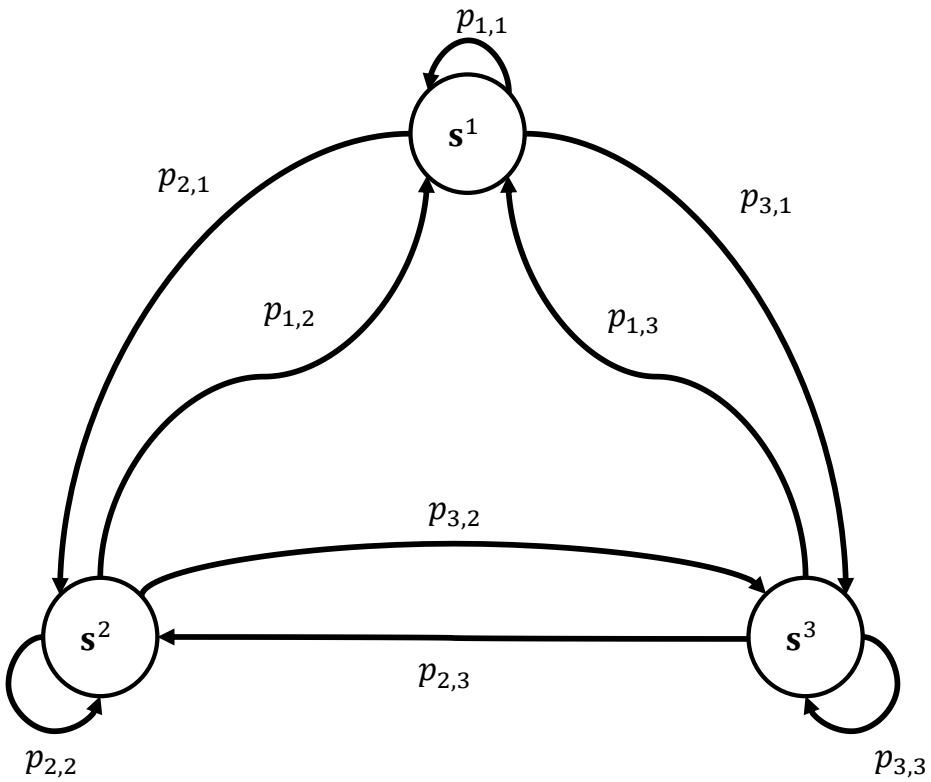


Figure A.2: Graph of transition probabilities of a Markov chain with $n = 3$ states.

A.2.3 Finite Markov Decision Process

A finite MDP extends the finite Markov chain framework by introducing actions and rewards. Recall from the RL context in Section A.2.1 that, in a given state $S_k \in \mathbb{S}$, the agent takes an action A_k and receives a reward R_{k+1}. The actions are chosen from a finite set $\mathbb{A}$ with $|\mathbb{A}| < +\infty$. Then, the system transitions to a new state S_{k+1}. This process is repeated over time. The sequence of successive states, actions, rewards, starting from $k = 0$, i.e. $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, S_3, \ldots$, is called a *trajectory*.

The rewards are in general real-valued and bounded, taking values in a set $\mathcal{R} \subseteq \mathbb{R}$, which is typically assumed to be uncountable. In [80], however, authors assume that the rewards take values in a finite set to simplify notation and derivations. In our scheduling application, the reward depends on the number of lost packets through a bounded function. Because the number of lost packets due to buffer overflow can, in theory, be arbitrarily large, $\mathcal{R}$ is countably infinite.

A finite MDP can be defined by the tuple $(\mathbb{S}, \mathbb{A}, p(\cdot, \cdot \mid \mathbf{s}, a))$, where $p(\cdot, \cdot \mid \mathbf{s}, a)$ is the four-argument function [80, (3.2)]

$$p(\mathbf{s}', r \mid \mathbf{s}, a) := \Pr\{S_k = \mathbf{s}', R_k = r \mid S_{k-1} = \mathbf{s}, A_{k-1} = a\}, \tag{A.2}$$

where we use as in [80] the convention $\mathbf{s}' = \mathbf{s}^k$ and $\mathbf{s} = \mathbf{s}^{k-1}$. Notice that (A.2) defines both the next state and the corresponding reward at the same time.

The function $p(\mathbf{s}', r \mid \mathbf{s}, a)$ defines the *dynamics* of the MDP and verifies

$$\sum_{\mathbf{s}' \in \mathbb{S}} \sum_{r \in \mathcal{R}} p(\mathbf{s}', r \mid \mathbf{s}, a) = 1, \quad \forall \mathbf{s} \in \mathbb{S}, a \in \mathbb{A}. \tag{A.3}$$

We now derive other expressions for the transition probabilities and expected rewards from function (A.2), leading to a different MDP definition.

An alternative way to define a finite MDP (equivalent to the one generated by (A.2)) is to define the three-argument transition probability $p(\mathbf{s}' \mid \mathbf{s}, a)$ along with the corresponding three-argument expected reward $r(\mathbf{s}, a, \mathbf{s}')$:

$$p(\mathbf{s}' \mid \mathbf{s}, a) := \Pr\{S_k = \mathbf{s}' \mid S_{k-1} = \mathbf{s}, A_{k-1} = a\}, \tag{A.4}$$

$$r(\mathbf{s}, a, \mathbf{s}') := \mathbb{E}\left[R_k \mid S_{k-1} = \mathbf{s}, A_k = a, S_k = \mathbf{s}'\right]. \tag{A.5}$$

The tuple $(\mathbb{S}, \mathbb{A}, p(\cdot \mid \mathbf{s}, a), r(\mathbf{s}, a, \mathbf{s}'))$ defines an MDP that is the same as the one defined by the tuple $(\mathbb{S}, \mathbb{A}, p(\cdot, \cdot \mid \mathbf{s}, a))$. This can be proved by showing that (A.4) and (A.5) can be expressed using (A.2).

To do so, one can first express $p(\mathbf{s}' \mid \mathbf{s}, a)$ as a function of the four-argument transition probability (A.2) using the identity:

$$p(\mathbf{s}' \mid \mathbf{s}, a) = \sum_{r \in \mathcal{R}} p(\mathbf{s}', r \mid \mathbf{s}, a). \tag{A.6}$$

Then, applying the expectation definition to (A.5), we get $r(\mathbf{s}, a, \mathbf{s}') = \sum_{r \in \mathcal{R}} r p(r \mid \mathbf{s}, a, \mathbf{s}')$, which after some calculations (proof is omitted) can be rewritten [80, (3.6)] as a function of the four-argument transition probability:

$$r(\mathbf{s}, a, \mathbf{s}') = \sum_{r \in \mathcal{R}} r \frac{p(\mathbf{s}', r \mid \mathbf{s}, a)}{p(\mathbf{s}' \mid \mathbf{s}, a)}. \tag{A.7}$$

Last, one can also define the two-argument expected reward $r(\mathbf{s}, a)$ that is often used in the optimal policy algorithm development (e.g. Section A.2.6 and subsequent sections) as [80, 3.5]:

$$r(\mathbf{s}, a) := \mathbb{E}\left[R_k \mid S_{k-1} = \mathbf{s}, A_k = a\right]. \tag{A.8}$$

Applying the expectation definition to (A.8), we get $r(\mathbf{s}, a) = \sum_{r \in \mathcal{R}} r p(r \mid \mathbf{s}, a)$. Using the following identity: $p(r \mid \mathbf{s}, a) = \sum_{\mathbf{s}' \in \mathbb{S}} p(\mathbf{s}', r \mid \mathbf{s}, a)$, we find [80, 3.5]:

$$r(\mathbf{s}, a) = \sum_{r \in \mathcal{R}} r \sum_{\mathbf{s}' \in \mathbb{S}} p(\mathbf{s}', r \mid \mathbf{s}, a). \tag{A.9}$$

For a given $(\mathbf{s}, a)$, if taking action a in state $\mathbf{s}$ always leads to a specific next state $\mathbf{s}'$, i.e. $p(\mathbf{s}' \mid \mathbf{s}, a) = 1$, the corresponding MDP is called *deterministic MDP*. On the other hand, if taking action a in state $\mathbf{s}$ can lead to multiple possible next steps $\mathbf{s}'$ with $p(\mathbf{s}' \mid \mathbf{s}, a) \neq 0$, the MDP is called *stochastic MDP*. This occurs when random perturbations affect the state transition. The scheduling problem considered in this thesis is modeled as a stochastic MDP.

As an example, Figure A.3 shows a graph representing the three-argument transition probabilities of an MDP with two states, $\mathbb{S} = \{\mathbf{s}^1, \mathbf{s}^2\}$, and two actions, $\mathbb{A} = \{a^1, a^2\}$.

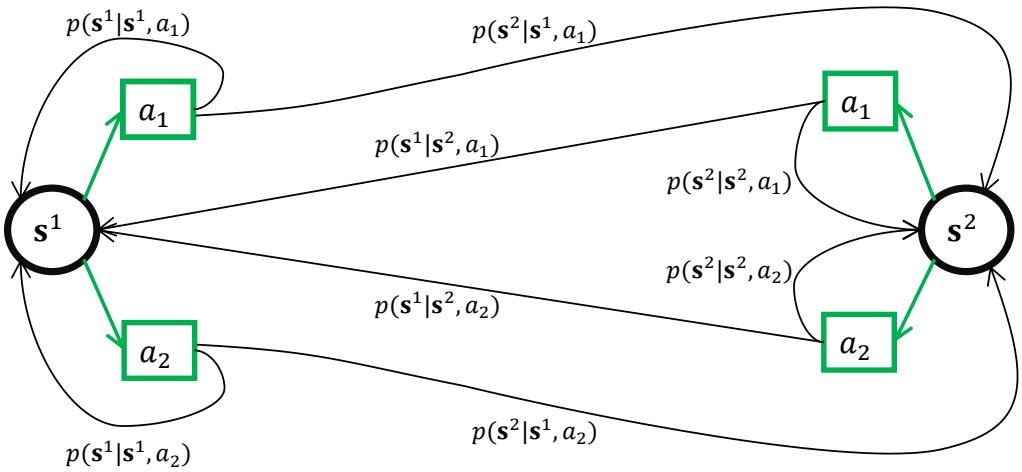


Figure A.3: Graph representing the three-argument transition probabilities of an MDP with two states, $\mathbb{S} = \{\mathbf{s}^1, \mathbf{s}^2\}$, and two actions, $\mathbb{A} = \{a^1, a^2\}$.

A.2.4 Optimal control framework and MDP

The optimal control framework for stochastic discrete-time dynamic systems, as developed for instance by Bertsekas [92], explicitly accounts for random perturbations occurring in the state transitions and is therefore well suited to the scheduling process with random packet arrivals considered in this thesis.

Following Bertsekas' framework and our notation conventions, a dynamic stochastic system is modeled by the equation:

$$S_{k+1} := f(S_k, A_k, w_k) \tag{A.10}$$

where f is the *state transition function*, or the *system function*, and $w_k \in \mathbb{W}$ is the stochastic perturbation at time k with probability distribution $p_W(\cdot)$. As in [92], we assume that the disturbance set $\mathbb{W}$ is countable.

A cost is associated to each transition, and is defined through the cost function g, referred to as *cost per stage* in [92], as:

$$C_k := g(S_k, A_k, w_k) \tag{A.11}$$

which is used to derive optimal policies that minimize the cumulative cost.

Given (A.10) and (A.11), the trajectory $S_0, A_0, C_1, S_1, A_1, C_2, S_2 \ldots$ is an MDP, which is defined by the tuple $\{\mathbb{S}, \mathbb{A}, \mathbb{W}, p_W, f, g\}$.

In the following, for the sake of clear comparison, we refer to $\{\mathbb{S}, \mathbb{A}, \mathbb{W}, p_W, f, g\}$ as MDP-OC (OC for optimal control) to distinguish it from the MDP defined in Section A.2.3, namely $(\mathbb{S}, \mathbb{A}, p(\cdot \mid \mathbf{s}, a), r(\mathbf{s}, a, \mathbf{s}'))$, which we refer to as MDP-C (C for conventional).

We now prove that, under simple conditions, the MDP-OC is equivalent to the MDP-C, as the latter can be deduced from the former. In the MDP-C, randomness is captured by transition probabilities rather than by explicit disturbances. This result is important because it enables the application of reinforcement learning frameworks developed for the MDP-C to scheduling problems modeled using the optimal control formulation.

In what follows, we show that all the rewards and transition probabilities defined in Section A.2.3 can be expressed as a function of f, g, and w. To do so, we switch to the Sutton and Barto notation, as in the previous section, for clarity. In addition, to simplify notation, and without loss of generality, we omit the random variable names, e.g., we write $\Pr\{S_k = \mathbf{s}' \mid S_{k-1} = \mathbf{s}, A_{k-1} = a\} = \Pr\{\mathbf{s}' \mid \mathbf{s}, a\}$, or $\mathbb{E}\left[R_k \mid S_{k-1} = \mathbf{s}, A_k = a, S_k = \mathbf{s}'\right] = \mathbb{E}\left[R \mid \mathbf{s}, a, \mathbf{s}'\right]$.

We identify the state transition (A.10) and cost (A.11) as:

$$\mathbf{s}' = f\left(\mathbf{s}, a, w\right), \tag{A.12}$$

$$c = g(\mathbf{s}, a, w). \tag{A.13}$$

We notice that in the MDP-C, each transition is associated to a reward, whereas in the MDP-OC, each transition is associated to a cost. Thus, to make the two MDPs equivalent, we need to define the cost as the opposite of the reward, i.e.:

$$r = -g(\mathbf{s}, a, w). \tag{A.14}$$

Regarding the four-argument transition probability, one can re-express (A.2) as:

$$p(\mathbf{s}', r \mid \mathbf{s}, a) = \Pr\{f\left(\mathbf{s}, a, w\right) = \mathbf{s}', g(\mathbf{s}, a, w) = -r \mid \mathbf{s}, a\}. \tag{A.15}$$

Introducing the set

$$\mathbb{W}_{\mathbf{s},a}^{\mathbf{s}',r} := \{w : f(\mathbf{s}, a, w) = \mathbf{s}', g(\mathbf{s}, a, w) = -r \mid \mathbf{s}, a\} \tag{A.16}$$

the right-hand side of (A.15) is computed as:

$$\Pr\{f\left(\mathbf{s}, a, w\right) = \mathbf{s}', g(\mathbf{s}, a, w) = -r \mid \mathbf{s}, a\} = \sum_{w \in \mathbb{W}_{\mathbf{s},a}^{\mathbf{s}',r}} p_W(w) \tag{A.17}$$

thus leading to:

$$p(\mathbf{s}', r \mid \mathbf{s}, a) = \sum_{w \in \mathbb{W}_{\mathbf{s},a}^{\mathbf{s}',r}} p_W(w). \tag{A.18}$$

Regarding the three-argument transition probabilities (A.4), we can write:

$$p(\mathbf{s}' \mid \mathbf{s}, a) = \Pr\{f(\mathbf{s}, a, w) = \mathbf{s}' \mid \mathbf{s}, a\}. \tag{A.19}$$

Introducing the set

$$\mathbb{W}_{\mathbf{s},a}^{\mathbf{s}'} := \{w : f(\mathbf{s}, a, w) = \mathbf{s}' \mid \mathbf{s}, a\} \tag{A.20}$$

the right-hand side of (A.19) is computed as:

$$\Pr\{f(\mathbf{s}, a, w) = \mathbf{s}' \mid \mathbf{s}, a\} = \sum_{w \in \mathbb{W}_{\mathbf{s},a}^{\mathbf{s}'}} p_W(w) \tag{A.21}$$

thus leading to

$$p(\mathbf{s}' \mid \mathbf{s}, a) = \sum_{w \in \mathbb{W}_{\mathbf{s},a}^{\mathbf{s}'}} p_W(w). \tag{A.22}$$

Posing:

$$\mathfrak{R}(\mathbf{s}, a, w) := -g(\mathbf{s}, a, w), \tag{A.23}$$

the three-argument expected reward (A.5) is computed as:

$$\begin{aligned} r(\mathbf{s}, a, \mathbf{s}') &= \mathbb{E}[\mathfrak{R}(\mathbf{s}, a, w) \mid \mathbf{s}, a, \mathbf{s}'] \\ &= \sum_{r \in \mathcal{R}} r \Pr\{g(\mathbf{s}, a, w) = -r \mid f(\mathbf{s}, a, w) = \mathbf{s}', \mathbf{s}, a\}. \end{aligned} \tag{A.24}$$

To compute the probability in (A.24), we apply the Bayes rule to take $f(\mathbf{s}, a, w) = \mathbf{s}'$ out of the conditioning:

$$\Pr\{g(\mathbf{s}, a, w) = -r \mid f(\mathbf{s}, a, w) = \mathbf{s}', \mathbf{s}, a\} = \frac{\Pr\{g(\mathbf{s}, a, w) = -r, f(\mathbf{s}, a, w) = \mathbf{s}' \mid \mathbf{s}, a\}}{\Pr\{f(\mathbf{s}, a, w) = \mathbf{s}' \mid \mathbf{s}, a\}}. \tag{A.25}$$

The numerator in (A.25) is given by (A.17), and the denominator by (A.22), yielding:

$$\Pr\{g(\mathbf{s}, a, w) = -r \mid f(\mathbf{s}, a, w) = \mathbf{s}', \mathbf{s}, a\} = \frac{\sum_{w \in \mathbb{W}_{\mathbf{s},a}^{\mathbf{s}',r}} p_W(w \mid \mathbf{s}, a)}{\sum_{w \in \mathbb{W}_{\mathbf{s},a}^{\mathbf{s}'}} p_W(w \mid \mathbf{s}, a)}. \tag{A.26}$$

Plugging (A.26) into (A.24) finally leads to:

$$r(\mathbf{s}, a, \mathbf{s}') = \sum_{r \in \mathbb{R}} r \frac{\sum_{w \in \mathbb{W}_{\mathbf{s},a}^{\mathbf{s}',r}} p_W(w \mid \mathbf{s}, a)}{\sum_{w \in \mathbb{W}_{\mathbf{s},a}^{\mathbf{s}'}} p_W(w \mid \mathbf{s}, a)}. \tag{A.27}$$

One can notice that (A.27) is consistent with (A.7) using (A.18) and (A.22).

The two-argument expected reward (A.8) is computed as:

$$r(\mathbf{s}, a) = \mathbb{E}[\mathfrak{R}(\mathbf{s}, a, w) \mid \mathbf{s}, a] \tag{A.28}$$

$$= \sum_{r \in \mathcal{R}} r \Pr\{g(\mathbf{s}, a, w) = -r \mid \mathbf{s}, a\}. \tag{A.29}$$

Introducing

$$\mathbb{W}_{\mathbf{s},a}^{r} := \{w : g(\mathbf{s}, a, w) = -r \mid \mathbf{s}, a\} \tag{A.30}$$

we have:

$$\Pr\{g(\mathbf{s}, a, w) = -r \mid \mathbf{s}, a\} = \sum_{w \in \mathbb{W}_{\mathbf{s},a}^{r}} p_W(w) \tag{A.31}$$

leading to:

$$r(\mathbf{s}, a) = \sum_{r \in \mathcal{R}} r \sum_{w \in \mathbb{W}_{\mathbf{s},a}^{r}} p_W(w). \tag{A.32}$$

An equivalent way of expressing $r(\mathbf{s}, a)$ with the MDP-OC parameters is obtained using (A.9) along with (A.18):

$$r(\mathbf{s}, a) = \sum_{r \in \mathcal{R}} r \sum_{\mathbf{s}' \in \mathbb{S}} p(\mathbf{s}', r \mid \mathbf{s}, a) \tag{A.33}$$

$$= \sum_{r \in \mathcal{R}} r \sum_{\mathbf{s}' \in \mathbb{S}} \sum_{w \in \mathbb{W}_{\mathbf{s},a}^{\mathbf{s}',r}} p_W(w). \tag{A.34}$$

In the following sections, we use equivalently both MDP modelings, MDP-OC or MDP-C depending on the context.

A.2.5 Optimal policy

In the previous sections, we defined and characterized an MDP without specifying how actions A_k are selected, thereby leaving the *decision* aspect of the MDP—that is, how actions should be chosen—unaddressed.

One of the main goal of modeling a problem with an MDP is to find the best way to select actions according to a given criterion. To formalize how actions are selected, we introduce the notion of policy π, which is a function that specifies the action to take in state S_k.

A policy can be classified as:

- **Deterministic:** $\pi(S_k) := A_k$, i.e., π is a function,
- **Stochastic:** the action A_k is drawn according to $\pi(A_k \mid S_k)$, i.e., π is a distribution.

In finite discounted MDPs, deterministic policies are sufficient for optimality [80, Section 4.2, p. 79], [92, Section 1.1.4, p. 13]. Stochastic policies are primarily useful for exploration or in settings with constraints, average-reward criteria, or multi-objective settings.

Therefore, unless otherwise specified, in the remainder of this document:

- We consider deterministic policies, thus $A_k = \pi(S_k)$.
- For the sake of notational simplicity, we may use A_k instead of $\pi(S_k)$ when convenient.
- We assume that the policies considered are *stationary*, i.e., independent of the time index k.

In that context, for the MDP-C case, the trajectories take the form: S_0, $\pi(S_0)$, R_1, S_1, $\pi(S_1)$, R_2, S_2, $\pi(S_2)$, R_3, $S_3 \ldots$

In an MDP, where the reward reflects the benefit of taking an action, a common performance criterion is the discounted return. The discounted return at time t is defined as the discounted sum of future rewards [80, (3.8)]:

$$G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{+\infty} \gamma^k R_{t+k+1} \tag{A.35}$$

where $\gamma \in [0, 1]$ is the *discount rate* or *discount factor*. Notice that in this thesis we always consider $\gamma \in (0, 1)$, see discussion in [80] for singular cases $\gamma = 0$ and $\gamma = 1$.

The discounted return corresponds to the outcome of a single trajectory generated by following a policy, and is therefore a random variable. To account for the variability across trajectories, we define the EDR as [80, (3.12)]:

$$J^\pi(\mathbf{s}) := \mathbb{E}\left[G_t \mid S_t = \mathbf{s}\right] \tag{A.36}$$

$$= \mathbb{E}\left[\sum_{k=0}^{+\infty} \gamma^k R_{t+k+1} \mid S_t = \mathbf{s}\right]. \tag{A.37}$$

This quantity evaluates the average discounted return obtained when starting from state $\mathbf{s}$ and following policy π. The expectation is taken with respect to the stochastic transitions and rewards defined by the MDP dynamics which depends on the MDP model. In the case of an MDP-OC, the dynamics are driven by the random perturbation w_k, so the expectation is taken with respect to w_k. In contrast, for an MDP-C, the dynamics are governed by the transition probabilities, and the expectation is taken over the resulting state sequence S_k.

Bertsekas [92, (1.2)] provides a slightly more general definition of (A.37):

$$J^\pi(\mathbf{s}) := \lim_{N\to\infty} \mathbb{E}\left[\sum_{k=0}^{N} \gamma^k R_{t+k+1} \mid S_t = \mathbf{s}\right]. \tag{A.38}$$

As discussed in the footnote on p. 4 of [92], this expression is equivalent to the infinite-series definition (A.37) under mild assumptions, namely that the discount factor satisfies $\gamma < 1$ and the rewards are bounded (or at least bounded in expectation). These conditions ensure convergence of the discounted return, which holds in our context. For the remainder of this Appendix, we use (A.37).

Notice that the EDR is also referred to as the *state-value* function and noted $V^\pi(\mathbf{s})$ or $v_\pi(\mathbf{s})$ depending on the authors. In the following, we note

$$V^\pi(\mathbf{s}) := J^\pi(\mathbf{s}). \tag{A.39}$$

This function plays a key role in the *value iteration* algorithm, described in Section A.2.6.

In what follows, we set $t = 0$ to simplify the notation, without loss of generality. Depending on the underlying MDP formulation, the EDR can be written as:

$$J^\pi(\mathbf{s}) = \mathbb{E}_{w_0,w_1,\dots}\left[\sum_{k=0}^{+\infty} \gamma^k \mathfrak{R}(S_k, A_k, w_k) \mid S_0 = \mathbf{s}\right] \tag{A.40}$$

in the case of an MDP-OC, or as:

$$J^\pi(\mathbf{s}) = \mathbb{E}_{S_1,S_2,\dots}\left[\sum_{k=0}^{+\infty} \gamma^k r(S_k, A_k) \mid S_0 = \mathbf{s}\right] \tag{A.41}$$

in the case of an MDP-C.

Solving an MDP consists in finding the optimal policy π^* that maximizes the EDR:

$$\pi^* := \arg\max_\pi J^\pi(\mathbf{s}). \tag{A.42}$$

By defining the reward of the MDP-OC as in (A.23), the two MDPs become equivalent. Consequently, solving (A.42) with either expression (A.40) or (A.41) yields the same optimal policy.

The following derivations adopt the MDP-OC framework, consistent with the approach in [92], to establish the subsequent results.

