N° d'ordre : 100/2024-D/INF

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie
- Houari Boumediène -
FACULTÉ D'INFORMATIQUE

Thèse de doctorat en Sciences

Présentée pour l'obtention du grade de docteur

Spécialité :

Informatique

Par :
M. NACEF Abdelhakim

Thème

Vers des réseaux automatisés basés sur la technologie SDN (Software-Defined Networking)

Soutenue publiquement, le 10/07/2024, devant le jury composé de :

M.	Mohamed GUERROUMI	Professeur	à l'USTHB, Alger	Président
M.	Youcef AKLOUF	Professeur	à l'USTHB, Alger	Directeur de thèse
M.	Rachid CHALAL	Professeur	à l'ESI, Alger	Examinateur
M.	Chaouki BOUFENAR	Maître de Conférences A	à l'Université d'Alger 1	Examinateur
M.	Khaled ZERAOULIA	Maître de Conférences A	à l'USTHB, Alger	Examinateur
M.	Djamel TANDJAOUI	Directeur de Recherche	au CERIST, Alger	Examinateur

Année universitaire 2023/2024

Dédicace

Je tiens à dédier ce modeste travail à mon défunt père.
A ma chère mère.
A ma femme et mes enfants qui ont partagé mes joies et mes pensées.
A mes chers frères et sœurs.
A toutes les personnes qui ont fait rentrer de la joie dans ma vie.

Abdelhakim

Vers des réseaux automatisés basés sur la technologie SDN (Software-Defined Networking)

Résumé

De nos jours, les réseaux de communication jouent un rôle vital dans la connectivité et la diffusion d'informations à travers une multitude d'applications et de services. Cependant, l'évolution rapide du paysage numérique engendre des demandes toujours plus exigeantes en matière de qualité de service (QoS), auxquelles les infrastructures réseau traditionnelles peinent à répondre efficacement. Cette thèse explore une approche originale intégrant le Software-Defined Networking (SDN) et le Machine Learning (ML). Son objectif principal est de concevoir une plateforme générique, dénommée DLO-SFC, pour un réseau intelligent et autonome, capable de s'auto-optimiser et de s'auto-orchestrer, en tenant compte des besoins évolutifs des utilisateurs. À cet effet, DLO-SFC utilise des techniques d'apprentissage profond et d'optimisation de réseau pour fournir des configurations optimales rapides pour tout problème de configuration et d'orchestration de réseau. Les résultats des expérimentations ont démontré la capacité de DLO-SFC à converger rapidement vers des configurations optimales, engendrant ainsi une réduction des coûts d'exploitation (OpEx) et une optimisation des temps de réponse.

Mots-clés : Software-Defined Networking (SDN), Machine Learning (ML), Optimisation de réseau, Routage multi-chemins, Qualité de service (QoS).

Towards automated networks based on SDN (Software-Defined Networking) technology

Abstract

Nowadays, communication networks play a vital role in connectivity and the dissemination of information across a multitude of applications and services. However, the rapid evolution of the digital landscape generates increasingly demanding requirements in terms of Quality of Service (QoS), to which traditional network infrastructures struggle to respond effectively. This thesis explores an original approach integrating Software-Defined Networking (SDN) and Machine Learning (ML). Its main objective is to design a generic platform, named DLO-SFC, for an intelligent and autonomous network capable of self-optimizing and self-orchestrating, taking into account the evolving needs of users. To this end, DLO-SFC employs deep learning techniques and network optimization to provide rapid optimal configurations for any network configuration and orchestration problem. The experimental results have demonstrated DLO-SFC's ability to quickly converge towards optimal configurations, thereby reducing operational costs (OpEx) and optimizing response times.

Keywords : Software-Defined Networking (SDN), Machine Learning (ML), Network Optimization, Multi-path routing, Quality of Service (QoS).