Solving (A.42) directly is intractable, since the number of possible policies grows exponentially. An alternative is to solve (A.42) iteratively by formulating it as a dynamic programming (DP) problem, which can be solved using the *Bellman operator* $\mathcal{T}$ (also called *mapping DP* in [92, (1.5)]):

$$(\mathcal{T}J)(\mathbf{s}) := \max_{a\in\mathbb{A}} \mathbb{E}_w\left[\mathfrak{R}(\mathbf{s}, a, w) + \gamma J(f(\mathbf{s}, a, w))\right]. \tag{A.43}$$

Notice that $f(\mathbf{s}, a, w) = \mathbf{s}'$, so (A.43) explicitly shows the dependence of the next state on the current state.

For a deterministic policy π, we also define the operator $\mathcal{T}_\pi$ as:

$$(\mathcal{T}_\pi J)(\mathbf{s}) := \mathbb{E}_w \left[\mathfrak{R}(\mathbf{s}, \pi(\mathbf{s}), w) + \gamma J(f(\mathbf{s}, \pi(\mathbf{s}), w))\right]. \tag{A.44}$$

The Bellman operator has some interesting properties that ensure the convergence to the optimal EDR and allow the characterization of the optimal policies. These properties are recalled below.

Let us denote $J^*(\mathbf{s})$ the optimal state value function (or EDR) when starting in state $\mathbf{s}$, and

$$\mathcal{T}^k := \underbrace{\mathcal{T} \circ \cdots \circ \mathcal{T}}_{k \text{ times}} \tag{A.45}$$

the k-fold composition of $\mathcal{T}$.

The main results, with proofs given in [92], are the following ones.

Proposition 1. [92, Proposition 1.2.1]

For any bounded function $J : \mathbb{S} \mapsto \mathbb{R}$, we have for all $\mathbf{s} \in \mathbb{S}$,

$$J^*(\mathbf{s}) = \lim_{k \to +\infty} (\mathcal{T}^k J)(\mathbf{s}). \tag{A.46}$$

This proposition states that applying the Bellman operator $\mathcal{T}$ infinitely many times to a function J converges to the optimal EDR function J^*, regardless of the initial state $\mathbf{s}$.

Proposition 2. [92, Proposition 1.2.3 (Bellman's Equation)]

The optimal cost function J^* satisfies for all $\mathbf{s} \in \mathbb{S}$:

$$J^*(\mathbf{s}) = \max_{a \in \mathbb{A}} \mathbb{E}_w \left[\mathfrak{R}(\mathbf{s}, a, w) + \gamma \left[J^*(f(\mathbf{s}, a, w))\right]\right]. \tag{A.47}$$

or equivalently

$$J^* = \mathcal{T} J^*. \tag{A.48}$$

Furthermore, J^* is the unique solution of this equation within the class of bounded functions. Moreover, for any bounded function J with $J \geq \mathcal{T} J$ (or $J \leq \mathcal{T} J$), we have $J \geq J^*$ (or $J \leq J^*$, respectively).

This proposition states that J^* is the unique fixed point of the Bellman operator $\mathcal{T}$. This uniqueness guarantees that iterative application of $\mathcal{T}$ converges to J^*, providing the foundation for computing optimal policies, for example, via the *value iteration* algorithm addressed in Section A.2.6. Equation (A.47) is called the *Bellman equation*.

Proposition 3. [92, Proposition 1.2.2] For any bounded function $J : \mathbb{S} \mapsto \mathbb{R}$ and for any stationary and deterministic policy π, we have for all $\mathbf{s} \in \mathbb{S}$,

$$J_\pi(\mathbf{s}) = \lim_{k \to +\infty} (\mathcal{T}_\pi^k J)(\mathbf{s}). \tag{A.49}$$

This proposition states that for any stationary policy π, applying the Bellman operator $\mathcal{T}_\pi$ infinitely many times to a function J will converge to the optimal EDR function J^π, regardless of the initial state $\mathbf{s}$.

Proposition 4. [92, Proposition 1.2.4]

For every stationary policy π, the associated cost function satisfies for all $\mathbf{s} \in \mathbb{S}$,

$$J_\pi(\mathbf{s}) = \mathbb{E}_w \left[\mathfrak{R}(\mathbf{s}, \pi(\mathbf{s}), w) + \gamma J_\pi(f(\mathbf{s}, \pi(\mathbf{s}), w))\right] \tag{A.50}$$

or, equivalently

$$J_\pi = \mathcal{T}_\pi J_\pi. \tag{A.51}$$

Furthermore, J_π is the unique solution of this equation within the class of bounded functions. Moreover, for any bounded function J with $J \geq \mathcal{T}_\pi J$ (or $J \leq \mathcal{T}_\pi J$), we have $J \geq J_\pi$ (or $J \leq J_\pi$, respectively).

This proposition states that J_π is the unique fixed point of $\mathcal{T}_\pi$.

Proposition 5. [92, Proposition 1.2.5 (Necessary and Sufficient Condition for Optimality)]

A stationary policy π is optimal if and only if $\pi(\mathbf{s})$ attains the maximum in Bellman's equation (A.47) for each $\mathbf{s} \in \mathbb{S}$; i.e.,

$$\mathcal{T} J^* = \mathcal{T}_\pi J^*. \tag{A.52}$$

This proposition states that a stationary policy π is optimal, i.e. $\pi = \pi^*$, if and only if its value function J_π which is the fixed point of the operator $\mathcal{T}_\pi$, is equal to the optimal value function J^*, the fixed point of the Bellman operator $\mathcal{T}$, is equal to the optimal, for all $\mathbf{s} \in \mathbb{S}$.

As a synthesis, we can summarize the different results as follows:

- Proposition 1 defines the optimal state value J^*, as the limit of the value function obtained by applying the Bellman operator $\mathcal{T}$ an infinite number of times, and Proposition 2 establishes that J^* is the unique fixed point of $\mathcal{T}$.
- Proposition 3 defines the state value J_π for a stationary policy π as the limit of the value function when repeatedly applying the operator $\mathcal{T}_\pi$, and Proposition 4 shows that J_π is the unique fixed point of $\mathcal{T}_\pi$.
- Proposition 5 characterizes the optimal policy, with $J_{\pi^*} = J^*$. It also states that achieving the optimal value J^* using the Bellman operator leads the optimal policy. This latter statement forms the basis of the value iteration algorithm discussed in the next Section.

A.2.6 Value iteration

This section addresses the VI algorithm, a common method for solving finite MDPs, i.e., for finding the optimal policy π^* and its associated value function that maximizes the EDR. Another well-known algorithm is *policy iteration*, see e.g., [80], which is not covered here.

We use the MDP-C notation where $V^\pi(\mathbf{s})$ is defined as (A.39) with J^π given by (A.41):

$$V^\pi(\mathbf{s}) := \mathbb{E}_{S_1, S_2, \ldots} \left[\sum_{k=0}^{+\infty} \gamma^k r(S_k, A_k) \mid S_0 = \mathbf{s}\right] \tag{A.53}$$

where $V^\pi(\mathbf{s})$ is interpreted as the EDR when starting from state $\mathbf{s}$ and following the policy π.

As stated in the previous section, the VI consists of applying the Bellman operator iteratively until convergence. Let $V_k(\mathbf{s})$ denote the value function at the kth iteration. At the beginning of the VI

algorithm, i.e. $k = 0$, the value function $V_0(\mathbf{s})$ is initialized for each $\mathbf{s} \in \mathbb{S}$. At the kth iteration the update rule is given by:

$$V_k(\mathbf{s}) = (\mathcal{T}V_{k-1})(\mathbf{s}), \tag{A.54}$$

which can be written explicitly as

$$V_k(\mathbf{s}) = \max_{a \in \mathbb{A}} \left[r(\mathbf{s}, a) + \gamma \sum_{\mathbf{s}' \in \mathbb{S}} p(\mathbf{s}' \mid \mathbf{s}, a) V_{k-1}(\mathbf{s}') \right]. \tag{A.55}$$

$$\tag{A.56}$$

From Proposition 1, we have:

$$V^*(\mathbf{s}) = \lim_{k \to +\infty} (\mathcal{T}^k V_0)(\mathbf{s}). \tag{A.57}$$

This guarantees that repeated application of the Bellman operator converges to the optimal value function, which, by Proposition 5, yields to the optimal policy. Since convergence is guaranteed regardless of initialization, the choice of $V_0(\mathbf{s})$ does not affect correctness but only the speed of convergence. In practice, the initial values are commonly set to zero, though random initialization is also possible.

Since the Bellman operator cannot be applied infinitely many times in practice, a stopping criterion is required to assess convergence. A typical choice in the VI algorithm is:

$$\max_{\mathbf{s} \in \mathbb{S}} |V_k(\mathbf{s}) - V_{k-1}(\mathbf{s})| \leq \epsilon, \tag{A.58}$$

where ϵ is predefined threshold.

When the optimal state-value function V^* is reached, thus equal to J^*, the optimal policy is given by:

$$\pi^*(\mathbf{s}) = \arg\max_{a \in \mathbb{A}} \left[r(\mathbf{s}, a) + \gamma \sum_{\mathbf{s}' \in \mathbb{S}} p(\mathbf{s}' \mid \mathbf{s}, a) V^*(\mathbf{s}') \right]. \tag{A.59}$$

A pseudo-code for the VI algorithm is given in Algorithm 2.

Algorithm 2: VI algorithm

Input: State space $\mathbb{S}$, action space $\mathbb{A}$, transitions $p(s'|s,a)$, rewards $r(\mathbf{s},a)$, discount $\gamma \in (0,1)$, tolerance $\varepsilon > 0$

Output: Approximate optimal value function V^* and optimal policy π^*

Initialize $V(\mathbf{s}) \leftarrow 0$ for all $\mathbf{s} \in \mathbb{S}$;

repeat

  $\Delta \leftarrow 0$;

  foreach $\mathbf{s} \in \mathbb{S}$ **do**

    $v \leftarrow V(\mathbf{s})$;

    $V(\mathbf{s}) \leftarrow \max_{a \in \mathbb{A}} \Big(r(\mathbf{s},a) + \gamma \sum_{\mathbf{s}'} p(\mathbf{s}'|\mathbf{s},a) V(\mathbf{s}') \Big)$;

    $\Delta \leftarrow \max\{\Delta, |v - V(\mathbf{s})|\}$;

  end

until $\Delta < \varepsilon$;

Define $\pi^*(\mathbf{s}) \in \arg\max_{a \in \mathbb{A}} \Big(r(\mathbf{s},a) + \gamma \sum_{\mathbf{s}'} p(\mathbf{s}'|\mathbf{s},a) V(\mathbf{s}') \Big)$;

return V, π^*;

The VI algorithm has the following limitations:

1. It requires complete knowledge of the transition probabilities.
2. Storing these probabilities requires significant memory, thus VI is feasible only for problems with relatively few states and actions, and becomes impractical for larger ones.

To overcome the first point, one can use the Q-Learning algorithm detailed in Section A.2.7.

A.2.7 Q-Learning

To overcome the limitation of VI that requires the complete knowledge of the transition probabilities, it is possible to use RL algorithms, such as the Q-Learning [93]. This section is dedicated to the presentation of the Q-Learning algorithm. Since the transition probabilities are unknown, Q-Learning is considered as a model-free RL algorithm.

Instead of computing the state-value function as for the VI, the Q-Learning estimates the state-action value function $Q : \mathbb{S} \times \mathbb{A} \to \Re$, which is also called the Q-function and is defined as:

$$Q(\mathbf{s}, a) := \mathbb{E}_{S_1,S_2,\dots}\left[\sum_{k=0}^{+\infty} \gamma^k r(S_k, A_k) \mid S_0 = \mathbf{s}, A_0 = a\right], \tag{A.60}$$

which corresponds to the EDR starting from state $\mathbf{s}$ by taking action a.

When the actions are taken according to a deterministic policy π, we define:

$$\begin{aligned} Q^\pi(\mathbf{s}, a) &:= \mathbb{E}_{S_1,S_2,\dots}\left[\sum_{k=0}^{+\infty} \gamma^k r(S_k, A_k) \mid S_0 = \mathbf{s}, A_0 = a\right], \\ &= \mathbb{E}_{S_1,S_2,\dots}\left[r(\mathbf{s}, a) + \sum_{k=1}^{+\infty} \gamma^k r(S_k, A_k) \mid S_0 = \mathbf{s}, A_0 = a\right] \end{aligned} \tag{A.61}$$

which evaluates the long-term state-action value of choosing the action a under policy π.

Expression (A.61) can be further developed as:

$$\begin{aligned} Q^\pi(\mathbf{s}, a) &= r(\mathbf{s}, a) + \mathbb{E}_{S_1,S_2,\dots}\left[\sum_{k=1}^{+\infty} \gamma^k r(S_k, A_k) \mid S_1 = \mathbf{s}'\right] \\ &= r(\mathbf{s}, a) + \gamma \mathbb{E}_{S_1}\left[\mathbb{E}_{S_2,\dots}\left[\sum_{k=0}^{+\infty} \gamma^k r(S_{k+1}, a_{k+1}) \mid S_1 = \mathbf{s}'\right]\right] \\ &= r(\mathbf{s}, a) + \gamma \mathbb{E}_{\mathbf{s}'}\left[Q^\pi(\mathbf{s}', \pi(\mathbf{s}'))\right]. \end{aligned}$$

The optimal state value function V^* is linked to the optimal Q-value function Q^*, through the following relation:

$$V^*(\mathbf{s}) = \max_{a \in \mathbb{A}} Q^*(\mathbf{s}, a). \tag{A.62}$$

The Q-Learning estimates the optimal Q-value by iteratively solving the following equation:

$$Q^*(S_k, A_k) - \left[r(S_k, A_k) + \gamma \mathbb{E}_{S_{k+1}}\left[Q^\pi(S_{k+1}, a')\right]\right] = 0, \tag{A.63}$$

where $a' = \arg\max_{a \in \mathbb{A}_d} Q^*(S_{k+1}, a)$.

The objective is thus to learn Q^*, the optimal policy being obtained by:

$$\pi^*(\mathbf{s}) := \arg\max_{a \in \mathbb{A}} Q^*(\mathbf{s}, a). \tag{A.64}$$

The Q-function is learned through interaction with the environment. The Q-function is represented as a table, called Q-*table*, whose entries $(\mathbf{s}, a)$ correspond to the estimated Q-value for a specific state-action pair. To handle such a table, both the state space $\mathbb{S}$ and the action space $\mathbb{A}$ must be discrete and of relatively small cardinality, to keep memory requirements manageable.

The entries of the Q-table are initialized with random values. At each step k, the agent observes a state S_k and selects an action A_k based for instance on an ϵ-greedy exploration strategy. This strategy chooses a random action with probability ϵ, or the action that maximizes the Q-value, i.e. $A_k = \arg\max_{a' \in \mathbb{A}} Q(\mathbf{s}_k, a')$, with a probability $1 - \epsilon$. The ϵ-greedy strategy balances exploitation (choosing the current optimal action) with exploration (trying a random action), allowing the agent to escape from suboptimal policies.

The agent observes a reward R_{k+1}, observes the next state S_{k+1} and updates the Q-table as follows [80, (6.8)]:

$$Q(S_k, A_k) \leftarrow Q(S_k, A_k) + \alpha \left(\underbrace{R_{k+1} + \gamma \max_{a \in \mathbb{A}} Q(S_{k+1}, a) - Q(S_k, A_k)}_{\text{temporal difference}} \right), \tag{A.65}$$

where α is the learning rate. It is worth noting that the Q-Learning algorithm minimizes the temporal difference (TD), therefore the TD is null when the algorithm has converged, and thus (A.63) is verified. The theoretical convergence of the Q-Learning is guaranteed [93]. A pseudo-code of the Q-Learning algorithm is given in Algorithm 3.

Algorithm 3: Q-Learning algorithm

Input: Learning rate α, discount factor γ, exploration rate ϵ
Initialize the Q-table arbitrarily;
while *not converged* **do**
  Initialize state S_0;
  for $k = 0$ *to* K **do**
    Choose action a_k using exploration policy:
    $$A_k \leftarrow \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a Q(S_k, a) & \text{with probability } 1 - \epsilon \end{cases}$$
    Take action A_k, observe reward R_{k+1} and next state S_{k+1};
    Update Q-value:
    $Q(S_k, A_k) \leftarrow Q(S_k, A_k) + \alpha \left(R_{k+1} + \gamma \max_{a \in \mathbb{A}} Q(S_{k+1}, a) - Q(S_k, A_k) \right)$;
    Set $S_k \leftarrow S_{k+1}$;
  end
end

During inference, once the Q-function has converged, exploration is not longer used, and the agent selects the actions that maximize the Q-values.

In the training phase, Q-Learning can suffer from overestimation bias when updating the Q-values, because the estimates are learned from finite data are noisy.

To tackle this issue, double Q-Learning [94] extends the Q-Learning algorithm by maintaining two separate Q-tables, Q_1 and Q_2, which mitigates the overestimation problem. The Q-tables are randomly

selected and updated as follows:

$$Q_i(S_k, A_k) \leftarrow Q_i(S_k, A_k) + \alpha \left(\underbrace{R_k + \gamma Q_j(S_{k+1}, A_{k+1}) - Q_i(S_k, A_k)}_{\text{temporal difference}} \right), \tag{A.66}$$

where $i, j \in \{1, 2\}$ with $j \neq i$, and $A_{k+1} = \arg\max_{a \in \mathbb{A}} Q_i(S_{k+1}, a)$. When both Q-tables have converged, they both should verify (A.63).

A pseudo-code of the double Q-Learning algorithm is given in Algorithm 4.

Algorithm 4: Double Q-Learning Algorithm

Input: Learning rate α, discount factor γ, exploration rate ϵ
Initialize two Q-value functions $Q_1(\mathbf{s}, a)$ and $Q_2(\mathbf{s}, a)$ arbitrarily;
while *not converged* **do**
  Initialize state S_0;
  for $k = 0$ *to* K **do**
    Choose action a_k using exploration policy:
    $a_k \leftarrow \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a Q_1(S_k, a) + Q_2(S_k, a) & \text{with probability } 1 - \epsilon \end{cases}$;
    Take action A_k, observe reward R_{k+1} and next state S_{k+1};
    Select i randomly in $\{1, 2\}$;
    Update the value of the corresponding Q-function:
    $Q_i(S_k, A_k) \leftarrow Q_i(S_k, A_k) + \alpha \left(R_k + \gamma Q_j(S_{k+1}, A_{k+1}) - Q_i(S_k, A_k) \right)$;
    where $A_{k+1} = \arg\max_{a \in \mathbb{A}} Q_i(S_{k+1}, a)$, and $j \neq i$;
    Set $S_k \leftarrow S_{k+1}$;
  end
end

Note that the following identity holds:

$$\max_a Q(S_{k+1}, a) = Q(S_{k+1}, \arg\max_{a \in \mathbb{A}} Q(S_{k+1}, a)). \tag{A.67}$$

Using this property, both Q-Learning and double Q-Learning can be expressed in the same update form:

$$Q_i \leftarrow Q_i(S_k, A_k) + \alpha \left(R_{k+1} + \gamma Q_j(S_{k+1}, a') - Q_i(S_k, A_k) \right), \tag{A.68}$$

where $a' = \arg\max_a Q_i(S_{k+1}, a)$ and:

$$\begin{cases} i = j & \text{for } Q\text{-Learning} \\ i \neq j & \text{for double } Q\text{-Learning.} \end{cases} \tag{A.69}$$

A.3 Deep reinforcement learning

When the number of states is large, it is not practically possible to store a table with the Q-values related to all the different states. This phenomenon is known as the *curse of dimensionality*. This problem can be handled through the use of function approximator such as DNN, yielding to DRL. There exists a lot of

DRL algorithms, see for instance [95]. This thesis focuses on the DQL algorithm and DDQL, which are extensions of Q-Learning and double Q-Learning respectively, using DNN. Different DNN architectures can be envisioned. Some of them are reviewed in Section A.4. The DNN are trained by minimizing a loss function.

In the DQL, the Q-function is approximated by a DNN called DQN. The DQN can be represented by a function parametrized by θ, which takes state $\mathbf{s} \in \mathbb{S}$ as inputs and outputs the Q-values for the different actions. Let $Q(\mathbf{s}, a; \theta)$ be the Q-values of state $\mathbf{s}$, action a obtained by the DNN of parameters θ. As seen in (A.65), the Q-function aims to minimize the TD. Hence, the loss used to train the DQN is:

$$\tilde{\mathcal{L}}(\theta) = \Big(r_k + \gamma \max_a Q(\mathbf{s}_{k+1}, a; \theta) - Q(\mathbf{s}_k, a_k; \theta)\Big)^2. \tag{A.70}$$

Then, this loss is used to update the weights of the DQN:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \tilde{\mathcal{L}}(\theta), \tag{A.71}$$

where α is the learning rate.

In practice, using (A.70) may lead to unstable results because of:

1. Correlations present in the sequence of states (updating the weights of the DQN at each time step in the same way as for a Q-table).
2. Correlation presents between the state-action value $Q(\mathbf{s}_k, a_k)$ and the target value, defined as:

$$r_k + \gamma \max_{a \in \mathbb{A}} Q(\mathbf{s}_{k+1}, a, \theta). \tag{A.72}$$

   This signifies that when the Q-value is updated to get closer of the target, the target also moves, leading to instabilities.

To address these issues, [3] introduces:

1. Experience replay is used to reduce the correlation between consecutive observations. It involves leveraging past transitions, known as experiences, to update the DNN. A transition at step k is represented by the tuple $(\mathbf{s}_k, a_k, r_k, \mathbf{s}_{k+1})$ (also written as $(\mathbf{s}, a, r, \mathbf{s}')$). These transitions are stored in a buffer $\mathcal{B}$, called *replay buffer*, which accumulates experiences collected during training.
2. A target network, parameterized by θ^-, is used as a delayed copy of θ. It is updated periodically or gradually (via soft updates), which helps reducing the correlation between the predicted Q-values and the target values, thereby stabilizing the learning process.

The DQN is trained by minimizing the TD, which serves as loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{s},a,r,\mathbf{s}')\sim\mathcal{B}}\left[(y - Q(\mathbf{s}, a; \theta))^2\right], \tag{A.73}$$

where $y := r + \gamma \max_a Q(\mathbf{s}, a; \theta^-)$ is the target value.

In practice, the DQN is trained as stated in Algorithm 5 and summarized as follows:

- The agent explores the state and action spaces using an ϵ-greedy exploration policy. It receives, as for the Q-Learning, a reward and the next state. However, instead of using the current state $\mathbf{s}_k$, the taken action a_k, the reward r_k and the next state $\mathbf{s}_{k+1}$ to update directly the DQN (as for the Q-Learning), this information is stored in the replay buffer $\mathcal{B}$. The set $(\mathbf{s}_k, a_k, r_k, \mathbf{s}_{k+1})$ is referred to as an experience.
- Once the replay buffer contains enough experiences, a batch of b experiences is randomly sampled to approximate (A.73) to train the DQN.
  Let $j \in \{1, \ldots, b\}$ index the sampled experiences. The TD is the difference between the evaluated Q-value $Q(\mathbf{s}_j, a_j)$ and the target value y_j which is defined as:

$$y_j := r_j + \gamma \max_a Q(\mathbf{s}_{j+1}, a; \theta^-), \tag{A.74}$$

and the approximated loss by:

$$\widehat{\mathcal{L}}(\theta) = \frac{1}{b}\sum_{j=1}^{b}\left(y_j - Q(\mathbf{s}_j, a_j; \theta)\right)^2. \tag{A.75}$$

It is worth noting that (A.70) corresponds to a loss with a single sample and without a target network, whereas (A.75) and (A.73) both include multiple samples, thanks to the replay buffer, as well as the target network.

- Then this loss is used to update the weights θ,

$$\theta \leftarrow \theta - \alpha\nabla_\theta\widehat{\mathcal{L}}(\theta). \tag{A.76}$$

- The target network is updated every C steps: $\theta^- \leftarrow \theta$. With the soft-update approach, the target weights are updated as $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$.

Algorithm 5: Deep Q-Learning (DQL)

```
Initialize replay memory B;
Initialize Q-network with random weights θ;
Initialize target Q-network with weights θ⁻ = θ;
for each episode do
    Initialize state s_0;
    for k = 0 to K do
        Choose action a_k using ε-greedy policy based on Q-network;
        Execute action a_k and observe reward r_k and next state s_{k+1};
        Store experience (s_k, a_k, r_k, s_{k+1}) in B;
        Sample random minibatch of experiences (s_j, a_j, r_j, s_{j+1}) from B;
        y_j = r_j + γ max_a Q(s_{j+1}, a, θ⁻);
        Compute the loss with (A.75) ;
        θ ← θ − α∇_θ L̂(θ);
        Update target Q-network every C steps;
    end
    Update ε according to schedule;
end
```

The DQL has been successfully used in [3] to play Atari games. However, the DQL produces overestimation for the same reasons as the Q-Learning does, and therefore, the DDQL was introduced in [96] to mitigate this phenomenum. In that case, the target value y_k becomes:

$$y_k = r_k + \gamma Q\left(\mathbf{s}_{k+1}, \arg\max_a Q(\mathbf{s}_{k+1}, a, \theta), \theta^-\right). \tag{A.77}$$

The rest remains the same as for the DQL algorithm.

One can remark that the general form of the target value can be written as:

$$y_k = r_k + \gamma Q\left(\mathbf{s}_{k+1}, a, \theta^-\right), \tag{A.78}$$

where

$$a = \begin{cases} \arg\max_{a'} Q\left(\mathbf{s}_{k+1}, a', \theta^-\right) & \text{for deep } Q\text{-Learning} \\ \arg\max_{a'} Q\left(\mathbf{s}_{k+1}, a', \theta\right) & \text{for double deep } Q\text{-Learning} \end{cases} \tag{A.79}$$

Algorithm 6: Double Deep Q-Learning (DDQL)

Initialize replay memory $\mathcal{B}$;
Initialize Q-network with random weights θ;
Initialize target Q-network with weights $\theta^- = \theta$;
for *each episode* **do**
  Initialize state $\mathbf{s}_0$;
  for *k = 0 to K* **do**
    Choose action a_k using ϵ-greedy policy based on Q-network;
    Execute action a_k and observe reward r_k and next state $\mathbf{s}_{k+1}$;
    Store experience $(\mathbf{s}_k, a_k, r_k, \mathbf{s}_{k+1})$ in $\mathcal{B}$;
    Sample random minibatch of experiences $(\mathbf{s}_j, a_j, r_j, \mathbf{s}_{j+1})$ from $\mathcal{B}$;
    $a \leftarrow \arg\max_{a'} Q(\mathbf{s}_{j+1}, a', \theta)$;
    $y_j \leftarrow r_j + \gamma Q(\mathbf{s}_{j+1}, a, \theta^-)$;
    Compute the loss ;
    $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$;
    Update target Q-network every C steps;
  end
  Update ϵ according to schedule;
end

A.4 Deep neural network architectures

This section introduces the different DNN architectures used throughout this thesis.

DNNs work as function approximator that seek to predict an output y based on a given input $\boldsymbol{x}$. The context with known (labeled) outputs y is referred to as "supervised learning" [97]. When y is unknown, the DNN can be trained either with unsupervised learning, or DRL, which is the focus of this thesis. The DNN architectures presented in this section are:

- The FC in Section A.4.1.
- The transformer in Section A.4.2.
- The AB in Section A.4.3.

A.4.1 Fully connected architecture

A FC DNN consists of multiple layers where each neuron in one layer is connected to each neuron in the subsequent layer.

For a given neuron j in the lth layer, the output $y_j^{(l)}$ is computed as:

$$y_j^{(l)} := \sigma^{(l)} \left(\sum_i w_{ij}^{(l)} y_i^{(l-1)} + b_j^{(l)} \right), \tag{A.80}$$

where:

- $w_{ij}^{(l)}$ are the weights connecting neuron i from the previous layer to neuron j in the current layer l.
- $b_j^{(l)}$ is the bias term for neuron j in the current layer l.
- $\sigma^{(l)}$ is the activation function applied to introduce non-linearity for the layer l.

Equation (A.80) can be re-expressed in matrix form as:

$$\boldsymbol{y}^{(l)} := \Phi_l\left(\boldsymbol{y}^{(l-1)}\right) \tag{A.81}$$

where

- $\Phi_l(\boldsymbol{x}) := \sigma^{(l)}\left(\boldsymbol{W}^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)}\right)$, $\Phi_l : \Re^{d_{l-1}} \to \Re^{d_l}$.
- $\boldsymbol{W}^{(l)} := \left(w_{ij}^{(l)}\right)$, $\boldsymbol{W}^{(l)} \in \Re^{d_l \times d_{l-1}}$.
- $\boldsymbol{y}^{(l)} := \left(y_j^{(l)}\right)$, $\boldsymbol{y}^{(l)} \in \Re^{d_l}$. It is worth noting that $\boldsymbol{y}^{(0)} = \boldsymbol{x}$ where $\boldsymbol{x}$ is the input vector.
- $\boldsymbol{b}^{(l)} := \left(b_j^{(l)}\right)$, $\boldsymbol{b}^{(l)} \in \Re^{d_l}$.

The output of the neural network can be represented as:

$$\boldsymbol{y} = \Phi(\boldsymbol{x}), \tag{A.82}$$

$$\Phi(\boldsymbol{x}) := \Phi_N \circ \Phi_{N-1} \circ \cdots \circ \Phi_1(\boldsymbol{x}). \tag{A.83}$$

For example, considering the DNN illustrated in Figure A.4, we identify:

$$\boldsymbol{y} = \Phi(\boldsymbol{x}) = \boldsymbol{W}^{(2)}\sigma^{(1)}\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}, \tag{A.84}$$

where $\boldsymbol{y} = [y_1, y_2]^T$, $\boldsymbol{x} = [x_1, x_2]^T$, $\boldsymbol{b}^{(1)} = [b_1^{(1)}, b_2^{(1)}, b_3^{(1)}]^T$, $\boldsymbol{b}^{(2)} = [b_1^{(2)}, b_2^{(2)}]^T$, $\boldsymbol{W}^{(1)} = \left(w_{ij}^{(1)}\right)_{\substack{i\in\{1,2,3\}\\ j\in\{1,2\}}}$ and $\boldsymbol{W}^{(2)} = \left(w_{ij}^{(2)}\right)_{\substack{i\in\{1,2\}\\ j\in\{1,2,3\}}}$. In this figure, the obtained vector after the first layer is $\boldsymbol{y}^{(1)} = \boldsymbol{W}^{(1)}\boldsymbol{x} = [y_1^{(1)}, y_2^{(1)}]$.

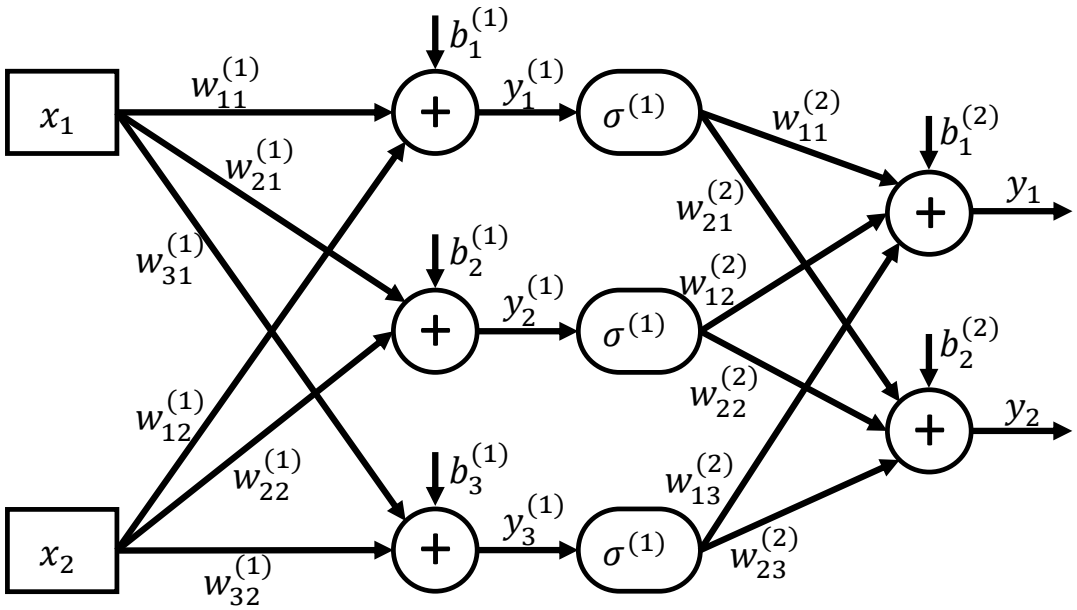


Figure A.4: Example of neural network.

A.4.2 Transformer architecture

The transformer architecture, introduced in [73], relies on the attention mechanism. The original transformer is composed by two different parts: an encoder and a decoder. An encoder and a decoder,

that can be used either jointly or separately, meaning that it is possible to use EOT or decoder only transformer (DOT). The transformer architecture has shown powerful performance:

- In natural language processing (NLP) field for:
  - Text translation [73].
  - Text classification and masked word prediction [98] with the bidirectional encoder representations from transformer (BERT) architecture, which is an EOT.
  - Text generation [99] with the generative pre-trained transformer (GPT) architecture, which is a DOT.
- In the computer vision (CV) field for image recognition, object detection, segmentation, image generation, image synthesis [100].
- In DRL field for a sequence modeling problem [101], [102], [103]. A survey for DRL application of transformer can be found in [104].

The attention mechanism helps the architecture to focus on certain elements of the set which are relevant for the prediction.

It is worth noting that the attention mechanism was introduced before the transformer architecture, originally in the NLP field for text translation using recurrent neural network (RNN)s such as LSTMs. The first instance, known as content-based attention, was proposed in [105], while additive attention, which involves a concatenation operation, was introduced in [106].

Then, [107] proposed another form of the attention mechanism, called dot-product attention, which was improved in [73] that proposed scaled-dot-product attention yielding transformer. The authors also introduced the multi-head attention reducing the training time, the inference time and improving the generalization. A review of the different attention mechanisms can be found in [108].

Let us describe the attention mechanism. For that, let us consider a matrix $\boldsymbol{X}$ of dimension $d_e \times n_L$ where each column $\boldsymbol{x}_\ell$ represents an element of the input set, containing n_L elements each represented by a vector of dimension $d_e \times 1$. For instance, in NLP, $\boldsymbol{x}_\ell$ can be a word or a token of the input sentence that belongs to a vocabulary set, which is the input set, and, to be more general, $\boldsymbol{x}_\ell$ belongs to a vocabulary set. The attention mechanism is illustrated in Figure A.5 and works as follows: first the $\boldsymbol{x}_\ell$ are projected into:

- *keys*: $\boldsymbol{k}_\ell := \boldsymbol{W_k}\boldsymbol{x}_\ell + \boldsymbol{b_k}$ for each ℓ, where $\boldsymbol{k}_\ell$ is a vector of dimension d_{attn}, $\boldsymbol{W_k}$ is of dimension $d_{\text{attn}} \times d_e$ and $\boldsymbol{b_k}$ is a vector of dimension d_{attn}. The elements projected into keys represent the context.
- *queries*: $\boldsymbol{q}_\ell := \boldsymbol{W_q}\boldsymbol{x}_\ell + \boldsymbol{b_q}$ for each ℓ, where $\boldsymbol{k}_\ell$ is a vector of dimension d_{attn}, $\boldsymbol{W_q}$ is of dimension $d_{\text{attn}} \times d_e$ and $\boldsymbol{b_q}$ is a vector of dimension d_{attn}. This allows to find the relevant elements of the set, for example in translation task to find the relevant word in a language to translate it correctly.
- *values*: $\boldsymbol{v}_\ell := \boldsymbol{W_v}\boldsymbol{x}_\ell + \boldsymbol{b_v}$ for each ℓ, where $\boldsymbol{k}_\ell$ is a vector of dimension d_v, $\boldsymbol{W_v}$ is of dimension $d_v \times d_e$ and $\boldsymbol{b_v}$ is a vector of dimension d_v. This holds the actual information that corresponds to each element.

$\boldsymbol{W_k}$, $\boldsymbol{W_q}$ and $\boldsymbol{W_v}$ are trainable weights matrices and $\boldsymbol{b_k}$, $\boldsymbol{b_q}$ and $\boldsymbol{b_v}$ are trainable biases. Let us note $\boldsymbol{Q} := [\boldsymbol{q}_1, \dots, \boldsymbol{q}_{n_L}]$, $\boldsymbol{K} := [\boldsymbol{k}_1, \dots, \boldsymbol{k}_{n_L}]$ and $\boldsymbol{V} := [\boldsymbol{v}_1, \dots, \boldsymbol{v}_{n_L}]$ the matrix of queries, keys and values respectively. $\boldsymbol{Q}$ and $\boldsymbol{K}$ are of dimension $d_{\text{attn}} \times n_L$ and $\boldsymbol{V}$ is $d_v \times n_L$.

The keys are compared with the queries thanks to the scale dot-product attention to determine how relevant each element is, i.e. what elements of the context are relevant for the different queries. Then, the inner-product between the queries and the keys is performed, producing a n_L-squared matrix $\boldsymbol{S}$, called *score matrix*:

$$\boldsymbol{S} = \boldsymbol{K}^T\boldsymbol{Q}. \tag{A.85}$$

Each entry $\boldsymbol{S}_{ij}$ represents the score between the key i and the query j. The higher this score, the most

relevant the information of $\boldsymbol{x}_i$ with respect to $\boldsymbol{x}_j$.

After that, the softmax function is applied producing the attention matrix $\boldsymbol{A}$:

$$\boldsymbol{A} := \operatorname{softmax}(\boldsymbol{S}). \tag{A.86}$$

The softmax function is defined column-wise ($\boldsymbol{A}_{ij} = \frac{\exp(\boldsymbol{S}_{ij})}{\sum_{j'} \exp(\boldsymbol{S}_{ij'})}$), producing weights between 0 and 1. The softmax function can lead to very small gradients due to the potential high magnitude of the scores [73].

To mitigate this effect, the inner product is divided by d_{attn}:

$$\widetilde{\boldsymbol{S}} = \frac{\boldsymbol{S}}{\sqrt{d_{attn}}}. \tag{A.87}$$

Therefore, the softmax is applied on $\widetilde{\boldsymbol{S}}$ instead of $\boldsymbol{S}$.

Finally, the values are summed for each element of the set according to their weights:

$$\widetilde{\boldsymbol{V}} = \boldsymbol{V}\boldsymbol{A}. \tag{A.88}$$

$\widetilde{\boldsymbol{V}}$ is of dimension $d_v \times n_L$ and is the result of the scaled dot-product attention. For single head-attention, $d_v = d_e$. These operations are illustrated in Figure A.5 and are summed up in the Algorithm 7. We consider that the attention is performed in one set only, such as described in the equation above.

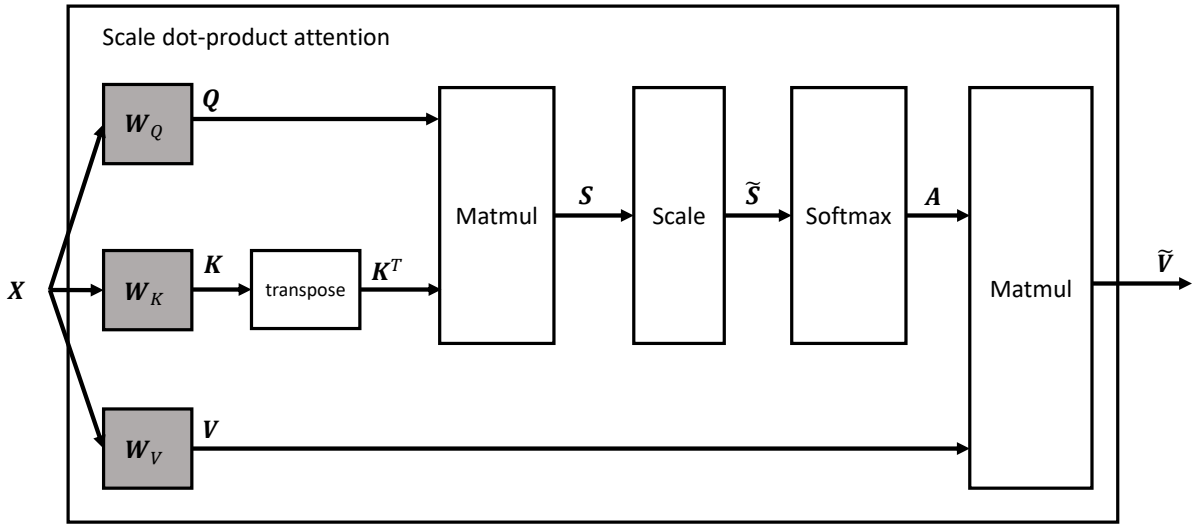


Figure A.5: Scale dot-product attention (gray boxes are trainable weights).

In Algorithm 7, $\mathbf{1}$ denotes a column vector of size n_L with all entries equal to one. Therefore, the product $\boldsymbol{b}\mathbf{1}^T$, where $\boldsymbol{b}$ is a vector bias of size d, is a matrix of size $d \times n_L$, where each column is equal to $\boldsymbol{b}$.

Algorithm 7: $\widetilde{\boldsymbol{V}} \leftarrow \texttt{Attention}(\boldsymbol{X} \mid \boldsymbol{\mathcal{W}_{qkv}})$

```
/* Computes a single self-attention head.                    */
```

Input: $\boldsymbol{X} \in \Re^{d_e \times n_L}$, representations of a set.

Output: $\widetilde{\boldsymbol{V}} \in \Re^{d_{\text{out}} \times \ell_{\text{x}_2}}$, updated representations of the input set $\boldsymbol{X}$.

Parameters: $\boldsymbol{\mathcal{W}_{qkv}}$ consisting of: $\boldsymbol{W_q} \in \Re^{d_{\text{attn}} \times d_e}$, $\boldsymbol{b_q} \in \Re^{d_{\text{attn}}}$ $\boldsymbol{W_k} \in \Re^{d_{\text{attn}} \times d_e}$, $\boldsymbol{b_k} \in \Re^{d_{\text{attn}}}$ $\boldsymbol{W_v} \in \Re^{d_v \times d_e}$, $\boldsymbol{b_v} \in \Re^{d_{\text{out}}}$.

$\boldsymbol{Q} \leftarrow \boldsymbol{W_q X} + \boldsymbol{b_q} \boldsymbol{1}^T \quad [\![\boldsymbol{Q}\text{uery} \in \Re^{d_{\text{attn}} \times \ell_{\text{x}}}]\!]$

$\boldsymbol{K} \leftarrow \boldsymbol{W_k X} + \boldsymbol{b_k} \boldsymbol{1}^T \quad [\![\boldsymbol{K}\text{ey} \in \Re^{d_{\text{attn}} \times \ell_{\text{x}}}]\!]$

$\boldsymbol{V} \leftarrow \boldsymbol{W_v X} + \boldsymbol{b_v} \boldsymbol{1}^T \quad [\![\boldsymbol{V}\text{alue} \in \Re^{d_{\text{out}} \times \ell_{\text{x}}}]\!]$

$\boldsymbol{S} \leftarrow \boldsymbol{K}^T \boldsymbol{Q} \quad [\![\boldsymbol{S}\text{core} \in \Re^{\ell_{\text{x}} \times \ell_{\text{x}}}]\!]$

$\boldsymbol{A} \leftarrow \text{softmax}\left(\frac{\boldsymbol{S}}{\sqrt{d_{attn}}}\right)$

return $\widetilde{\boldsymbol{V}} = \boldsymbol{V A}$

For multi-head attention, which is illustrated in Figure A.6 and summarized in Algorithm 8, the elements of the input set are projected into *keys*, *queries* and *values* on H heads, and the scaled-dot product is performed on each of these H heads, resulting on $\widetilde{\boldsymbol{V}}^h$ on head h.

The results on the different heads are combined by concatenated them:

$$\underline{\boldsymbol{V}} = [\widetilde{\boldsymbol{V}}^1, \ldots, \widetilde{\boldsymbol{V}}^H], \tag{A.89}$$

and by applying a final projection:

$$\widetilde{\boldsymbol{V}} = \boldsymbol{W_o} \underline{\boldsymbol{V}} + \boldsymbol{b_o} \boldsymbol{1}^T, \tag{A.90}$$

where $\boldsymbol{W_o}$ is a $d_e \times H d_v$ trainable matrix and $\widetilde{\boldsymbol{V}}$ is a $d_e \times n_L$ matrix representing the different elements with their context, i.e. with the information of the other elements.

Let us note $\boldsymbol{\mathcal{W}}$ the entire set of parameters (*query*, *key*, *value* and output linear projections) required by a multi-head attention layer:

$$\boldsymbol{\mathcal{W}} := \begin{pmatrix} \boldsymbol{W_q^h} \in \Re^{d_{\text{attn}} \times d_e}, & \boldsymbol{b_q^h} \in \Re^{d_{\text{attn}}}, & h \in [H] \\ \boldsymbol{W_k^h} \in \Re^{d_{\text{attn}} \times d_e}, & \boldsymbol{b_k^h} \in \Re^{d_{\text{attn}}}, & h \in [H] \\ \boldsymbol{W_v^h} \in \Re^{d_v \times d_e}, & \boldsymbol{b_v^h} \in \Re^{d_v}, & h \in [H] \\ \boldsymbol{W_o} \in \Re^{d_e \times H d_v}, & \boldsymbol{b_o} \in \Re^{d_e} & \end{pmatrix} \tag{A.91}$$

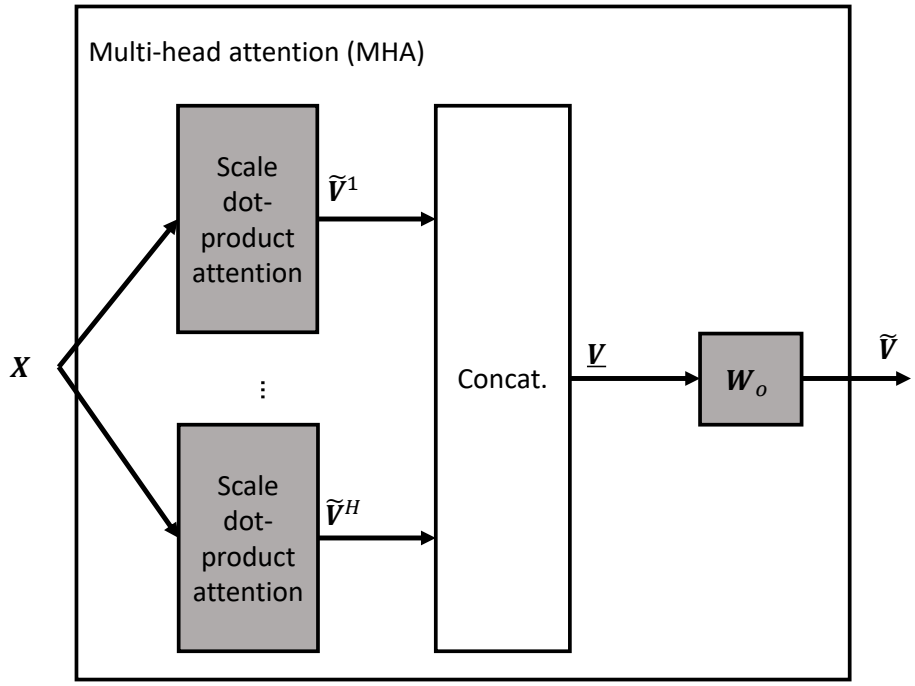


Figure A.6: Multi-head attention (gray boxes contain trainable weights).

Algorithm 8: $\widetilde{\boldsymbol{V}} \leftarrow \texttt{MHAttention}(\boldsymbol{X} \mid \boldsymbol{\mathcal{W}})$

`/* Computes Multi-Head self-attention layer. */`

Input: $\boldsymbol{X} \in \Re^{d_e \times n_L}$, representation of a set.

Output: $\widetilde{\boldsymbol{V}} \in \Re^{d_v \times n_L}$, updated representations of the input set $\boldsymbol{X}$.

Hyperparameters: H, number of attention heads

Parameters: $\boldsymbol{\mathcal{W}}$ consisting of

$\forall h \in [H]$, $\boldsymbol{\mathcal{W}}_{\boldsymbol{qkv}}^h$ consisting of:

$\boldsymbol{W}_{\boldsymbol{q}}^h \in \Re^{d_{\text{attn}} \times d_e}$, $\boldsymbol{b}_{\boldsymbol{q}}^h \in \Re^{d_{\text{attn}}}$,

$\boldsymbol{W}_{\boldsymbol{k}}^h \in \Re^{d_{\text{attn}} \times d_e}$, $\boldsymbol{b}_{\boldsymbol{k}}^h \in \Re^{d_{\text{attn}}}$,

$\boldsymbol{W}_{\boldsymbol{v}}^h \in \Re^{d_v \times d_e}$, $\boldsymbol{b}_{\boldsymbol{v}}^h \in \Re^{d_v}$.

$\boldsymbol{W}_{\boldsymbol{o}} \in \Re^{d_e \times H d_v}$, $\boldsymbol{b}_{\boldsymbol{o}} \in \Re^{d_e}$.

for $h \in [H]$ **do**

  $\widetilde{\boldsymbol{V}}^h \leftarrow \texttt{Attention}(\boldsymbol{X} \mid \boldsymbol{\mathcal{W}}_{\boldsymbol{qkv}}^h)$

end

$\underline{\boldsymbol{V}} \leftarrow [\widetilde{\boldsymbol{V}}^1; \widetilde{\boldsymbol{V}}^2; \ldots; \widetilde{\boldsymbol{V}}^H]$

return $\widetilde{\boldsymbol{V}} = \boldsymbol{W}_{\boldsymbol{o}} \underline{\boldsymbol{V}} + \boldsymbol{b}_{\boldsymbol{o}} \boldsymbol{1}^T$

There exist multiple manners to use the attention, depending on the task:

- Bi-directional self-attention (also called unmasked self-attention), used in the BERT architecture [98], and in general in the transformer-encoder. The attention is performed on $\boldsymbol{X}$ and, for an element $\boldsymbol{x}_\ell$ projected into queries, the atttention is performed on all elements $\{\boldsymbol{x}_{\ell'}\}$. This is the type of attention used in this thesis.
- Unidirectional self-attention (also called masked or left-only attention), used in the transformer-decoder for the generated sequence.The attention is only performed on the input matrix $\boldsymbol{X}$, and for

an element $\boldsymbol{x}_\ell$ projected into *queries*, the attention is performed on all elements $\{\boldsymbol{x}_{\ell'}\}_{\ell' \leq \ell}$, which are projected into *keys* and *values*.