نحو شبكات مؤتمتة تعتمد على تكنولوجيا SDN (الشبكات المعرفة بالبرمجيات)

ملخص

في أيامنا هذه، تلعب شبكات الاتصالات دورًا حيويًا في التواصل ونشر المعلومات عبر مجموعة متنوعة من التطبيقات والخدمات. ومع ذلك، فإن التطور السريع للمشهد الرقمي ينتج متطلبات متزايدة باستمرار فيما يتعلق بجودة الخدمة (QoS) ، والتي تجد البنى التحتية الشبكية التقليدية صعوبة في التعامل معها بشكل فعال. تستكشف هذه الأطروحة نهجًا مبتكرًا يدمج الشبكات المعرفة بالبرمجيات (SDN) والتعلم الآلي (ML) . هدفها الرئيسي هو تصميم منصة عامة، تسمى (DLO-SFC) ، لشبكة ذكية ومستقلة، قادرة على التحسين الذاتي والتنسيق الذاتي، مع مراعاة الاحتياجات المتطورة للمستخدمين. لهذا الغرض، تستخدم (DLO-SFC) تقنيات التعلم العميق وتحسين الشبكة لتوفير تكوينات مثلى سريعة لأي مشكلة تكوين وتنسيق شبكة. أظهرت نتائج التجارب قدرة (DLO-SFC) على التقارب بسرعة نحو التكوينات المثلى، مما أدى إلى تقليل التكاليف التشغيلية (OpEx) وتحسين أوقات الاستجابة.

الكلمات المفتاحية

الشبكات المعرفة بالبرمجيات (SDN) ، التعلم الآلي (ML) ، تحسين الشبكة، التوجيه متعدد المسارات، جودة الخدمة (QoS) .

Remerciements

Avant tout, je tiens à remercier Dieu pour m'avoir donné la force et la persévérance nécessaires pour mener à bien ce projet de recherche.

Je voudrais également exprimer ma profonde gratitude à mes amis Dr. Abdellah KACI et Dr. Miloud BAGAA pour leur précieuse aide tout au long de cette aventure. Leurs conseils, leurs encouragements et leur soutien moral ont été essentiels pour surmonter les moments difficiles et avancer dans la bonne direction.

Je tiens à remercier chaleureusement mon directeur de thèse, Pr. AKLOUF Youcef, pour sa confiance, son soutien et ses précieux conseils tout au long de la réalisation de ce travail de recherche.

Je remercie les membres de mon jury d'avoir accepté d'évaluer mon travail, M. Mohamed GUERROUMI, M. Rachid CHALAL, M. Chaouki BOUFENAR, M. Khaled ZERAOULIA et M. Djamel TANDJAOUI.

Je remercie aussi tous les collègues et amis qui ont participé à ce projet de recherche d'une manière ou d'une autre.

Enfin, je tiens à exprimer ma reconnaissance à ma famille pour leur amour, leur soutien et leur compréhension tout au long de ce parcours académique.

Merci infiniment à tous ceux qui ont contribué à la réalisation de ce travail de recherche et à cette étape importante de ma vie.

Table des matières

Page

Table des Figures **iii**

Liste des Tableaux **iv**

Liste des Algorithmes **iv**

Liste des Abreviations **v**

Introduction **1**
0.1 Contexte 1
0.2 Problématique 3
0.3 Objectif 4
0.4 Organisation de la thèse 5

I Partie 1 : État de l'art **7**

1 Réseaux définis par logiciel et optimisation du routage multi-chemins **8**
1.1 Introduction 8
1.2 Réseaux définis par logiciel (SDN) 9
1.2.1 Histoire de SDN 10
1.2.2 Concept SDN 12
1.2.3 Contrôleurs SDN 14
1.2.4 Environnement type de SDN 14
1.3 Virtualisation des fonctions réseau (NFV) 17
1.3.1 Fonctions réseau virtualisées (VNF) 18
1.3.2 Chaînage de fonctions service (SFC) 19
1.3.3 Synthèse 20
1.4 Techniques d'optimisation 20
1.4.1 Problème d'optimisation 20
1.4.2 Classification des techniques d'optimisation 21
1.4.3 Optimisation discrète 22
1.4.4 Optimisation continue 23
1.5 Optimisation du routage multi-chemins dans les réseaux SDN 25
1.6 Conclusion 28