- Cross-attention is utilized in the transformer-decoder to compare two distinct sets, $\boldsymbol{X}$ and $\boldsymbol{Z}$. For example, in a translation task, the first set $\boldsymbol{X}$ corresponds to the generated translated sequence, while the second set $\boldsymbol{Z}$ represents the text to be translated. In this process, the elements $\boldsymbol{x}_\ell$ from $\boldsymbol{X}$ are transformed into *queries*, and the elements $\{\boldsymbol{z}_{\ell'}\}$ from $\boldsymbol{Z}$ are converted into *keys* and *values*.

Most of the used transformers utilize the positional encoding (also called positional embedding) to model a sequence. Without the positional encoding, the transformer is PE (property of the attention mechanism) [74], i.e. for every permutation on the inputs, the output undergoes the same permutation.

In this thesis, we use EOT and thus the decoder part of the transformer is not detailed. The EOT is depicted in Figure A.7 and works as follows for an input set $\boldsymbol{X} \in \Re^{d_e \times n_L}$:

- The set passes in the multi-head attention (MHA) layer of the EOT, resulting in $\widetilde{\boldsymbol{V}}$ the transformed input set where each element has its context.
- The original value of the different elements is added to the corresponding transformed element $(\boldsymbol{x}_\ell + \tilde{\boldsymbol{v}}_\ell)$, corresponding to the residual connection operation [109]. This operation improves the training and the gradient propagation, particularly by mitigating the vanishing gradient problem.
- This result passes in the layer norm [110], resulting in $\widehat{\boldsymbol{X}}$. The layer normalization works for a vector $\boldsymbol{u} = [u_1, \ldots, u_{d_e}]^T$ of size d_e as follows:
  1. The mean value μ of the vector $\boldsymbol{u}$ is computed:

$$\mu = \frac{1}{d_e} \sum_{i=1}^{d_e} u_i \tag{A.92}$$

  2. The variance σ^2 is then computed:

$$\sigma^2 = \frac{1}{d_e} \sum_{i=1}^{d_e} (u_i - \mu)^2 \tag{A.93}$$

  3. Then the layer normalization on the vector $\boldsymbol{u}$ is performed, leading to vector $\hat{\boldsymbol{u}}$:

$$\hat{\boldsymbol{u}} = \boldsymbol{\gamma} \frac{\boldsymbol{u} - \mu}{\sigma} + \boldsymbol{\beta} \tag{A.94}$$

$$= \text{layer\_norm}(\boldsymbol{u} \mid \boldsymbol{\gamma}, \boldsymbol{\beta}) \tag{A.95}$$

  where $\boldsymbol{\gamma}$ is a diagonal matrix with dimension d_e to scale the normalization and $\boldsymbol{\beta}$ is an offet vector of dimension d_e Both $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are trainable parameters.
- Each normalized element passes in a feed forward network (FFN) consisting of two layers with a non-linear activation function between them, e.g. ReLU activation function. The first layer projects from dimension d_e into dimension d_{mlp} and the second layer projects from dimension d_{mlp} into dimension d_e. This operation results in the representation $\widehat{\boldsymbol{X}}$.
- The residual connection and the layer normalization are performed again, obtaining the encoded set $\boldsymbol{E} \in \Re^{d_e \times n_L}$ as the output.

It is essential to note that transformer layers can be stacked. Specifically, an encoded representation of $\boldsymbol{X}$ denoted as $\boldsymbol{E}$ can be processed through additional EOT layers. This sequence results in n_{eot} transformations, producing representations within a $\Re^{d_e \times n_L}$ space. For simplicity, we refer to $\boldsymbol{E}$ as the result of all transformations applied by the EOT layers.

The Algorithm 9 outlines the operation of the EOT architecture. This algorithm is adapted from [111].

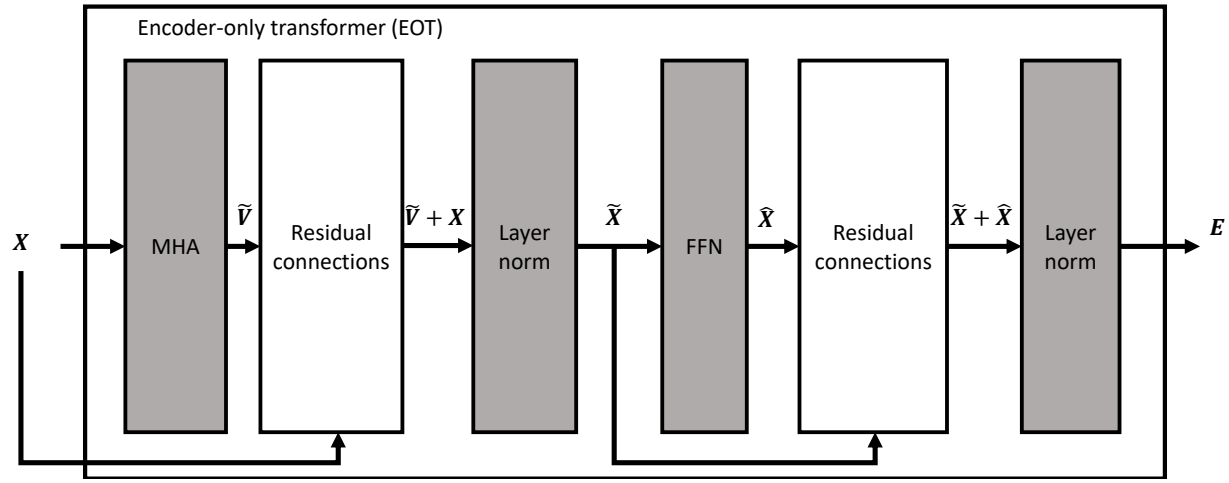


Figure A.7: Encoder-only transformer (gray boxes contain trainable weights).

Algorithm 9: $\boldsymbol{E} \leftarrow \texttt{EOT}(\boldsymbol{X} \mid \boldsymbol{\theta})$

`/* EOT forward pass */`

Input: $\boldsymbol{X} \in \Re^{d_\text{e} \times n_L}$.

Output: $\hat{\boldsymbol{X}} \in \Re^{d_\text{e} \times n_L}$.

Hyperparameters: $n_\text{eot}, H, d_\text{e}, d_\text{mlp} \in \mathbb{N}$

Parameters: $\boldsymbol{\theta}$ includes all of the following parameters:

$\forall t \in \{1, \ldots, n_\text{eot}\}$:

| $\boldsymbol{\mathcal{W}}_t$, multi-head attention parameters for layer t, see (A.91),

| $\boldsymbol{\gamma}_t^1, \boldsymbol{\beta}_t^1, \boldsymbol{\gamma}_t^2, \boldsymbol{\beta}_t^2 \in \mathbb{R}^{d_\text{e}}$, two sets of layer-norm parameters,

| $\boldsymbol{W}_\text{mlp1}^t \in \mathbb{R}^{d_\text{mlp} \times d_\text{e}}$, $\boldsymbol{W}_\text{mlp2}^t \in \mathbb{R}^{d_\text{e} \times d_\text{mlp}}$, $\boldsymbol{b}_\text{mlp1}^t \in \mathbb{R}^{d_\text{mlp}}$, $\boldsymbol{b}_\text{mlp2}^t \in \mathbb{R}^{d_\text{e}}$, FFN parameters.

for $t = 1, 2, \ldots, n_\text{eot}$ **do**

$\boldsymbol{X} \leftarrow \boldsymbol{X} + \texttt{MHAttention}(\boldsymbol{X} \mid \boldsymbol{\mathcal{W}}_t)$

for $\ell \in [n_L]$: $\boldsymbol{x}_\ell \leftarrow \texttt{layer\_norm}(\boldsymbol{x}_\ell \mid \boldsymbol{\gamma}_t^1, \boldsymbol{\beta}_t^1)$ `/* Defined in` (A.95) `*/`

$\boldsymbol{X} \leftarrow \boldsymbol{X} + \boldsymbol{W}_\text{mlp2}^t \texttt{RELU}(\boldsymbol{W}_\text{mlp1}^t \boldsymbol{X} + \boldsymbol{b}_\text{mlp1}^t \mathbf{1}^T) + \boldsymbol{b}_\text{mlp2}^t \mathbf{1}^T$

for $\ell \in [n_L]$: $\hat{\boldsymbol{x}}_\ell \leftarrow \texttt{layer\_norm}(\boldsymbol{x}_\ell \mid \boldsymbol{\gamma}_t^2, \boldsymbol{\beta}_t^2)$

end

return $\boldsymbol{E}$

The computational complexity of one EOT layer is outlined in Table A.1. Bias is considered, as well as the ReLU activation of the FFN. However, transposition operations are not included in the calculations. To have the total complexity of the whole EOT, the result of Table A.1 must be multiplied by n_eot.

Table A.1: Computational complexity of one EOT layer.

Operation	FLOPs
Query, key projections	$2 \times 2 \times Hd_{\text{attn}} \times d_e \times n_L$
Value projections	$2 \times Hd_v \times d_e \times n_L$
(A.85)	$2 \times n_L \times Hd_{\text{attn}} \times n_L - n_L \times n_L$
(A.86) and (A.87)	$4 \times H \times n_L \times n_L$
(A.88)	$2 \times Hd_v \times n_L \times n_L - n_L Hd_v$
(A.90)	$2 \times d_e \times Hd_v \times n_L$
Layer norm	$2 \times 8 \times d_e \times n_L$
Residual connections	$2 \times d_e \times n_L$
FFN	$2 \times 2 \times d_e \times d_{\text{mlp}} \times n_L$
Total	$2n_L \left[d_e(2Hd_{\text{attn}} + 2Hd_v + 2d_{\text{mlp}} + 9)\right.$ $\left.+2Hn_L + Hd_{\text{attn}}n_L + Hd_v n_L\right] - n_L^2 - n_L Hd_v$
Total if $d_e = Hd_{\text{attn}} = Hd_v$	$2n_L \left(d_e(4d_e + 2n_L + 2d_{\text{mlp}} + 9) + 2Hn_L\right) - n_L^2 - n_L d_e$

A.4.3 Action branching architecture

The AB architecture was introduced in [77] for the DRL field. Its objective is to address setups where the number of actions is large and cannot be handle by conventional architectures, by dividing the action space into multiple branches. This approach is generally applicable to tasks where the action space $\mathbb{A}$ has a high cardinality and can be decomposed as: $\mathbb{A} = \mathbb{A}_1 \times \cdots \times \mathbb{A}_{N_D}$, meaning that the action $\boldsymbol{a} \in \mathbb{A}$ can be written as $\boldsymbol{a} = [a_1, \ldots, a_{N_D}]$, where $a_d \in \mathbb{A}_d$ is called a sub-action and can take n_d possible values. Each branch of the AB architecture corresponds to one of these sub-action spaces, with action selection taking place within each branch.

The cardinality of the original action space $\mathbb{A}$ is given by $\text{card}(\mathbb{A}) = N$ and thanks to the action space decomposition, we can write $N = \prod_{d=1}^{N_D} \text{card}(\mathbb{A}_d) = \prod_{d=1}^{N_D} n_d$. If the number of possible actions is the same for each dimension, meaning $n_d = n$ for all $d \in \{1, \ldots, N_D\}$, we have $N = n^{N_D}$. By implementing action branching, selection is performed on each branch, resulting in N_D separate selections, each with n action possibilities. Instead of computing an argmax over n^{N_D} possible actions, which becomes impractical when n^{N_D} is large, this approach simplifies the process by performing $n \times N_D$ argmax operations, provided that n remains reasonably small.

The action branching, represented in Figure A.8 and described in Algorithm 10, works as follows:

1. The state $\mathbf{s}$ goes in a first DNN of parameters θ_0 that produces a shared representation of the state for all branches. This shared representation is noted $\widetilde{\mathbf{s}}$:

$$\widetilde{\mathbf{s}} = \Phi_0(\mathbf{s}, \theta_0) \tag{A.96}$$

2. The original AB architecture utilizes dueling [112], i.e. it splits the Q-values as the sum of the state value V, which is a scalar, and the advantage of the different actions $\boldsymbol{A}$, which is a vector:
   - The shared representation goes in a DNN of parameters θ_V that produces the state value V:

$$V = \Phi_V(\widetilde{\mathbf{s}}, \theta_V) \tag{A.97}$$

- For each branch d, the shared representation goes in a DNN of parameters θ_{A^d}, producing the advantage $\boldsymbol{A}^d$ of the different actions of branch d. Therefore, $\boldsymbol{A}^d$ is a vector of size n_d.

$$\boldsymbol{A}^d = \Phi_{A^d}(\widetilde{\mathbf{s}}, \theta_{A^d}) \tag{A.98}$$

3. For each branch d, the state value V is added to the advantage $\boldsymbol{A}^d$ producing the Q-values $\boldsymbol{Q}^d = \boldsymbol{A}^d + V$ of the different actions of branch d. Let us note $[a^{d,1}, \ldots, a^{d,n_d}]$ the different actions of branch d. Therefore, $\boldsymbol{Q}^d$ can be written as: $\boldsymbol{Q}^d = [Q^d(\mathbf{s}, a^{d,1}), \ldots, Q^d(\mathbf{s}, a^{d,n_d})]$.

It is important to note that the values associated with each branch do not strictly correspond to the Q-values defined in Section A.2.7, as they represent partial actions. However, they resemble the true Q-values of the complete action due to the approximation provided by the neural network.

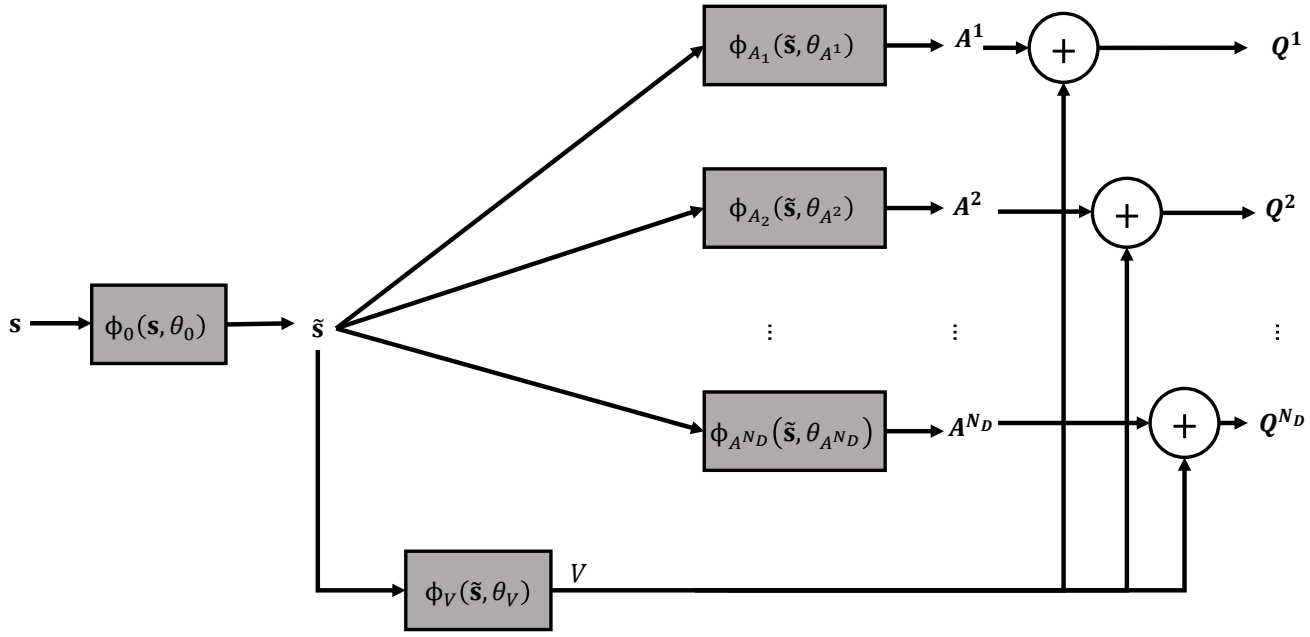


Figure A.8: Action branching (gray boxes contain trainable weights).

Algorithm 10: Action Branching

`/* Action branching forward pass */`

Input: $\boldsymbol{s} \in \mathbb{S}$.

Output: $[\boldsymbol{Q}^1, \ldots, \boldsymbol{Q}^{N_D}]$ and $[a^1, \ldots, a^{N_D}]$, the Q-values of all the branches and the taken actions.

Parameters: $\theta = \{\theta_0, \theta_V, \theta_{A^1}, \ldots, \theta_{A^{N_D}}\}$

N_D number of branches

$\tilde{\boldsymbol{s}} \leftarrow \Phi(\boldsymbol{s}, \theta_0)$

$V \leftarrow \Phi_V(\tilde{\boldsymbol{s}}, \theta_V)$

for $d = 1$ *to* N_D **do**

  $\boldsymbol{Q}^d \leftarrow \Phi_{A^d}(\tilde{\boldsymbol{s}}, \theta_{A^d}) + V$

  $a^d \leftarrow \arg\max_{a \in \mathbb{A}_d} \boldsymbol{Q}^d$

end

return $[\boldsymbol{Q}^1, \ldots, \boldsymbol{Q}^{N_D}]$, $[a^1, \ldots, a^{N_D}]$

Since all the parameters are trained jointly, let us note $\theta = \{\theta_0, \theta_V, \theta_{A^1}, \dots, \theta_{A^{N_D}}\}$ and the Q-values obtained on branch d, for state $\mathbf{s}$ and action $a \in \mathbb{A}_d$, $Q^d(\mathbf{s}, a, \theta)$.

In [77], the AB architecture is trained with the double DQL presented in Section A.3. Hence, the target value of the dth branch is written as:

$$y_d = r + \gamma Q^d(\mathbf{s}_{k+1}, \arg\max_{a \in \mathbb{A}_d} Q^d(\mathbf{s}_{k+1}, a, \theta), \theta^-). \tag{A.99}$$

After computing the target value of all the branches, the loss is computed as:

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{s}, \boldsymbol{a}, r, \mathbf{s}') \sim \mathcal{B}} \left[\frac{1}{N_D} \sum_{d=1}^{N_D} \left(y_d - Q^d(\mathbf{s}, a^d, \theta) \right)^2 \right], \tag{A.100}$$

which corresponds to an aggregation of the TD of all the branches. This loss is used to train jointly the set of parameters θ with (A.76).

The action taken on each branch d is:

$$a^d := \arg\max_{a \in \mathbb{A}_d} Q^d(\mathbf{s}, a, \theta), \tag{A.101}$$

and the total action is:

$$\boldsymbol{a} := [a^1, \dots, a^{N_D}]. \tag{A.102}$$

A.5 Conclusion

This Appendix provided an overview of the core principles of ML that are employed throughout this thesis. Specifically, it introduced the MDP, which serves as the mathematical framework for formulating the scheduling and resource allocation problem. Additionally, the chapter covered DQL, the learning algorithm utilized in this thesis for addressing these MDPs. Finally, it discussed the different architectures used to approximate the Q-function, with a particular focus on the EOT and the AB.

Appendix B

Number of possible states for APD

Let us consider one buffer with DC. We assume that we know the value of all the entries of this buffer. Consequently, we also know the number of entries equal to -1, the number of entries equal to 0, and so on, up to the number of entries equal to D. Since there are $k = D + 2$ distinct possible values, the problem reduces to determining the number of ways to express the total buffer size B as a sum of k non-negative integers. We claim that this number is given by the binomial coefficient:

$$\binom{B + D + 1}{D + 1}. \tag{B.1}$$

We establish this result using induction.

Recursive formulation. Define $F_k(B)$ as the number of ways to represent B using k non-negative integers. Fixing one of the k integers to take a value ℓ, the remaining $k - 1$ integers must then sum to $B - \ell$. Summing over all possible values of ℓ, we obtain:

$$F_k(B) = \sum_{\ell=0}^{B} F_{k-1}(\ell). \tag{B.2}$$

Base case. For $k = 1$, we are left with a single integer that must sum to B. Clearly, there is exactly one such representation:

$$F_1(B) = 1 = \binom{B}{0}.$$

Thus, the base case holds.

Inductive hypothesis. Assume that for some $k \geq 1$, the following closed form holds:

$$F_k(B) = \binom{B + k - 1}{k - 1}.$$

Inductive step. We now prove the formula for $k+1$. Using the recursive definition (B.2), we have:

$$\begin{aligned} F_{k+1}(B) &= \sum_{\ell=0}^{B} F_k(\ell) \\ &= \sum_{\ell=0}^{B} \binom{\ell+k-1}{k-1}. \end{aligned} \tag{B.3}$$

The combinatorial identity in [113, Section 0.15] states that:

$$\sum_{\ell=0}^{B} \binom{\ell+k}{k} = \binom{B+k+1}{k+1} \tag{B.4}$$

Substituting (B.4) into (B.3), we obtain

$$F_{k+1}(B) = \binom{B+k}{k}.$$

Conclusion. By the principle of mathematical induction, it follows that

$$F_k(B) = \binom{B+k-1}{k-1},$$

for all $k \geq 1$ and $B \geq 0$. In particular, when $k = D+2$, the number of ways to represent B as a sum of $D+2$ non-negative integers is

$$\binom{B+D+1}{D+1}.$$

This completes the proof. □

Appendix C

Figures for Chapter 2

The following figures provide the distribution of packet delays across the different buffers for the evaluated slot-based schedulers in Chapter2. They also provide the PLR associated with the different buffers. The inference was performed with $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$. These distributions allow a fine analysis of the behavior of the different schedulers under these conditions. The discussion of these figures is provided in Section 2.6.2.3.

C.1 Figure for DC traffic

C.1.1 Heuristics

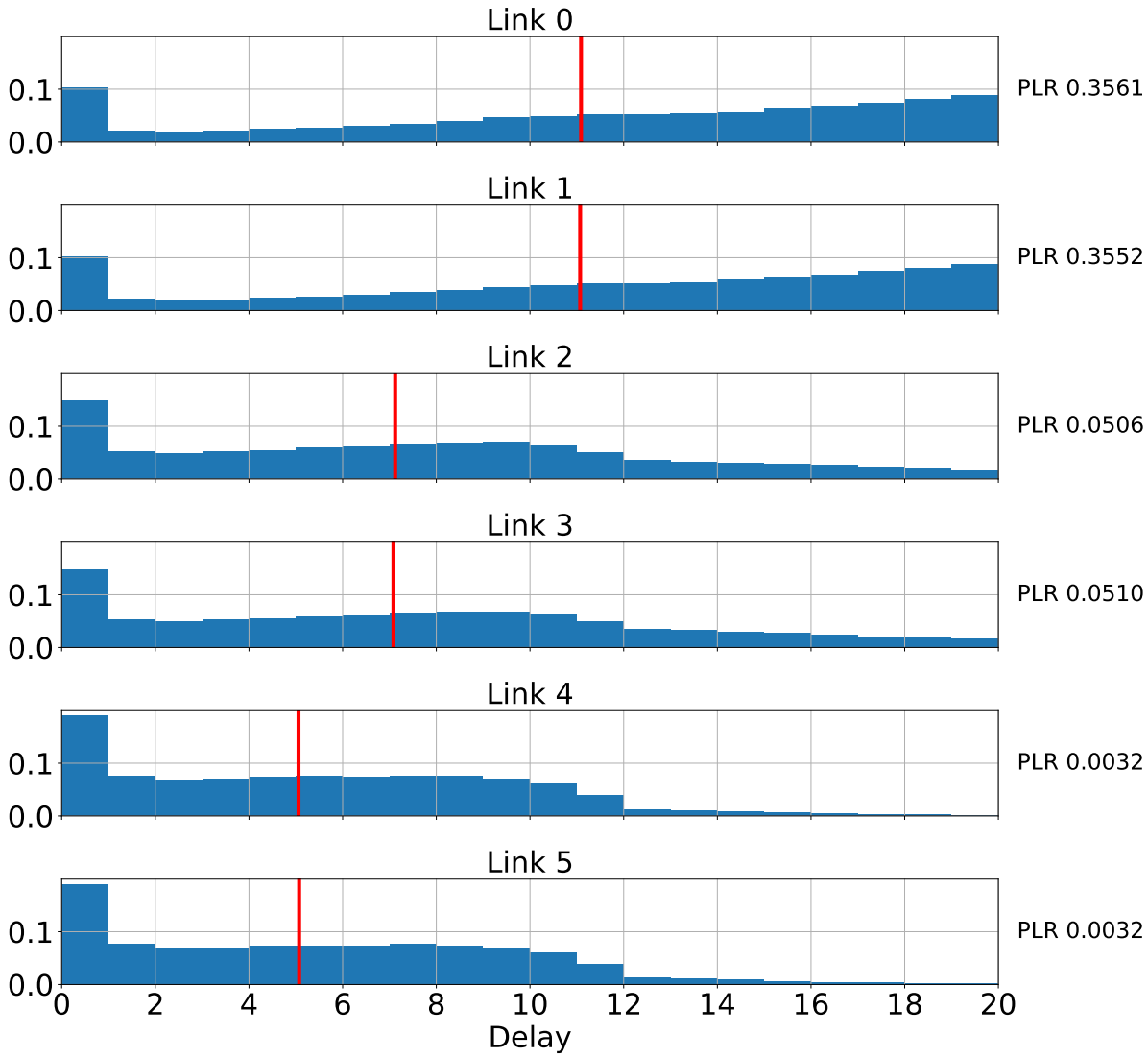


Figure C.1: Distribution of the packet delay for DC traffic using RR, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$

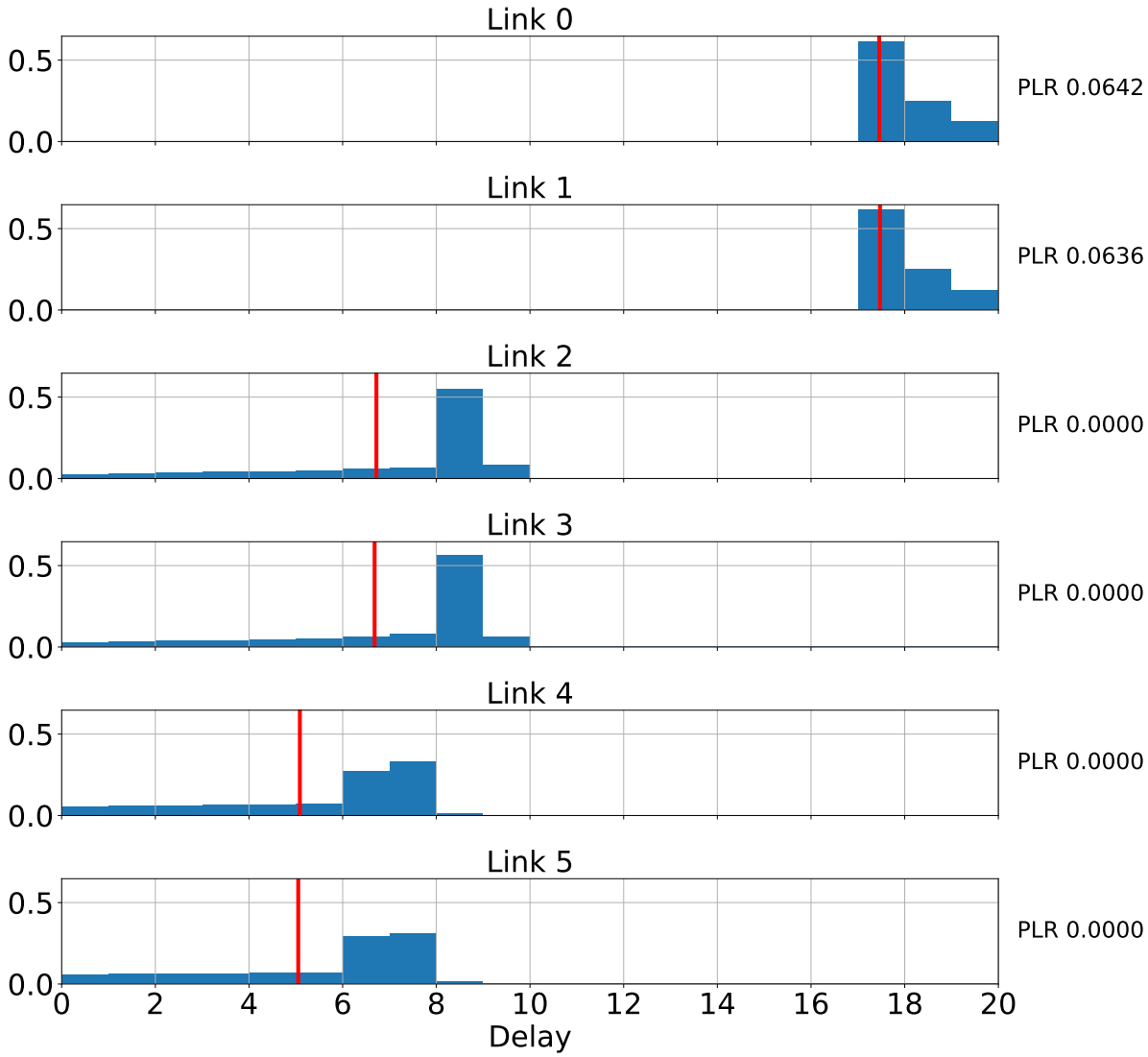


Figure C.2: Distribution of the packet delay for DC traffic using LOG-rule, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

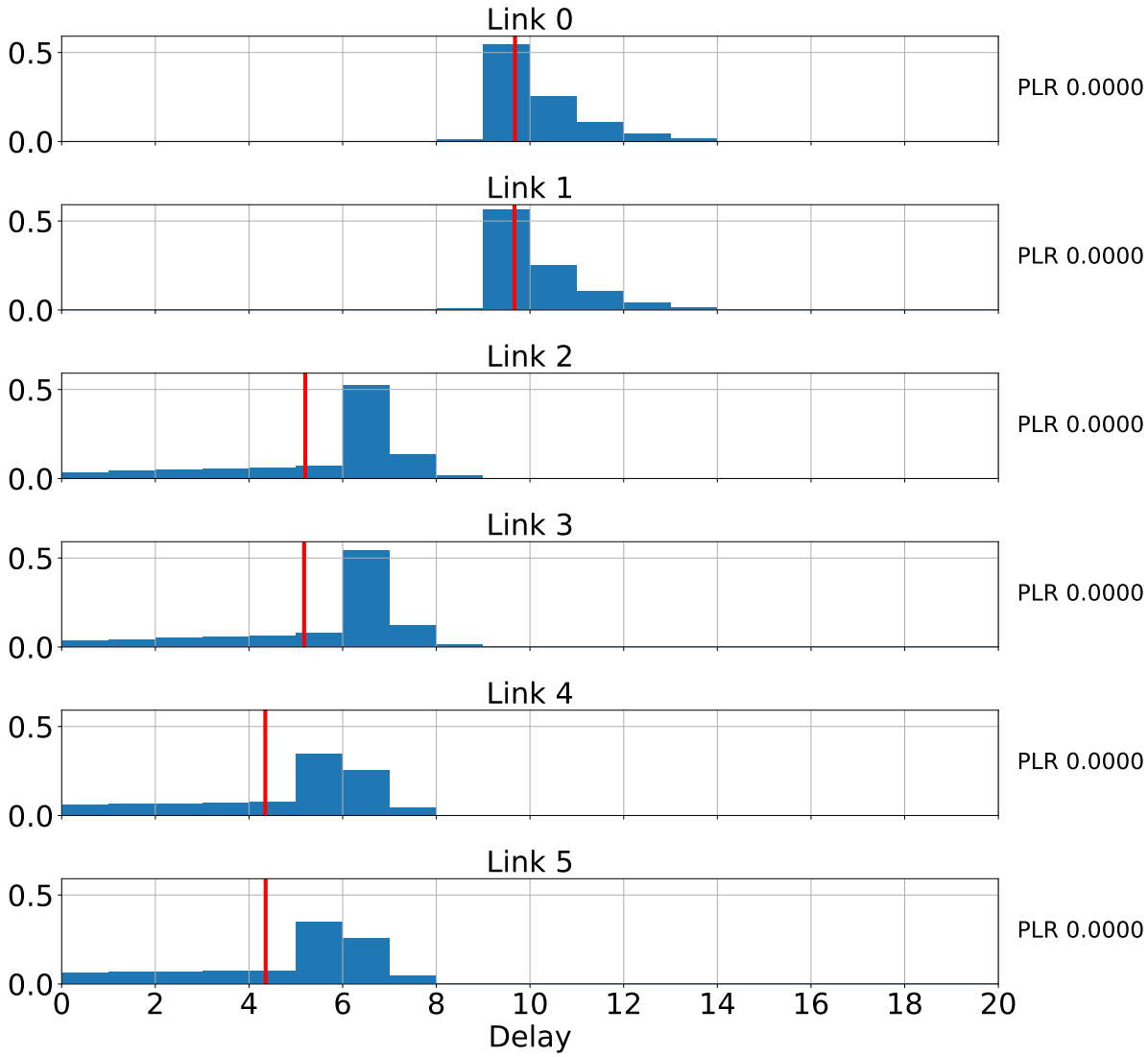


Figure C.3: Distribution of the packet delay for DC traffic using MLWDF, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

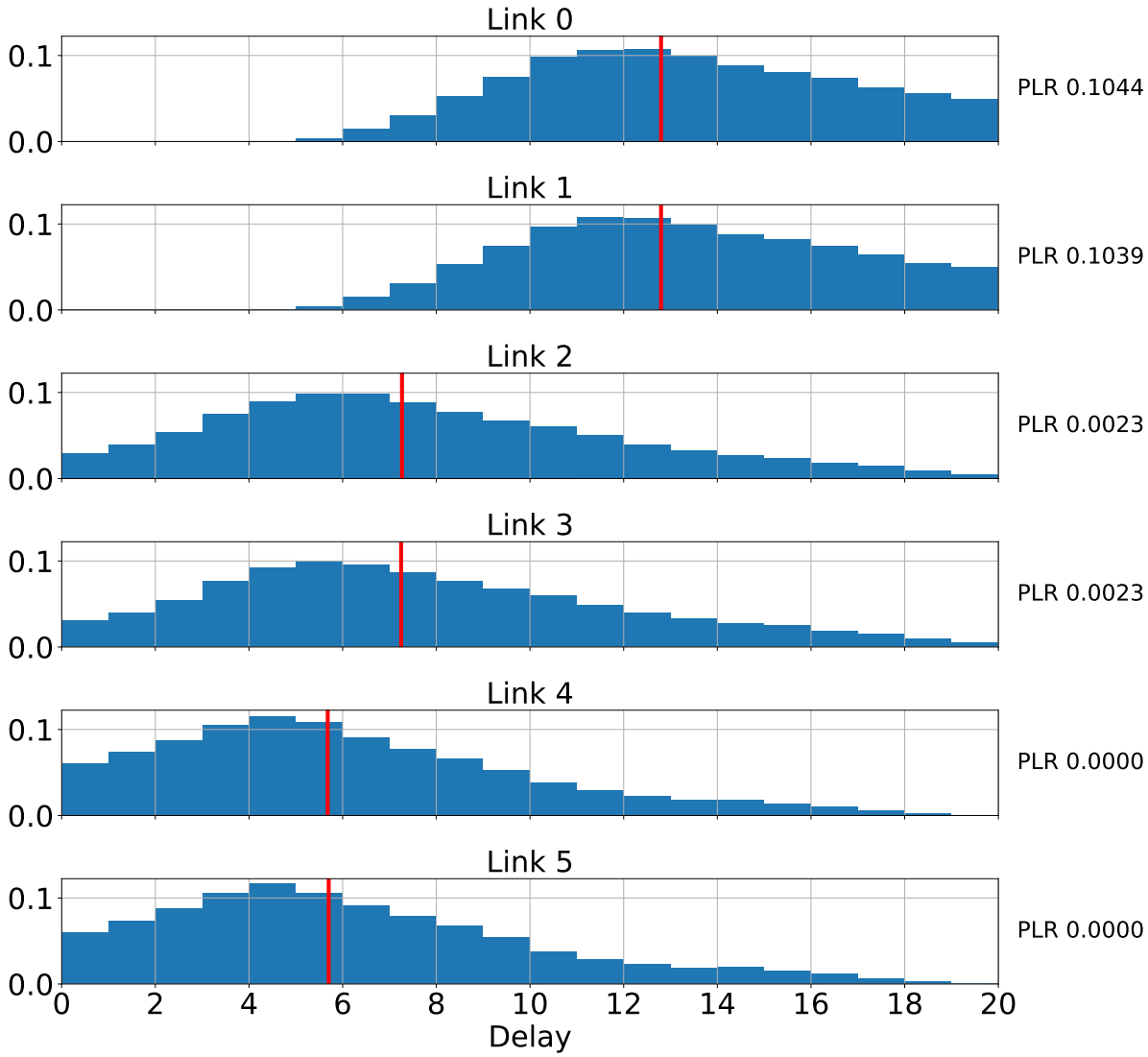


Figure C.4: Distribution of the packet delay for DC traffic using EXP-rule, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

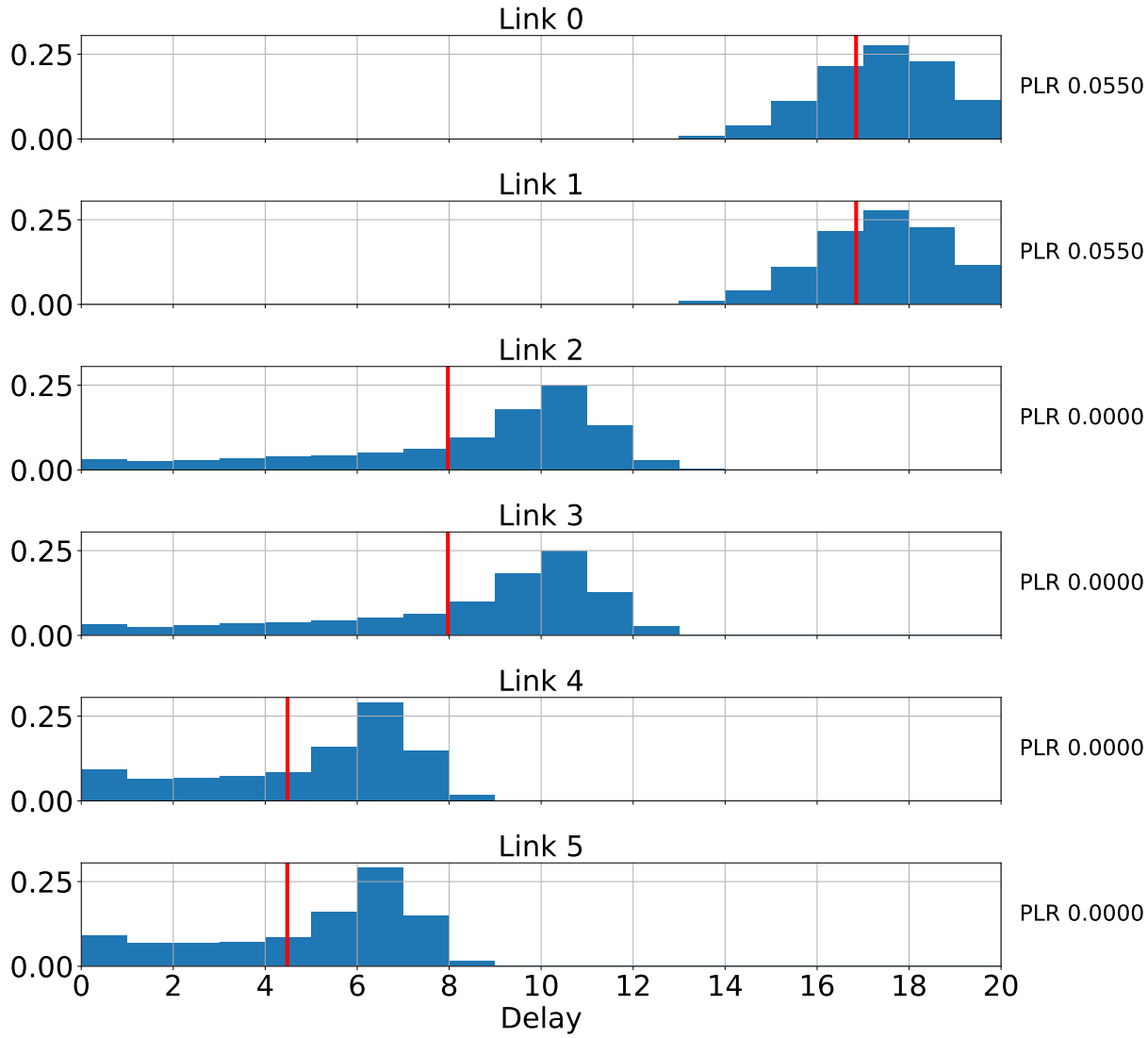


Figure C.5: Distribution of the packet delay for DC traffic using KP, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

C.1.2 Fully connected

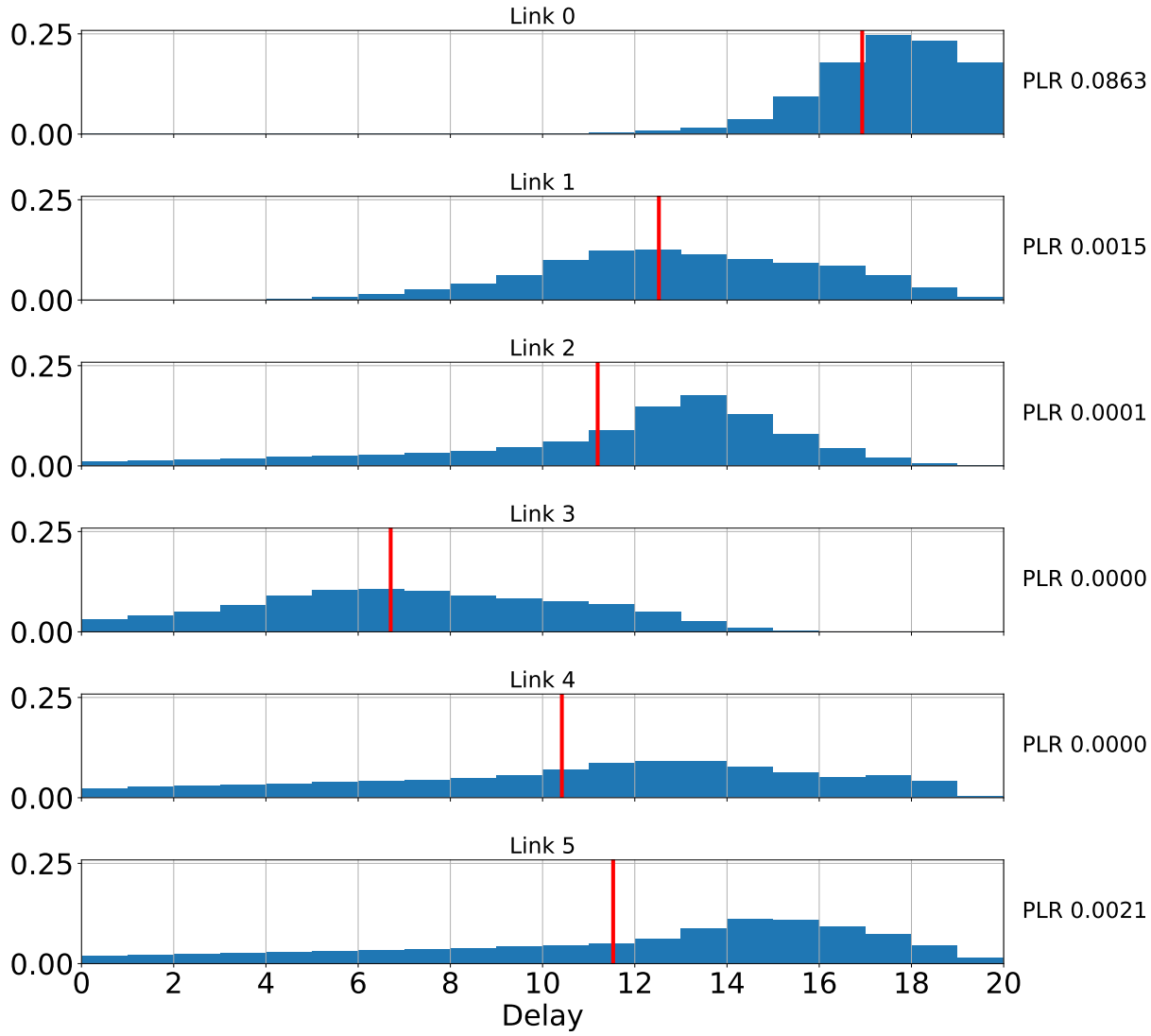


Figure C.6: Distribution of the packet delay for DC traffic using FC-HoL, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

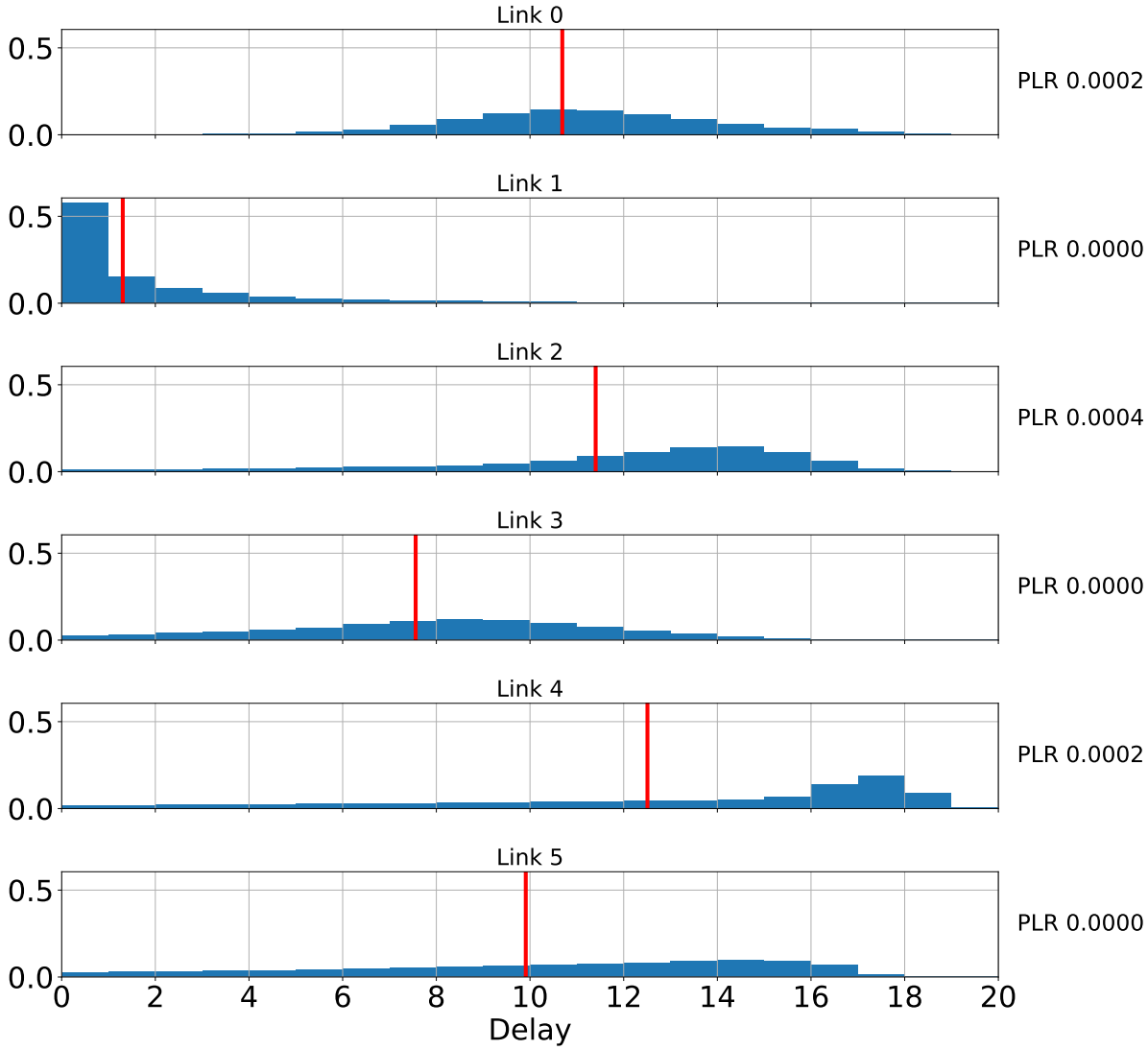


Figure C.7: Distribution of the packet delay for DC traffic using FC-xHoL, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