2 Utilisation de l'apprentissage automatique dans les réseaux définis par logiciel 31
2.1 Introduction . . . 31
2.2 Intelligence artificielle . . . 32
2.2.1 Définition . . . 32
2.2.2 Histoire de l'intelligence artificielle . . . 32
2.3 Machine learning . . . 36
2.3.1 Définition . . . 36
2.3.2 Classification de l'apprentissage automatique . . . 37
2.3.3 Évaluation des performances des modèles d'apprentissage automatique 40
2.4 Deep Learning . . . 43
2.4.1 Définition . . . 43
2.4.2 Classification de Deep Learning . . . 44
2.4.3 Réseaux de Neurones . . . 44
2.4.4 Réseaux de neurones convolutifs (CNN) . . . 50
2.5 Utilisation du Machine Learning dans les réseaux définis par logiciel . . . 54
2.6 Conclusion . . . 57

II Partie 2 : Contributions 60

3 Framework d'optimisation basé sur l'IA 61
3.1 Introduction . . . 61
3.2 DLO-SFC : Framework d'optimisation basé sur le *Deep Learning* . . . 62
3.2.1 Vue générale de la plateforme DLO-SFC . . . 62
3.2.2 Composants de la plateforme DLO-SFC . . . 63
3.3 Déploiement de la plateforme DLO-SFC . . . 68
3.4 Conclusion . . . 70

4 Routage multi-chemins intelligent 72
4.1 Introduction . . . 72
4.2 Optimisation du routage multi-chemins dans les SDN . . . 73
4.2.1 FPR : Full Paths Re-computation . . . 74
4.2.2 HPR : Heuristic Paths Re-computation . . . 74
4.3 iPare : Intelligent Path Re-computation . . . 76
4.3.1 Présentation d'iPare . . . 77
4.3.2 Entraînement du modèle *Mc-DLN* . . . 80
4.3.3 Exemple illustratif . . . 81
4.3.4 Model Checker (MC) . . . 84
4.4 Routage multi-chemins dans la plateforme DLO-SFC . . . 86
4.5 Conclusion . . . 90

5 Étude expérimentale 92
5.1 Introduction . . . 92
5.2 Configuration expérimentale . . . 93
5.3 Évaluation du mode d'entraînement iPare . . . 98
5.4 Évaluation du mode d'inférence d'iPare . . . 99
5.5 Conclusion . . . 109

Conclusion et perspectives **111**
Conclusion 111
Perspectives 112

Bibliographie **114**

A Publications et contributions scientifiques **121**
A.1 Liste des publications 121
A.2 Contributions scientifiques 121

Table des figures

		Page
1	Réseau mobile 5G	5
1.1	Architecture SDN	12
1.2	Logo d'Open vSwitch de la *Linux Foundation*	16
1.3	Architecture de la NFV	18
1.4	Exemple d'utilisation du SFC	19
1.5	Classification des techniques d'optimisation	21
2.1	Classification de l'apprentissage automatique[1]	37
2.2	Exemple de Régression et de Classification	38
2.3	Fonctions Linéaire et Non Linéaire	39
2.4	Deep Learning et l'IA	43
2.5	Types de modèles de *Deep Learning*	44
2.6	Perceptron	45
2.7	Etapes d'apprentissage	47
2.8	Perceptron multicouches	48
2.9	Différentes fonctions d'activation	49
2.10	Opération de convolution	51
2.11	Couche de convolution	52
2.12	Opération de *padding*	52
2.13	Opération de *pooling*	53
2.14	Opération de *flattening*	54
2.15	Architecture d'un *CNN*	54
3.1	Architecture du Framework d'optimisation *DLO-SFC*	63
3.2	Diagramme de séquence du framework DLO-SFC	65
3.3	Diagramme d'activité du framework DLO-SFC	67
3.4	Déploiement de la plateforme DLO-SFC dans un Cluster Kubernetes	70
4.1	Architecture d'*iPare*	78
4.2	Exemple illustratif	82
4.3	Architecture micro-services de la plateforme DLO-SFC pour le routage multi-chemins	87
5.1	Architecture du CNN (Mc-DLN)	94
5.2	La perte (*Loss*) et la précision (*Accuracy*) d'iPare pendant le mode d'entraînement.	97