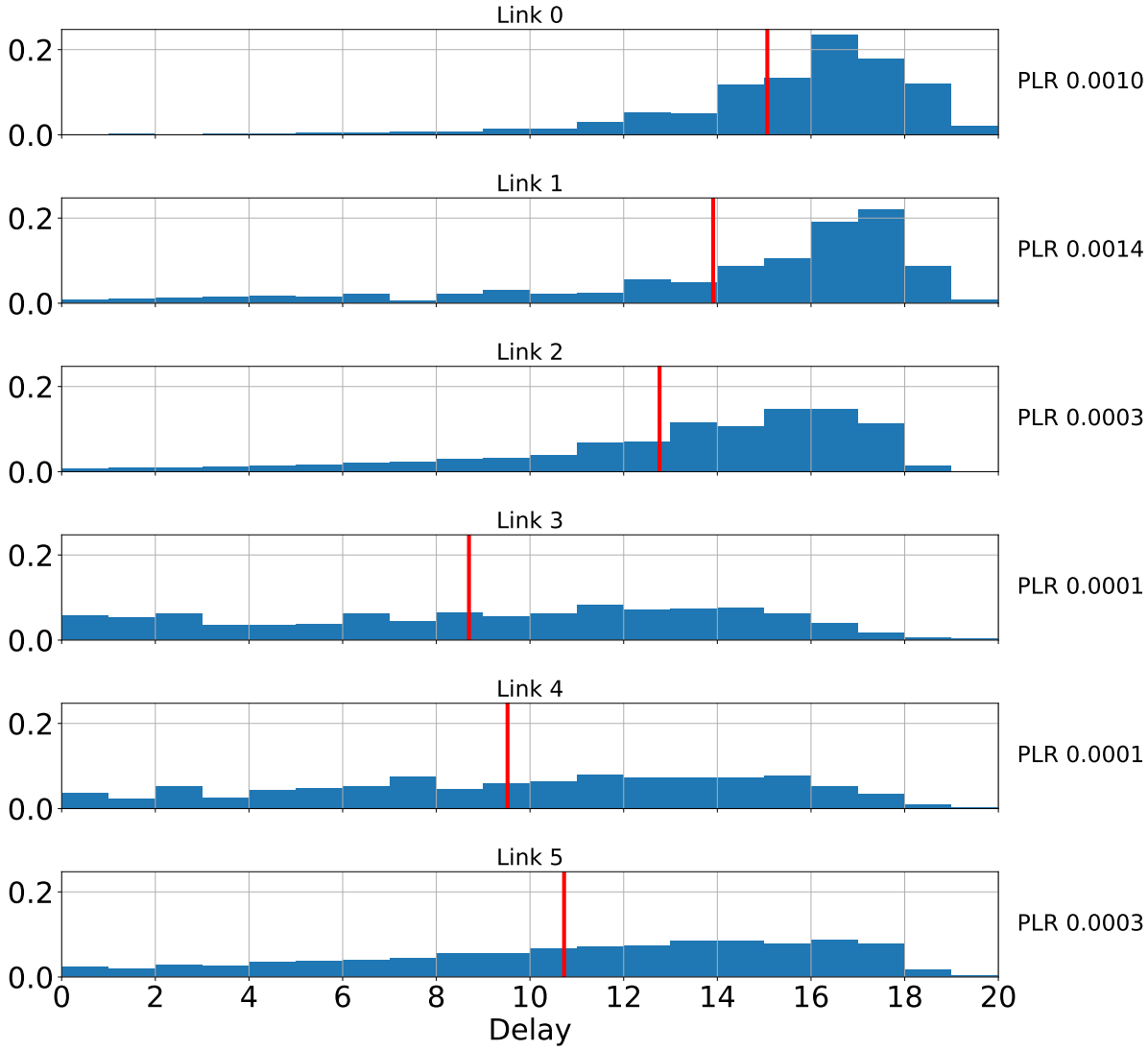


Figure C.8: Distribution of the packet delay for DC traffic using FC-APD, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

C.1.3 EOT

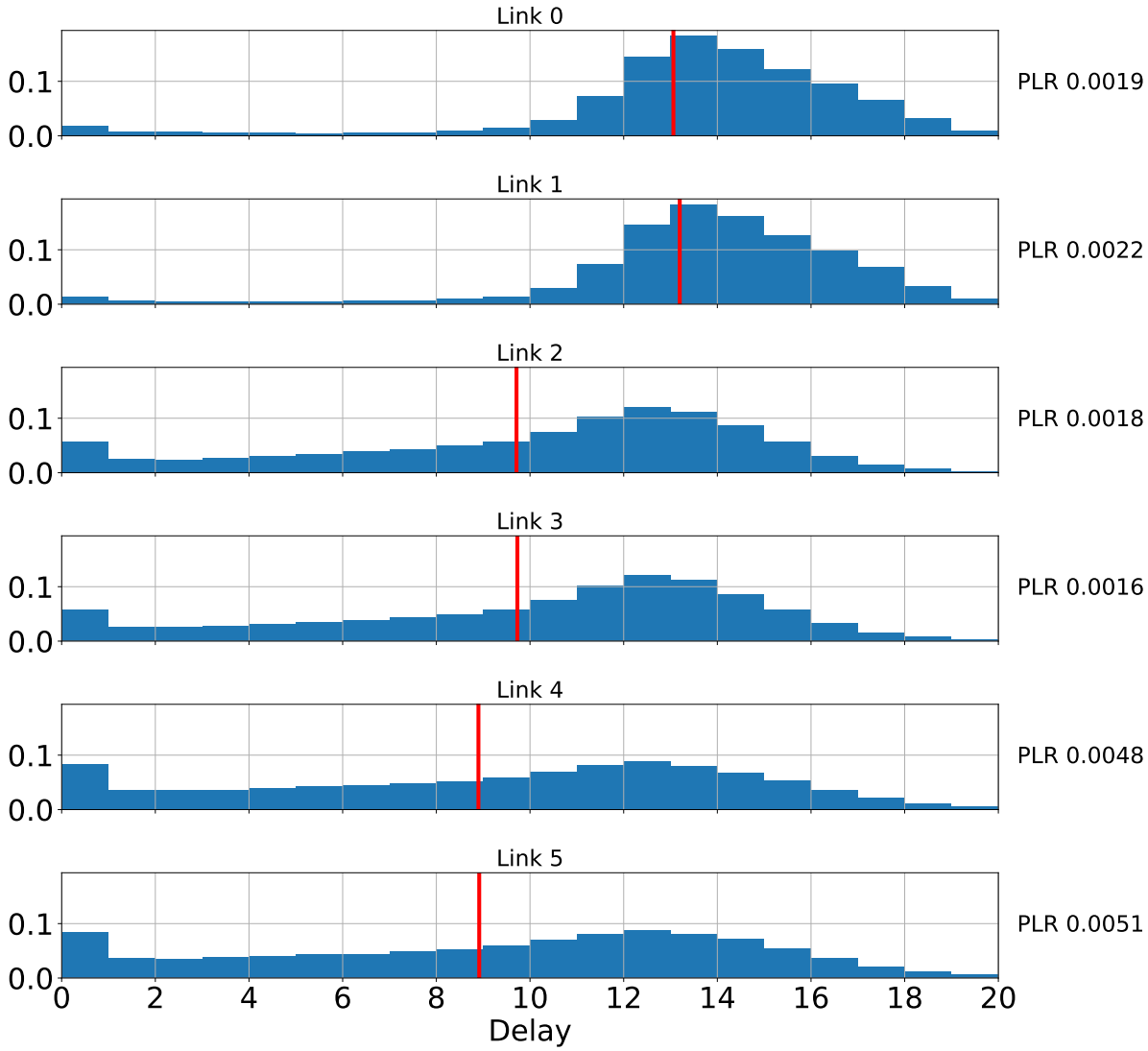


Figure C.9: Distribution of the packet delay for DC traffic using EOT-HoL, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

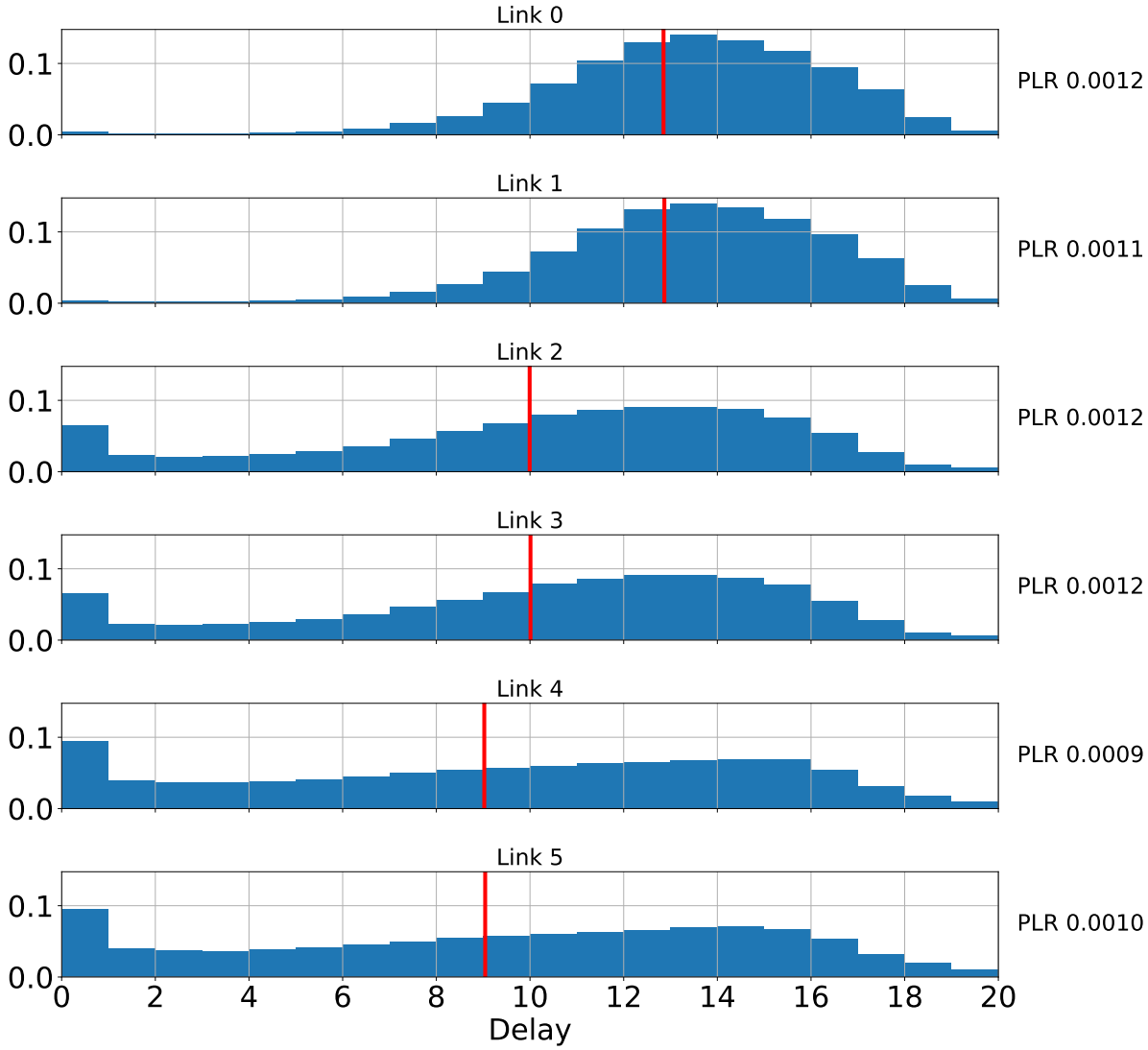


Figure C.10: Distribution of the packet delay for DC traffic using EOT-xHoL, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

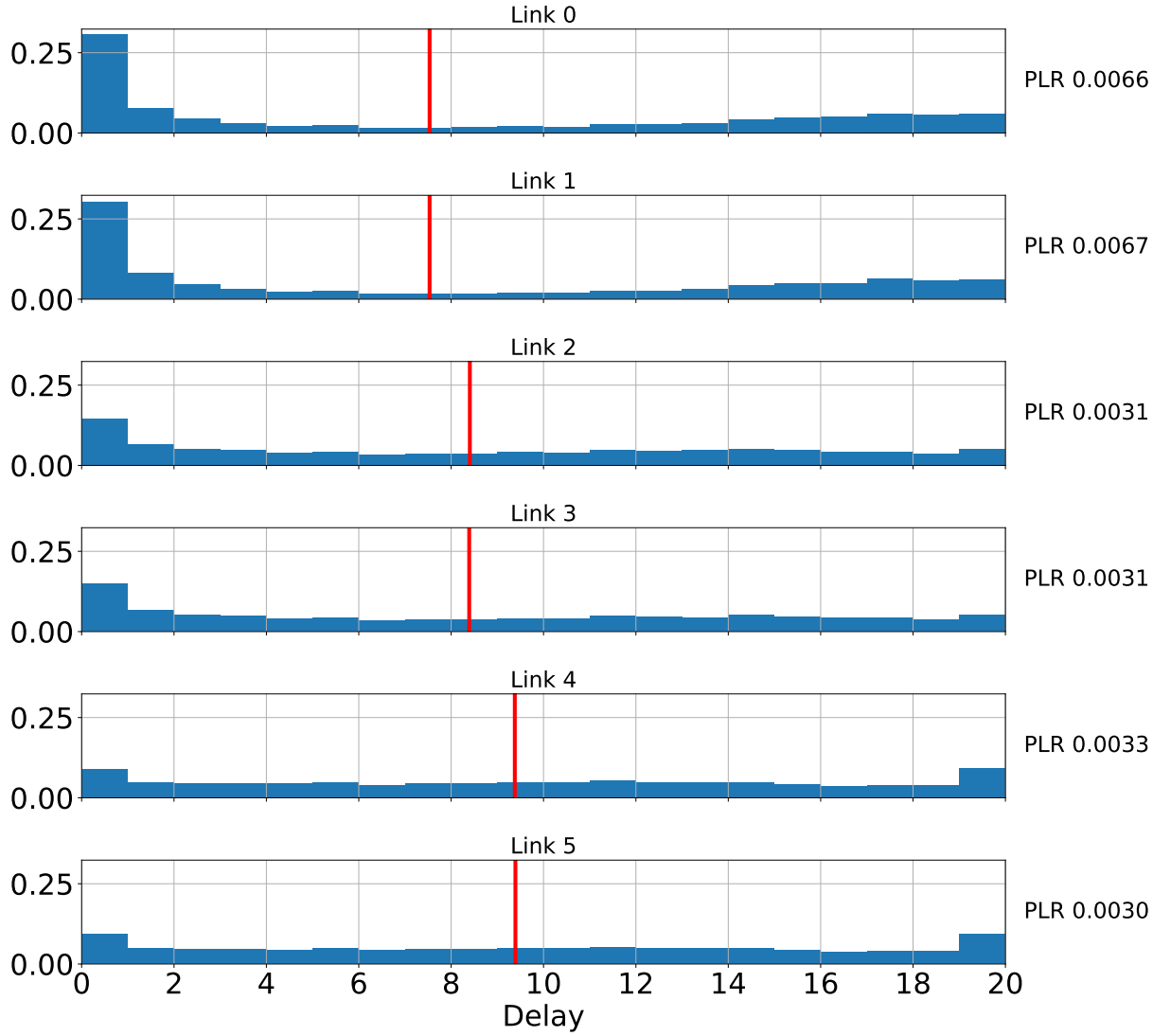


Figure C.11: Distribution of the packet delay for DC traffic using EOT-APD, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

C.2 Figure for BE traffic

C.2.1 Heuristics

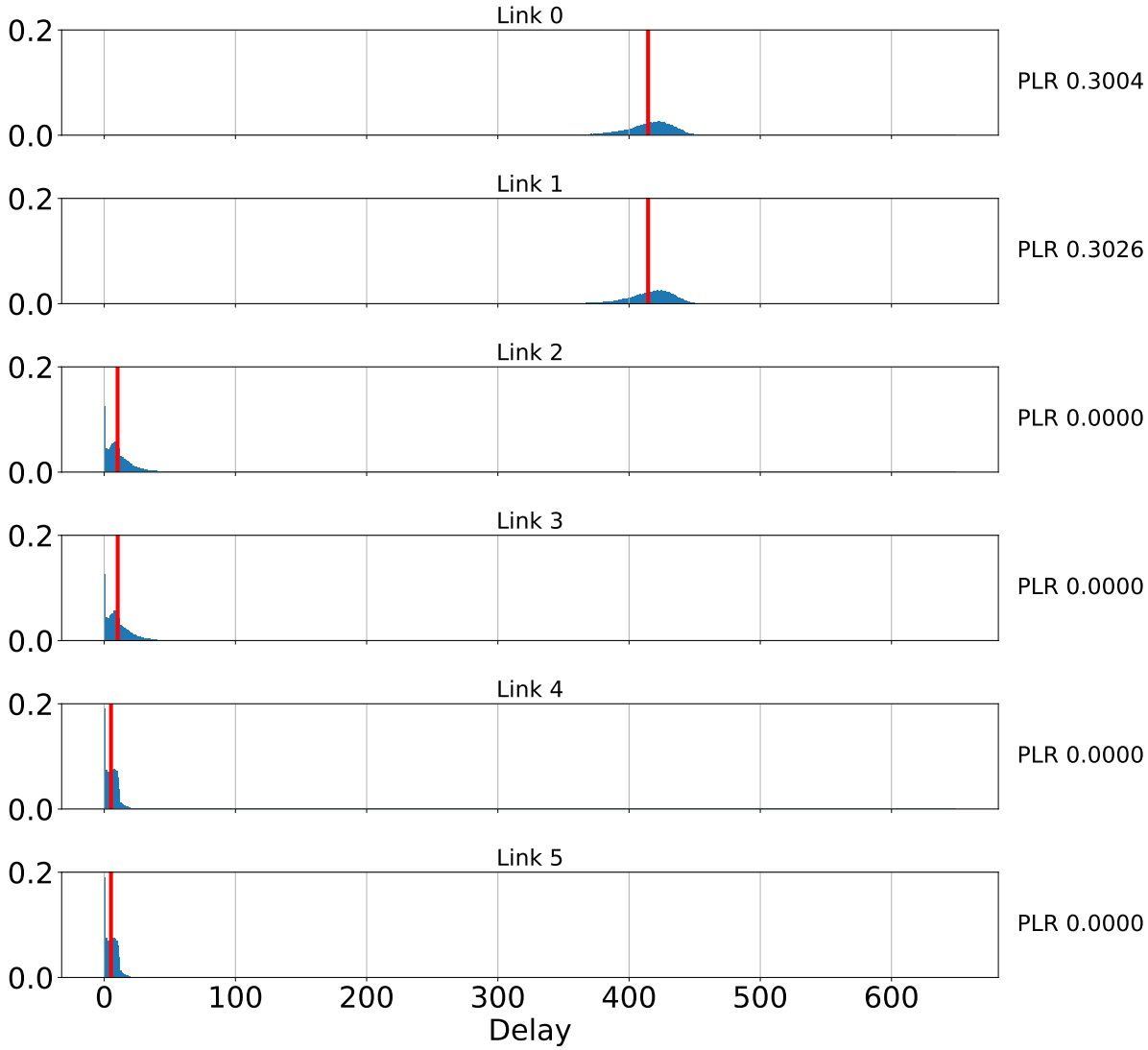


Figure C.12: Distribution of the packet delay for BE traffic using RR, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

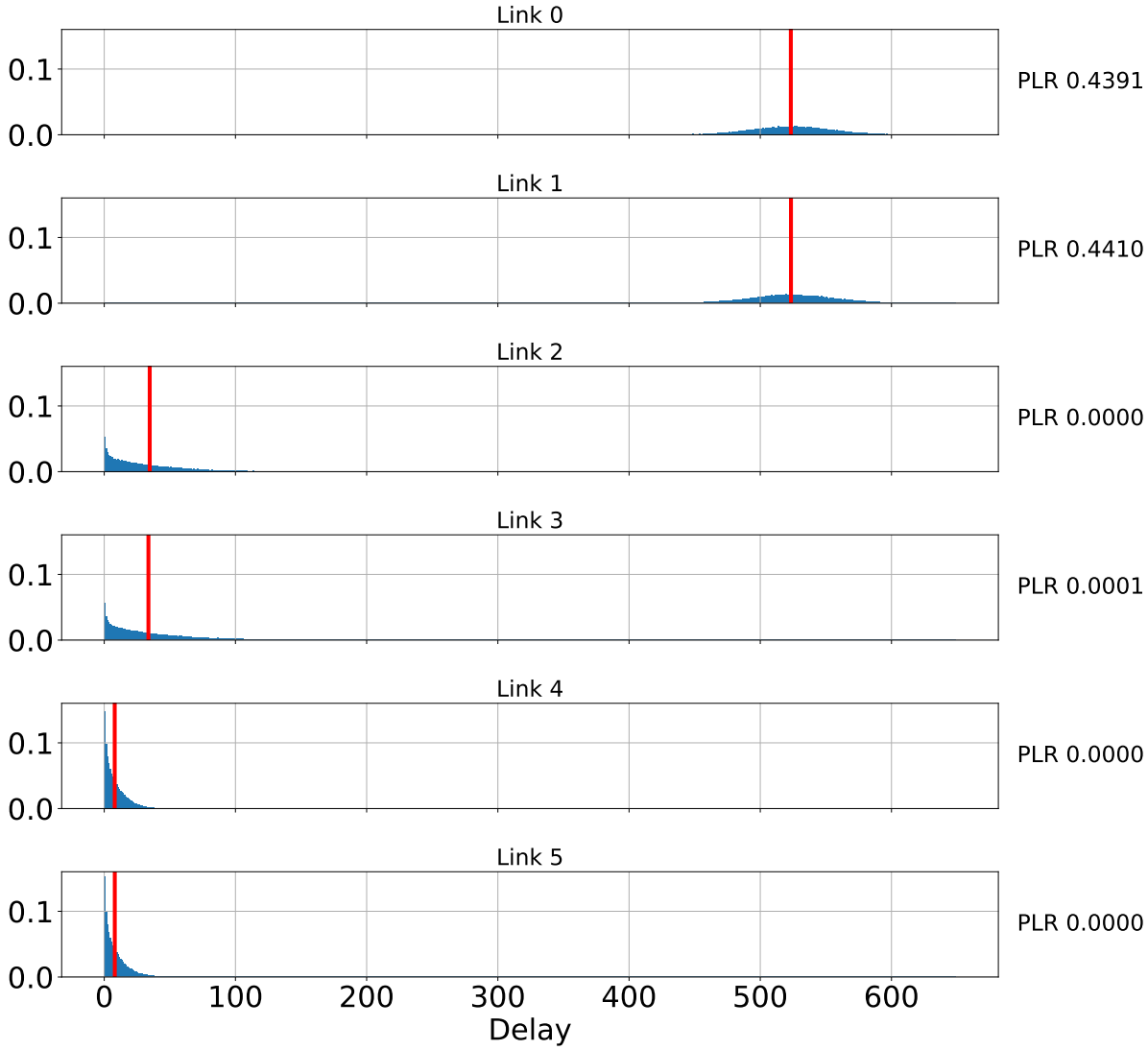


Figure C.13: Distribution of the packet delay for BE traffic using LOG-rule, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

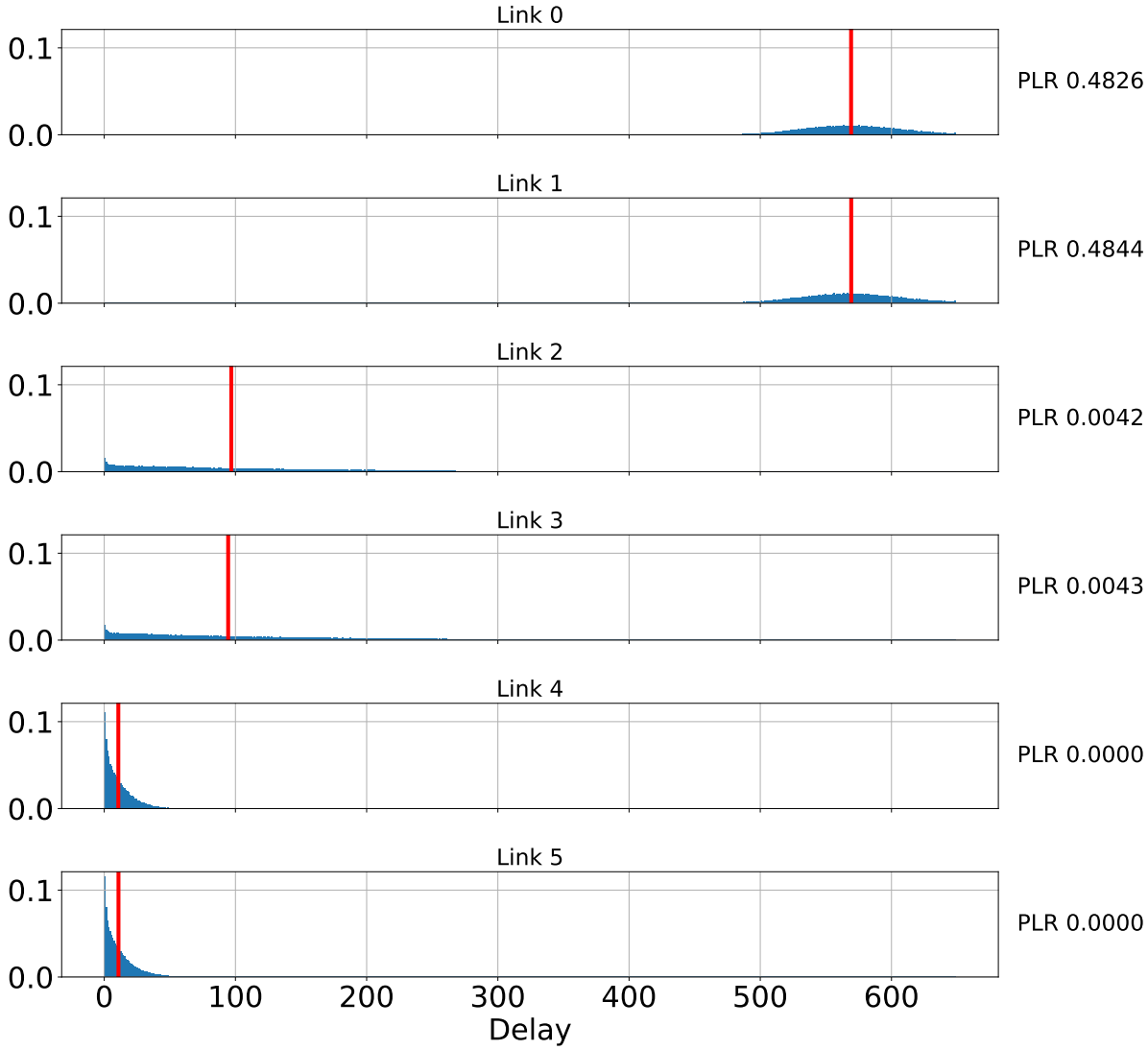


Figure C.14: Distribution of the packet delay for BE traffic using MLWDF, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

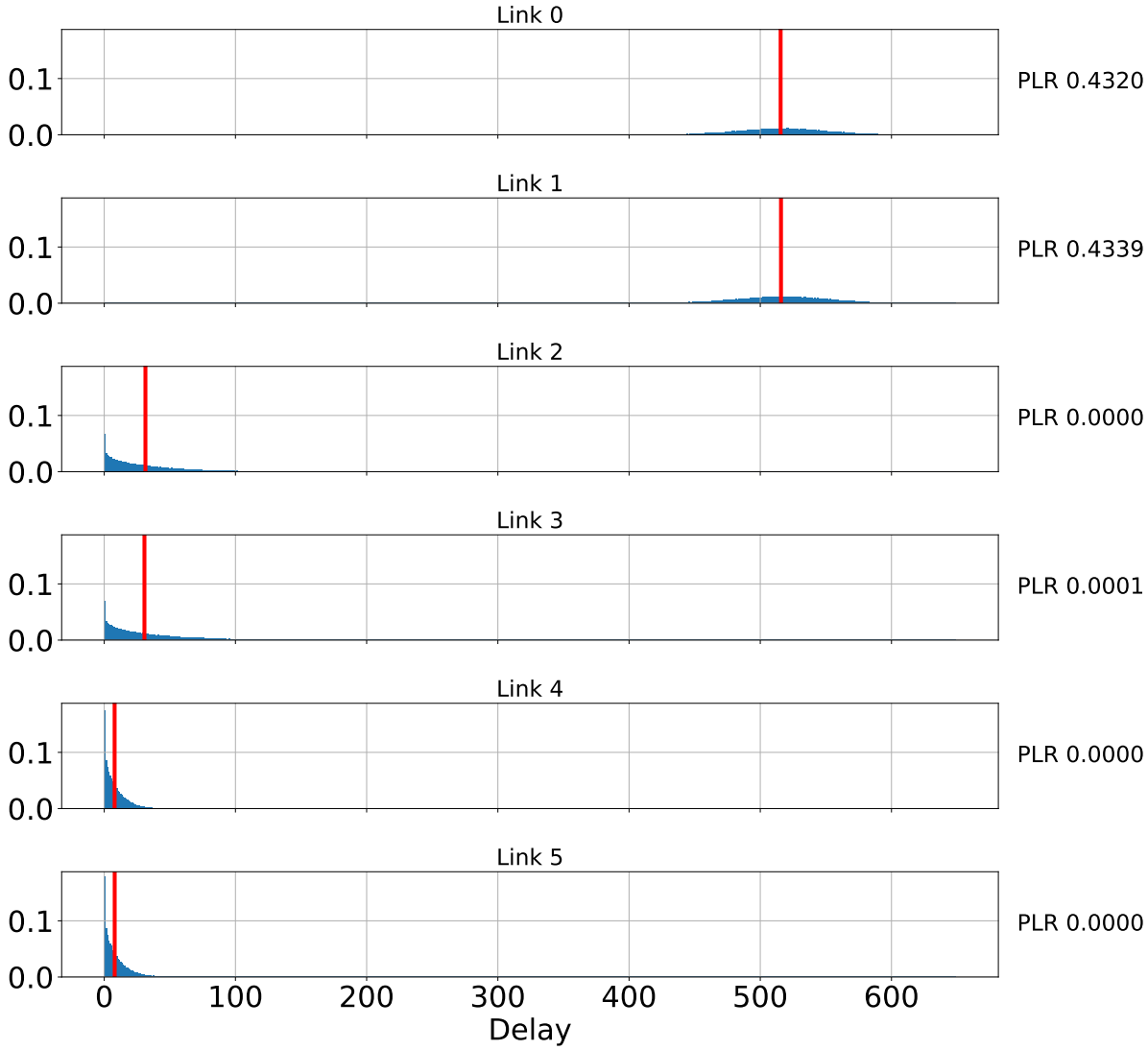


Figure C.15: Distribution of the packet delay for BE traffic using EXP-rule, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

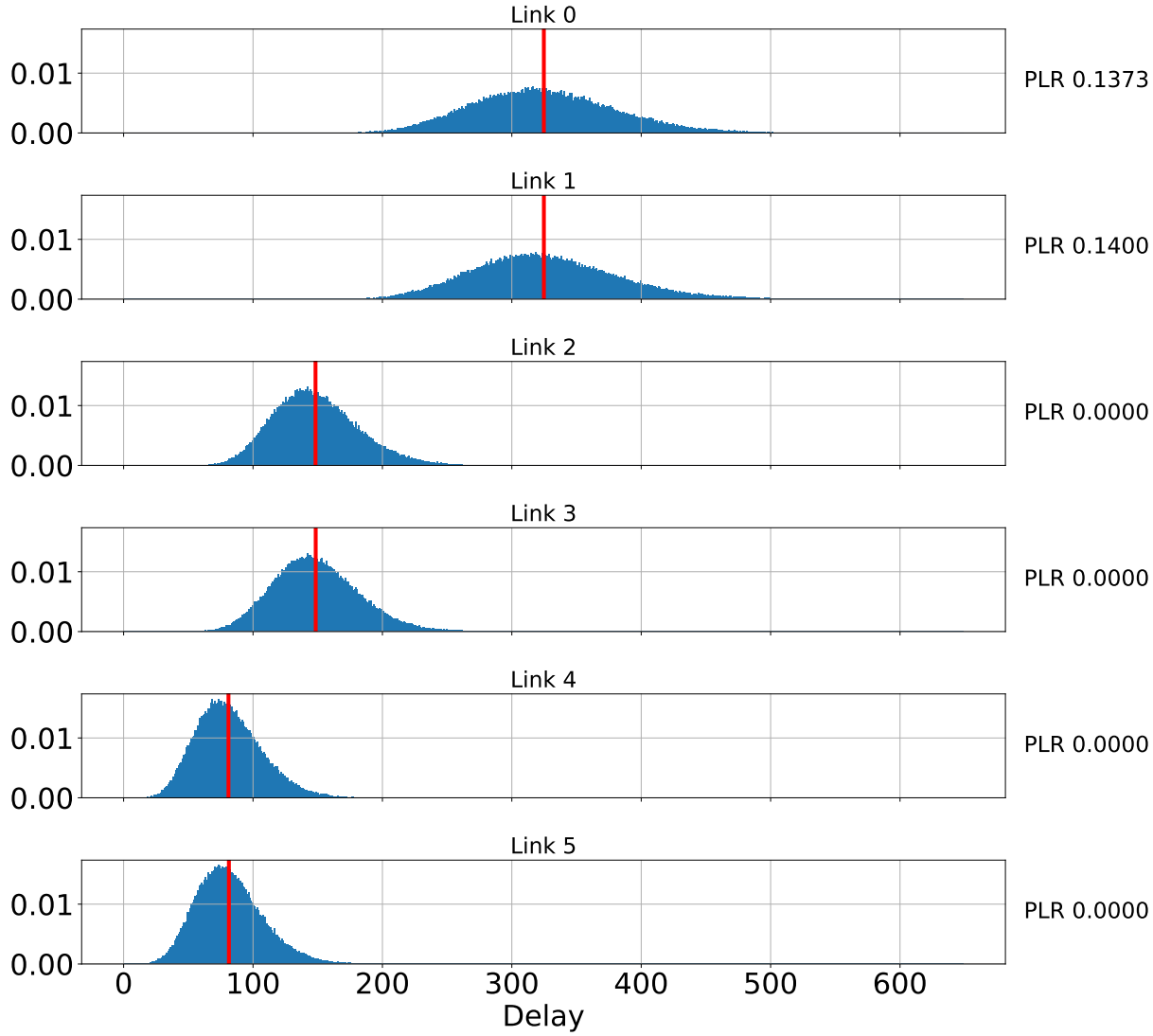


Figure C.16: Distribution of the packet delay for BE traffic using KP, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

C.2.2 Fully connected

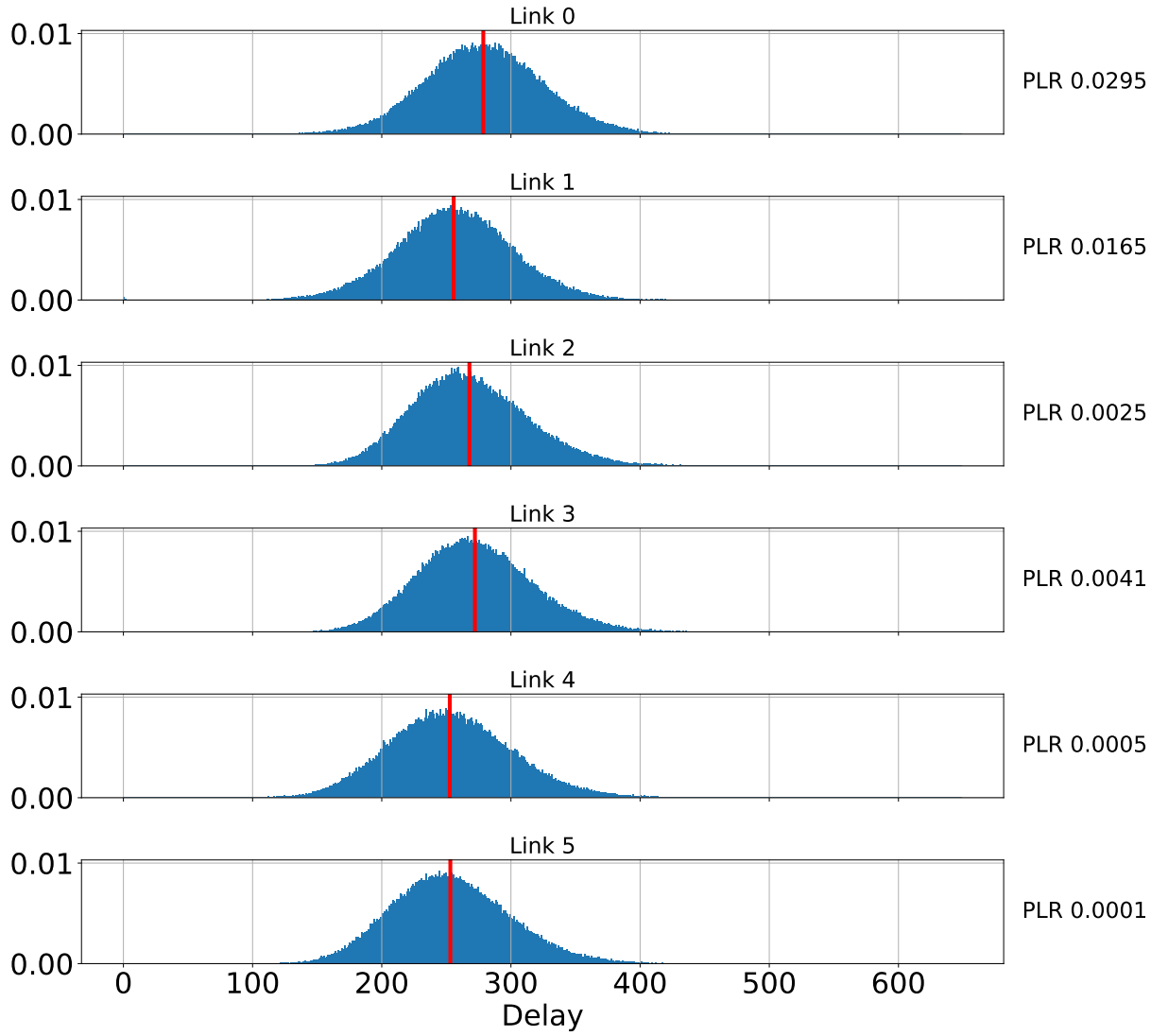


Figure C.17: Distribution of the packet delay for BE traffic using FC-HoL, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

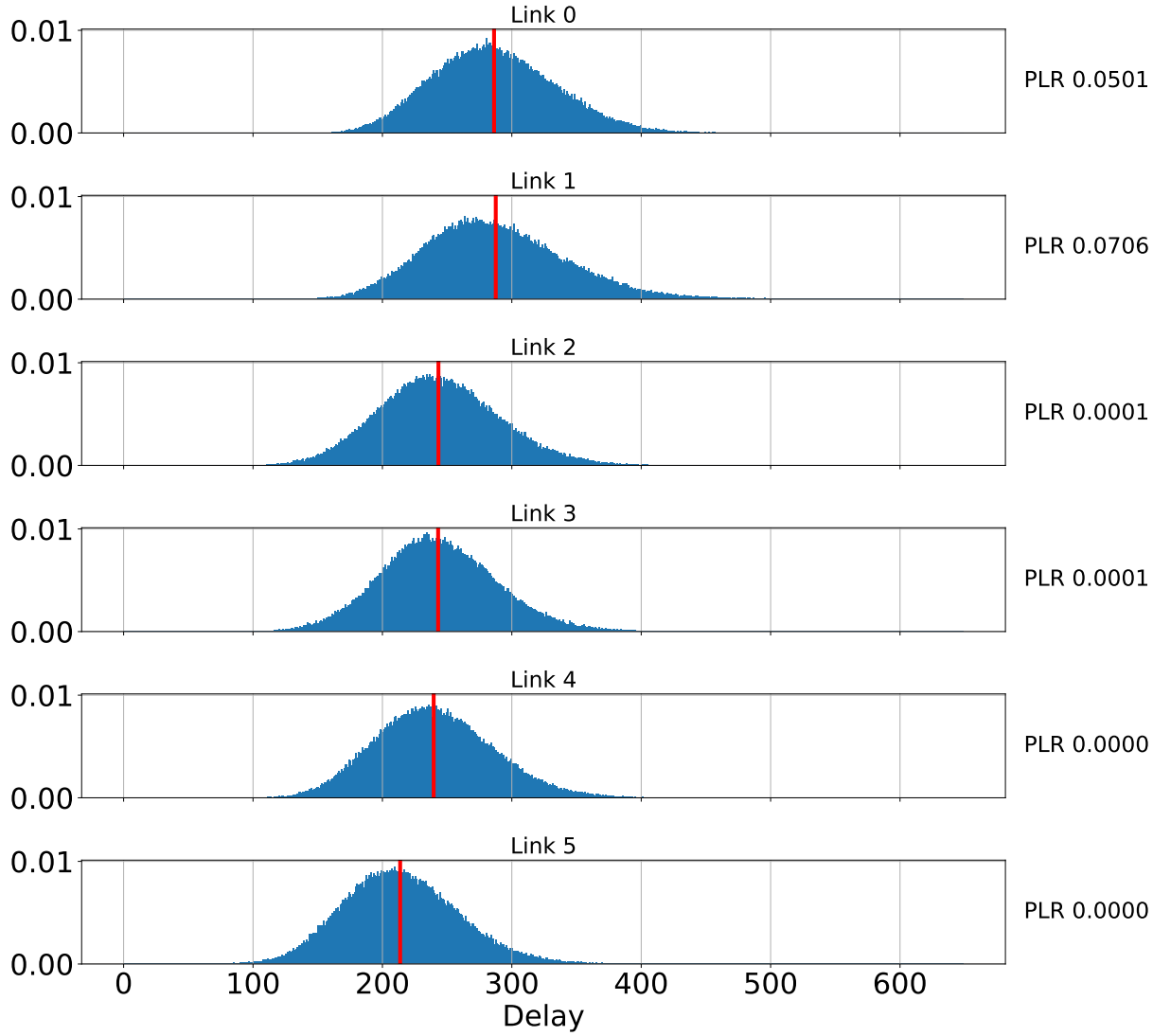


Figure C.18: Distribution of the packet delay for BE traffic using FC-xHoL, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

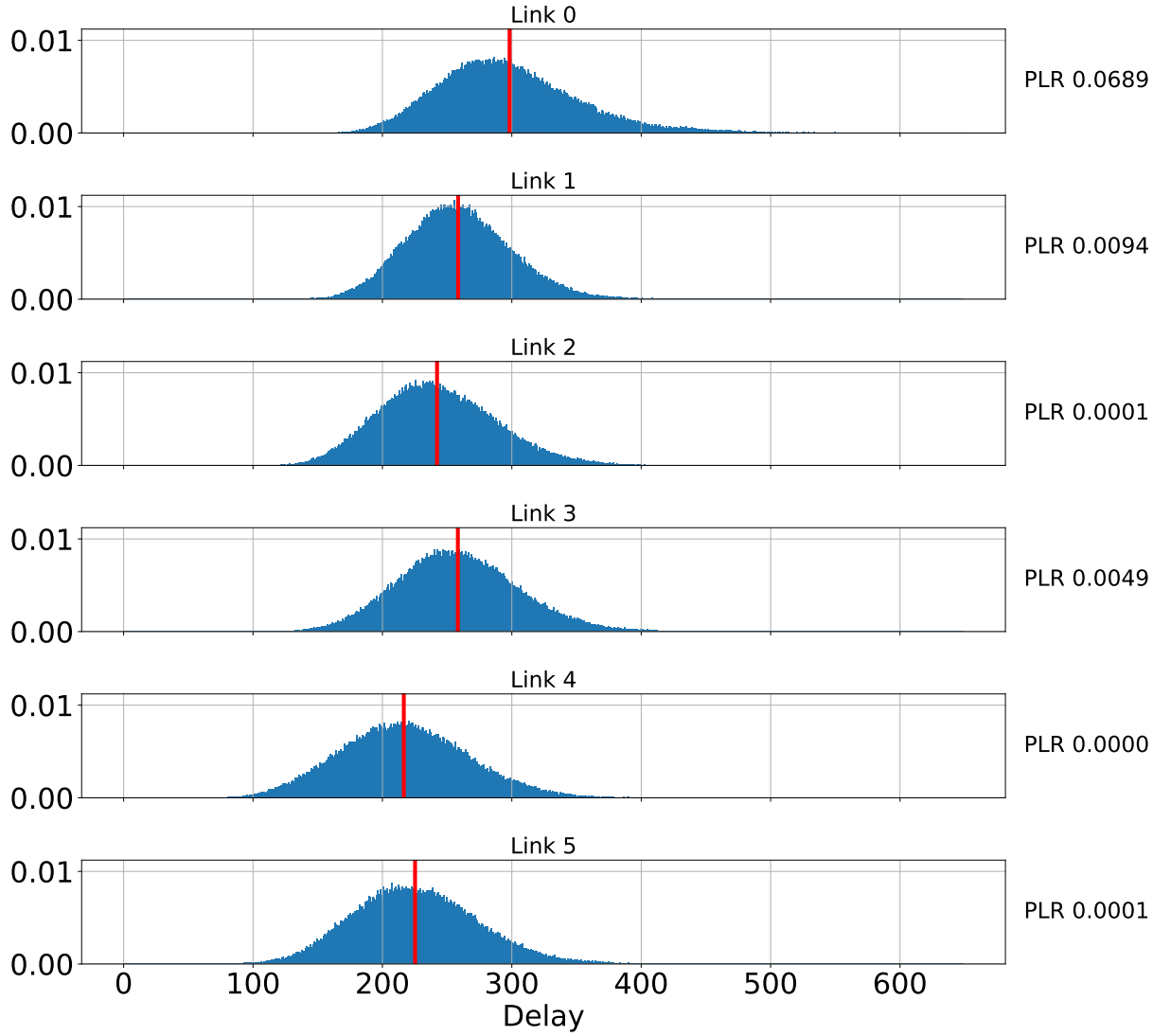


Figure C.19: Distribution of the packet delay for BE traffic using FC-APD, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

C.2.3 EOT

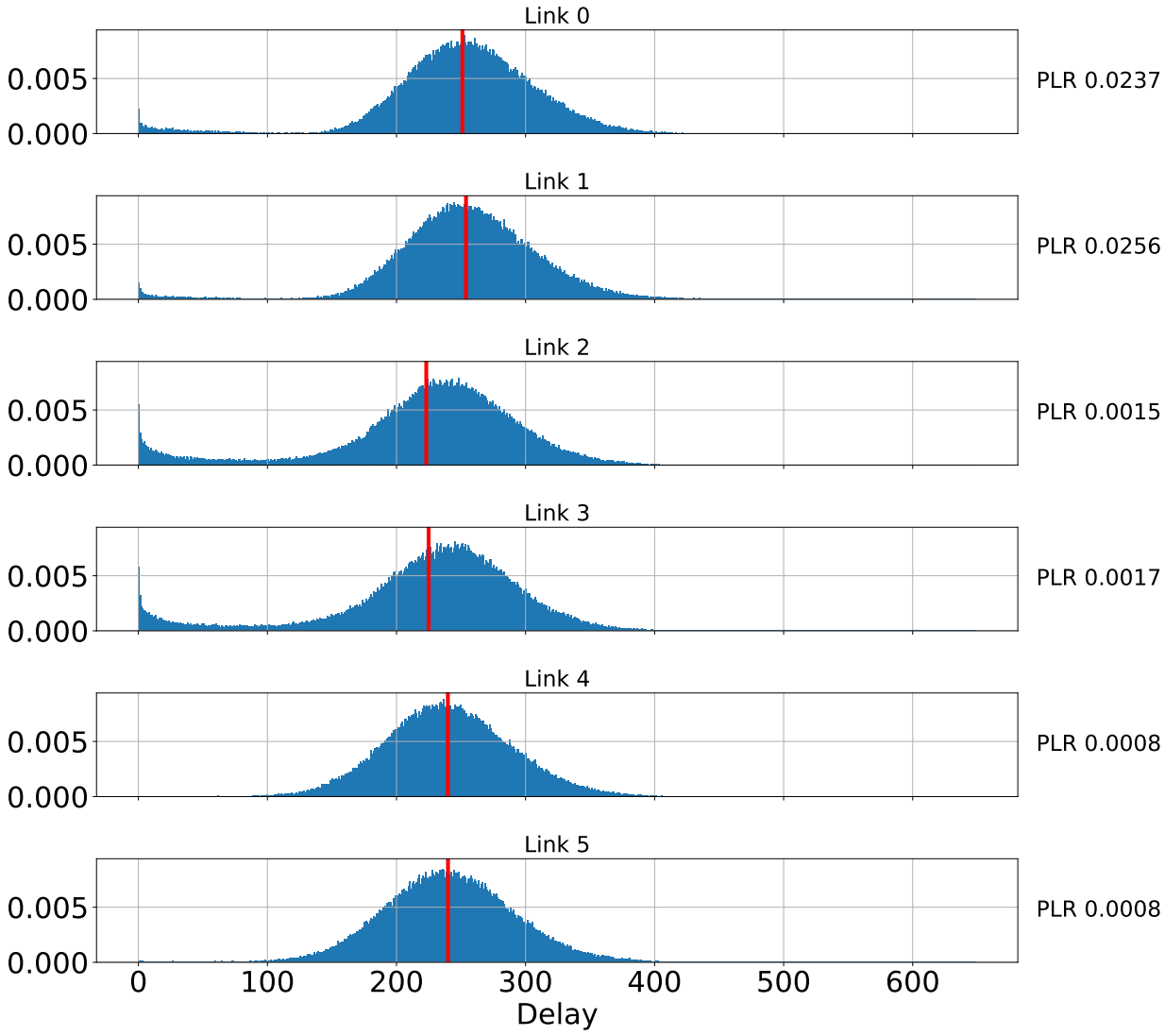


Figure C.20: Distribution of the packet delay for BE traffic using EOT-HoL, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

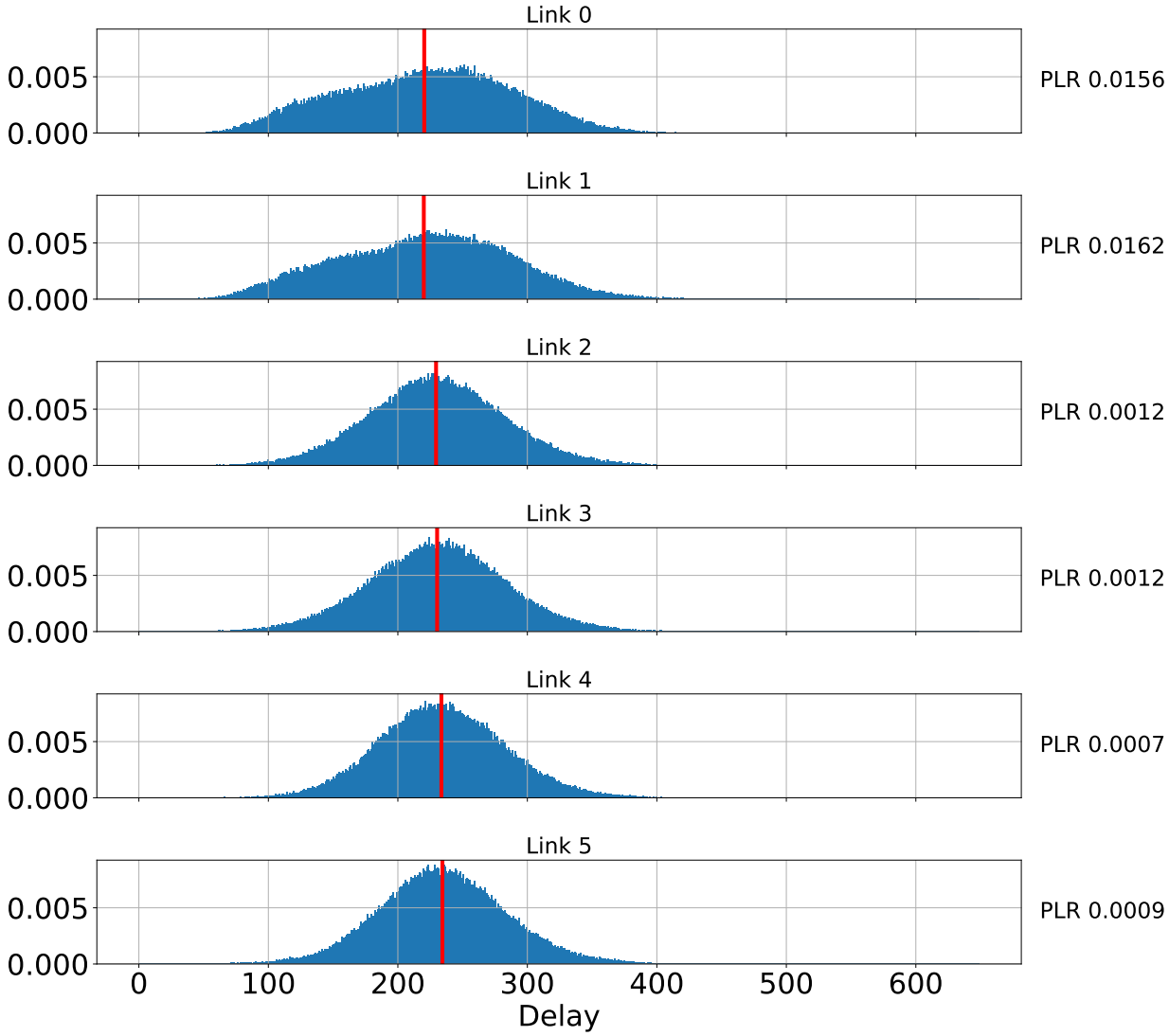


Figure C.21: Distribution of the packet delay for BE traffic using EOT-xHoL, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

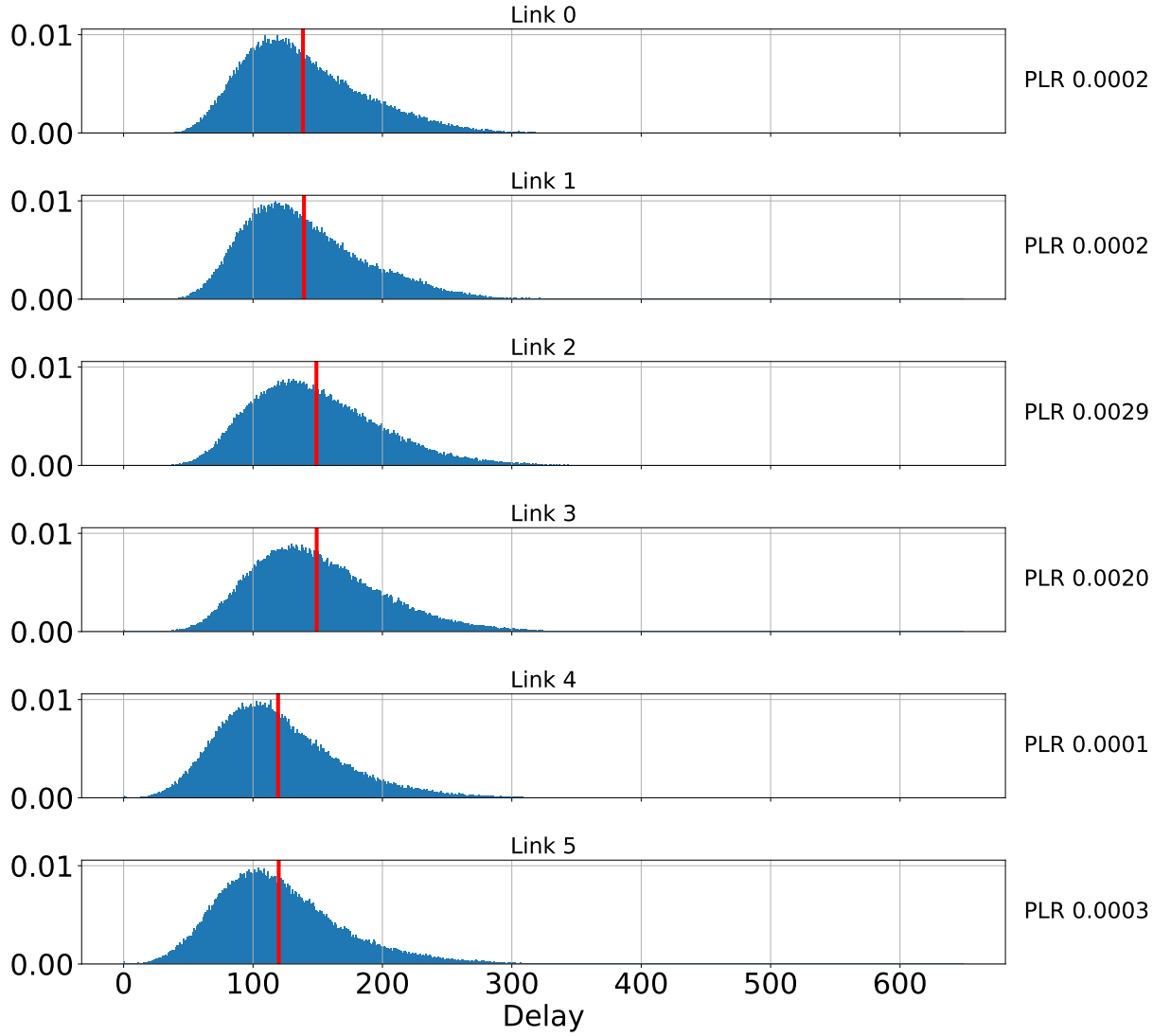


Figure C.22: Distribution of the packet delay for BE traffic using EOT-APD, for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$ and $\Lambda = 1.6$.

Bibliography

[1] X. Leturc, "Resource allocation for HARQ in mobile ad hoc networks," Ph.D. dissertation, Université Paris Saclay (COmUE), 2018.

[2] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE communications magazine*, vol. 57, no. 8, pp. 84–90, 2019.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[5] I. Fawaz, M. Sarkiss, and P. Ciblat, "Optimal resource scheduling for energy harvesting communications under strict delay constraint," in *IEEE International Conference on Communications (ICC)*, 2018.

[6] A. Valcarce. Wireless suite: A collection of problems in wireless telecommunications. [Online]. Available: https://github.com/nokia/wireless-suite

[7] E. Dahlman, S. Parkvall, and J. Skold, *5G NR: The next generation wireless access technology*. Academic Press, 2020.

[8] 3GPP, *TS 37.324, Service Data Adaptation Protocol (SDAP) specification (Release 17)*.

[9] ——, *TS 38.323, Packet Data Convergence Protocol (PDCP) specification (Release 17)*.

[10] ——, *TS 38.322, Radio Link Control (RLC) protocol specification (Release 17)*.

[11] ——, *TS 38.321, Medium Access Control (MAC) protocol specification (Release 17)*.

[12] ——, *TS 38.300, NR and NG-RAN Overall Description (Release 17)*.

[13] ——, *TR 21.905, Vocabulary for 3GPP Specifications (Release 17)*.

[14] F. Capozzi, G. Piro, L. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, 2013.

[15] A. Mamane, M. Fattah, M. E. Ghazi, M. E. Bekkali, Y. Balboul, and S. Mazer, "Scheduling algorithms for 5G networks and beyond: Classification and survey," *IEEE Access*, vol. 10, pp. 51 643–51 661, 2022.

[16] M. Elkael, M. Polese, R. Prasad, S. Maxenti, and T. Melodia, "ALLSTaR: Automated LLM-driven scheduler generation and testing for intent-based RAN," *arXiv preprint arXiv:2505.18389*, 2025.

[17] L. Kleinrock, "Analysis of a time-shared processor," *Naval research logistics quarterly*, vol. 11, no. 1, pp. 59–73, 1964.

[18] H. Hellerman, "Some principles of time-sharing scheduler strategies," *IBM Systems Journal*, vol. 8, no. 2, pp. 94–117, 1969.

[19] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar, "Providing quality of service over a shared wireless link," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 150–154, 2001.

[20] P. Viswanath, D. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Transactions on Information Theory*, vol. 48, no. 6, 2002.

[21] A. Stolyar and K. Ramanan, "Largest weighted delay first scheduling: Large deviations and optimality," *The Annals of Applied Probability*, vol. 11, 02 2001.

[22] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "CDMA data QoS scheduling on the forward link with variable channel conditions," *Technical Memorandum*, 2000.

[23] S. Shakkottai and A. L. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: The exponential rule," *Translations of the American Mathematical Society-Series 2*, vol. 207, 2002.

[24] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "Scheduling in a queuing system with asynchronously varying service rates," *Probability in the Engineering and Informational Sciences*, vol. 18, no. 2, 2004.

[25] J.-H. Rhee, J. Holtzman, and D.-K. Kim, "Scheduling of real/non-real time services: adaptive EXP/PF algorithm," in *The 57th IEEE Semiannual Vehicular Technology Conference, 2003. VTC 2003-Spring.*, vol. 1, 2003, pp. 462–466 vol.1.

[26] C. Wengerter, J. Ohlhorst, and A. von Elbwart, "Fairness and throughput analysis for generalized proportional fair frequency scheduling in OFDMA," in *2005 IEEE 61st Vehicular Technology Conference*, vol. 3, 2005, pp. 1903–1907 Vol. 3.

[27] B. Sadiq, S. J. Baek, and G. de Veciana, "Delay-optimal opportunistic scheduling and approximations: The log rule," in *IEEE INFOCOM 2009*, 2009, pp. 1692–1700.

[28] ——, "Delay-optimal opportunistic scheduling and approximations: The log rule," *IEEE/ACM Transactions on Networking*, vol. 19, no. 2, 2011.

[29] M. Brehm and R. Prakash, "Overload-state downlink resource allocation in LTE MAC layer," *Wireless networks*, vol. 19, pp. 913–931, 2013.

[30] N. Ferdosian, M. Othman, K. Y. Lun, and B. M. Ali, "Overload-state downlink resource scheduling and its challenges towards 5G networks," in *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2016, pp. 154–156.

[31] L. Georgiadis, M. J. Neely, L. Tassiulas *et al.*, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends® in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[32] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of queueing theory*. John wiley & sons, 2011, vol. 627.

[33] L. Kleinrock, "A conservation law for a wide class of queueing disciplines," *Naval Research Logistics Quarterly*, vol. 12, no. 2, pp. 181–192, 1965.

[34] I. Mitrani and P. J. King, "Multiprocessor systems with preemptive priorities," *Performance Evaluation*, vol. 1, no. 2, pp. 118–125, 1981.

[35] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *29th IEEE Conference on Decision and Control*. IEEE, 1990, pp. 2130–2132.

[36] S. Shakkottai and A. L. Stolyar, "Scheduling algorithms for a mixture of real-time and non-real-time data in HDR," in *Teletraffic Science and Engineering*. Elsevier, 2001, vol. 4.

[37] B. Sadiq, R. Madan, and A. Sampath, "Downlink scheduling for multiclass traffic in LTE," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, 2009.

[38] 3GPP, *TS 23.501, System architecture for the 5G System (5GS) (Release 17)*.

[39] I.-S. Comsa, A. De-Domenico, and D. Ktenas, "QoS-driven scheduling in 5G radio access networks - a reinforcement learning approach," in *IEEE Global Communications Conference (GLOBECOM)*, 2017.

[40] M. Seguin, A. Omer, M. Koosha, F. Malandra, and N. Mastronarde, "Deep reinforcement learning for downlink scheduling in 5G and beyond networks: A review," in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2023, pp. 1–6.

[41] F. Al-Tam, N. Correia, and J. Rodriguez, "Learn to schedule (LEASCH): A deep reinforcement learning approach for radio resource scheduling in the 5G MAC layer," *IEEE Access*, vol. 8, pp. 108 088–108 101, 2020.

[42] S. Nérondat, X. Leturc, P. Ciblat, and C. J. Le Martret, "Efficient 5G resource block scheduling using action branching and transformer networks," in *2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, 2025, pp. 1–6.

[43] I.-S. Comșa, S. Zhang, M. E. Aydin, P. Kuonen, Y. Lu, R. Trestian, and G. Ghinea, "Towards 5G: A reinforcement learning-based scheduling solution for data traffic management," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, 2018.

[44] S. Shen, T. Zhang, S. Mao, and G.-K. Chang, "DRL-based channel and latency aware radio resource allocation for 5G service-oriented RoF-MmWave RAN," *Journal of Lightwave Technology*, vol. 39, no. 18, 2021.

[45] J. Song, Y. Nam, H. Kwon, I. Sim, S. J. Maeng, and S. Jang, "Adaptive generalized proportional fair scheduling with deep reinforcement learning," in *IEEE GLOBECOM Workshops*, 2022.

[46] Y. Hao, F. Li, C. Zhao, and S. Yang, "Delay-oriented scheduling in 5G downlink wireless networks based on reinforcement learning with partial observations," *IEEE/ACM Transactions on Networking*, vol. 31, no. 1, 2023.

[47] J. Wang, C. Xu, Y. Huangfu, R. Li, Y. Ge, and J. Wang, "Deep reinforcement learning for scheduling in cellular networks," in *International Conference on Wireless Communications and Signal Processing (WCSP)*, 2019.

[48] F. AL-Tam, A. Mazayev, N. Correia, and J. Rodriguez, "Radio resource scheduling with deep pointer networks and reinforcement learning," in *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020.

[49] C. Xu, J. Wang, T. Yu, C. Kong, Y. Huangfu, R. Li, Y. Ge, and J. Wang, "Buffer-aware wireless scheduling based on deep reinforcement learning," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[50] A. Anand, R. Balakrishnan, V. S. Somayazulu, and R. Vannithamby, "Model-assisted deep reinforcement learning for dynamic wireless scheduling," in *Asilomar Conference on Signals, Systems, and Computers*, 2020.

[51] E.-M. Bansbach, V. Eliachevitch, and L. Schmalen, "Deep reinforcement learning for wireless resource allocation using buffer state information," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[52] W. AlQwider, T. F. Rahman, and V. Marojevic, "Deep Q-network for 5G NR downlink scheduling," in *IEEE ICC Workshops*, 2022.

[53] B. Toure, D. Tsilimantos, T. Giannakas, O. Esrafilian, and M. Kountouris, "Multi-objective scheduling in wireless networks with deep reinforcement learning," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025, pp. 1–6.

[54] B.-M. Robaglia, M. Coupechoux, and D. Tsilimantos, "Deep reinforcement learning for uplink scheduling in NOMA-URLLC networks," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 1142–1158, 2024.

[55] J. S. Shekhawat, R. Agrawal, K. G. Shenoy, and R. Shashidhara, "A reinforcement learning framework for QoS-driven radio resource scheduler," in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

[56] J. Li and X. Zhang, "Deep reinforcement learning-based joint scheduling of eMBB and URLLC in 5G networks," *IEEE Wireless Communications Letters*, vol. 9, no. 9, pp. 1543–1546, 2020.

[57] Z. Gu, C. She, W. Hardjawana, S. Lumb, D. McKechnie, T. Essery, and B. Vucetic, "Knowledge-assisted deep reinforcement learning in 5G scheduler design: From theoretical framework to implementation," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, 2021.

[58] S. Gracla, E. Beck, C. Bockelmann, and A. Dekorsy, "Learning resource scheduling with high priority users using deep deterministic policy gradients," in *ICC 2022-IEEE international conference on communications*. IEEE, 2022, pp. 4480–4485.

[59] A. Paz-Pérez, A. Tato, J. J. Escudero-Garzás, and F. Gómez-Cuba, "Flexible reinforcement learning scheduler for 5G networks," in *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, 2024, pp. 566–572.

[60] A. Avranas, P. Ciblat, and M. Kountouris, "Deep reinforcement learning for resource constrained multiclass scheduling in wireless networks," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 1, 2023.

[61] T. Zhang, S. Shen, S. Mao, and G.-K. Chang, "Delay-aware cellular traffic scheduling with deep reinforcement learning," in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

[62] N. Sharma, S. Zhang, S. R. Somayajula Venkata, F. Malandra, N. Mastronarde, and J. Chakareski, "Deep reinforcement learning for delay-sensitive LTE downlink scheduling," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–6.

[63] A. Robinson and T. Kunz, "Downlink scheduling in LTE with deep reinforcement learning, LSTMs and pointers," in *IEEE Military Communications Conference (MILCOM)*, 2021.

[64] V. H. L. Lopes, C. V. Nahum, R. M. Dreifuerst, P. Batista, A. Klautau, K. V. Cardoso, and R. W. Heath, "Deep reinforcement learning-based scheduling for multiband massive MIMO," *IEEE Access*, vol. 10, 2022.

[65] A. Giovanidis, M. Leconte, S. Aroua, T. Kvernvik, and D. Sandberg, "Online frequency scheduling by learning parallel actions," in *2024 3rd International Conference on 6G Networking (6GNet)*, 2024, pp. 153–160.

[66] X. Ye and L. Fu, "Joint MCS adaptation and RB allocation in cellular networks based on deep reinforcement learning with stable matching," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 549–565, 2024.

[67] Q. Wang, T. Nguyen, and B. Bose, "Towards adaptive packet scheduler with deep-Q reinforcement learning," in *International Conference on Computing, Networking and Communications (ICNC)*, 2020.

[68] S. Mollahasani, M. Erol-Kantarci, M. Hirab, H. Dehghan, and R. Wilson, "Actor-critic learning based QoS-aware scheduler for reconfigurable wireless networks," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, 2022.

[69] S. Nérondat, X. Leturc, C. J. Le Martret, and P. Ciblat, "Transformer-based packet scheduling under strict delay and buffer constraints," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2025.

[70] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

[71] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[72] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," *arXiv preprint arXiv:1511.06391*, 2015.

[73] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[74] H. Xu, L. Xiang, H. Ye, D. Yao, P. Chu, and B. Li, "Permutation equivariance of transformers and its applications," 2024. [Online]. Available: https://arxiv.org/abs/2304.07735

[75] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv preprint arXiv:2006.14171*, 2020.

[76] R. K. Jain, D.-M. W. Chiu, W. R. Hawe *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, p. 1, 1984.

[77] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *Proceedings of the aaai conference on artificial intelligence*, vol. 32, no. 1, 2018.

[78] A. Aniket and A. Chattopadhyay, "Online reinforcement learning in periodic MDP," *IEEE Transactions on Artificial Intelligence*, 2024.

[79] Z. Chen, K. K. Leung, S. Wang, L. Tassiulas, K. Chan, and P. J. Baker, "Learning technique to solve periodic markov decision process for network resource allocation," in *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, 2023, pp. 464–470.

[80] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[81] S. Chandra and A. K. Bharti, "Speed distribution curves for pedestrians during walking and crossing," *Procedia-Social and Behavioral Sciences*, vol. 104, pp. 660–667, 2013.

[82] G. Piro, N. Baldo, and M. Miozzo, "An LTE module for the ns-3 network simulator."

[83] 3GPP, *TS 38.214, Physical layer procedures for data (Release 17)*.

[84] P. M. de Sant Ana and N. Marchenko, "Radio access scheduling using CMA-ES for optimized QoS in wireless networks," in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020, pp. 1–6.

[85] S. Nérondat, X. Leturc, C. J. Le Martret, and P. Ciblat, "Procédé d'ordonnancement dynamique de communications entre une pluralité d'équipements utilisateurs," Patent FR 24 11 578, filed on October 23, 2024.

[86] J. Gaveau, "Allocation des ressources pour la gestion dynamique du spectre dans les réseaux ad hoc clustérisés," Ph.D. dissertation, Université Paris Saclay, soutenue le 11 juillet 2018.

[87] S. Nérondat, X. Leturc, C. J. Le Martret, and P. Ciblat, "Ordonnancement et ACM conjoint sur canal aléatoire basé sur un transformer entrainé par apprentissage profond par renforcement," in *GRETSI*, 2025.

[88] P. Series, "Multipath propagation and parameterization of its characteristics," *ITU recommmendations*, pp. 1407–8, 2021. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.1407-8-202109-I!!PDF-E.pdf

[89] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[90] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[91] P. Laige and T. Yoshimasa, "Improving action branching for deep reinforcement learning with a multi-dimensional hybrid action space," vol. 2019, pp. 80–85, 2019.

[92] D. P. Bertsekas *et al.*, *Dynamic programming and optimal control 4th edition, volume ii*, 2012.

[93] C. J. Watkins and P. Dayan, "Q-Learning," *Machine learning*, vol. 8, 1992.

[94] H. Van Hasselt, "Double Q-Learning," *Advances in neural information processing systems*, vol. 23, 2010.

[95] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5064–5078, 2024.

[96] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[97] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[98] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[99] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[100] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[101] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[102] K. Esslinger, R. Platt, and C. Amato, "Deep transformer Q-networks for partially observable reinforcement learning," *arXiv preprint arXiv:2206.01078*, 2022.

[103] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," *Advances in neural information processing systems*, vol. 34, pp. 1273–1286, 2021.

[104] P. Agarwal, A. A. Rahman, P.-L. St-Charles, S. J. Prince, and S. E. Kahou, "Transformers in reinforcement learning: A survey," *arXiv preprint arXiv:2307.05979*, 2023.

[105] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[106] D. Bahdanau, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[107] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[108] L. Weng, "Attention? attention!" *lilianweng.github.io*, 2018. [Online]. Available: https://lilianweng.github.io/posts/2018-06-24-attention/

[109] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[110] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv e-prints*, pp. arXiv–1607, 2016.

[111] M. Phuong and M. Hutter, "Formal algorithms for transformers," *arXiv preprint arXiv:2207.09238*, 2022.

[112] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[113] I. S. Gradshteyn and I. M. Ryzhik, *Table of integrals, series, and products*. Academic press, 2014.

ECOLE
DOCTORALE

Titre : Apprentissage par renforcement profond pour l'optimisation conjointe de l'allocation de ressource et de l'ordonnancement pour les réseaux ad hoc

Mots clés : Allocation de ressources, Ordonnancement, Apprentissage par renforcement profond

Résumé : Les réseaux de communication modernes font face à des défis croissants en raison de la diversité des applications, allant des services ultra-fiables à faible latence aux communications massives de type IoT. Ces applications génèrent des demandes hétérogènes nécessitant une planification efficace des paquets, qui consiste à allouer les ressources radio dans le temps (slots) et en fréquence (blocs de ressources, RBs) afin de garantir le débit, l'équité et la latence. Les méthodes heuristiques traditionnelles manquent de flexibilité pour gérer ces besoins contradictoires, ce qui rend les solutions basées sur l'IA plus adaptées, en particulier pour les systèmes 5G et 6G.
Cette thèse est consacrée au développement de méthodes basées sur l'IA pour la planification dans les réseaux de communication sans fil. Le problème d'optimisation global implique un espace d'états et d'actions de grande dimension et ne peut être résolu par des méthodes analytiques ; d'où l'adoption de techniques d'apprentissage par renforcement, combinées à des réseaux de neurones profonds. L'objectif est d'explorer comment le DRL peut être utilisé efficacement pour optimiser l'allocation des ressources dans des environnements complexes, caractérisés par des schémas de trafic variés et des conditions de canal dynamiques.
Pour ce faire, nous identifions d'abord les propriétés essentielles qu'un réseau de neurones doit satisfaire pour réaliser la planification, en particulier la capacité à gérer un nombre variable d'utilisateurs. Nous évaluons ensuite une architecture répondant à ces critères dans des conditions de complexité croissante. Dans un premier temps, la solution est testée dans un environnement slot par slot avec un seul RB par slot. Ensuite, elle est étendue afin de réaliser la planification sur plusieurs RBs simultanément. Enfin, l'approche est adaptée pour traiter conjointement la planification des RBs et la sélection du schéma de modulation et de codage (MCS).

Title : Deep reinforcement learning for joint optimization of scheduling and resource allocation in mobile ad hoc networks

Keywords : Resource allocation, Scheduling, Deep reinforcement learning

Abstract : Modern communication networks face rising challenges due to diverse applications, from ultra-reliable low-latency services to massive IoT traffic. These generate heterogeneous demands requiring efficient packet scheduling, which allocates radio resources in time (slots) and frequency (resource blocks, RBs) to ensure throughput, fairness, and latency. Traditional heuristic methods lack the flexibility to manage these conflicting needs, making AI-based solutions more suitable, especially for 5G and 6G systems.
This thesis is devoted to the development of AI-based methods for scheduling in wireless communication networks. The global optimization problem involves a high-dimensional state and action space and cannot be solved using analytical methods, hence the adoption of reinforcement learning techniques powered by deep neural networks. The objective is to explore how DRL can be effectively employed to optimize resource allocation in complex environments characterized by diverse traffic patterns and dynamic channel conditions.
To achieve this, we first identify the essential properties a neural network must satisfy for scheduling, particularly the ability to handle a variable number of users. We then evaluate an architecture that meets these requirements under increasingly complex conditions. Initially, the solution is tested in a slot-by-slot environment with a single RB per slot. Next, it is extended to perform scheduling across multiple RBs simultaneously. Finally, the approach is adapted to jointly handle RB scheduling and modulation and coding scheme (MCS) selection.

Institut Polytechnique de Paris
91120 Palaiseau, France