5.3 La complexité d'exécution de la solution iPare comparée à FPR et HPR, en fonction du nombre de clients. 100
5.4 La complexité d'exécution de la solution iPare comparée à FPR et HPR, en fonction du nombre de OVSs. 101
5.5 La complexité d'exécution de la solution iPare comparée à FPR et HPR, en fonction du nombre de serveurs. 101
5.6 L'effet du seuil d'activation (*Activation threshold*) sur la précision (*Accuracy*) et la surcharge (*Overhead*) d'iPare pendant le mode d'inférence. . . . 104
5.7 Comparaison avec la solution heuristique *HPR* en fonction de la variation du nombre de OvSs. 106
5.8 Comparaison avec la solution heuristique *HPR* en fonction de la variation du nombre de clients. 107
5.9 Comparaison avec la solution heuristique *HPR* en fonction de la variation du nombre de serveurs. 107

Liste des tableaux

Page

1.1 Comparaison entre le Réseau Traditionnel et SDN [2] 10
1.2 Contrôleurs SDN Open Source 15
1.3 Paramètres de modélisation du problème du plus court chemin. 24
1.4 Travaux de recherche sur le routage multi-chemins dans les réseaux SDN. . 30

2.1 Matrice de confusion 41
2.2 Exemple illustrant le *MAPE* 43
2.3 Travaux de recherche sur l'apprentissage automatique et les réseaux SDN . 58

3.1 Synthèse des notations utilisées 62

4.1 Synthèse des notations utilisées 77
4.2 The Capacity Channel (CC) 83
4.3 The Request Channel (RC) 83
4.4 The Allocation Channel (AC) 84
4.5 Description des opérations 87

5.1 Hyperparamètres du modèle CNN (Mc-DLN) 94
5.2 Spécifications matérielles et logicielles utilisées 95
5.3 Données expérimentales 95

Liste des abreviations

CapEx	Capital Expense
DLC	Deep Learning Component
DL	Deep Learning
DLO-SFC	Deep Learning Optimization for Service Function Chaining
FPR	Full Paths Re-computation
HC	Heuristic Component
HPR	Heuristic Paths Re-computation
IA	Intelligence Artificielle
iPare	Intelligent Path Re-computation
Mc-DLN	Multi-label Classification Deep Learning Network
ML	Machine Learning
ND	Network Driver
NCG	Network Configuration Generator
NFV	Network Function Virtualization
NMA	Network Monitoring Agent
OC	Optimisation Component
OpEx	Operating Expense
OvS	Open vSwitch
SDN	Software-Defined Networking
SFC	Service Function Chaining
VNF	Virtualized Network Functions

Introduction

> La civilisation a pour but non pas le progrès de la science et des machines, mais celui de l'homme.
>
> Alexis Carrel (1873-1944)

0.1 Contexte

L'évolution rapide des technologies de l'information et de la communication a engendré une augmentation considérable de l'utilisation des réseaux à travers le monde, plus particulièrement Internet. En effet, selon une étude de Cisco [3], il est projeté qu'en 2023, le nombre d'utilisateurs d'Internet atteindra 5,3 milliards, marquant une expansion sans précédent de l'accès à la toile.

Cette croissance exponentielle s'accompagne d'une hausse significative de la connectivité, avec une moyenne estimée à 3,6 appareils et connexions par habitant. Cette tendance reflète l'omniprésence croissante des technologies numériques dans la vie quotidienne.

Egalement, la vitesse de la bande passante fixe atteindra en moyenne 110 Mbps à l'échelle mondiale. Alors que les connexions Machine-To-Machine (M2M) représenteront la moitié des appareils et connexions connectés dans le monde en 2023, avec 14,7 milliards de connexions M2M.

En 2022, le trafic IP mondial a atteint une estimation de 4,8 Zettaoctets, ce qui équivaut à environ 396 Exaoctets par mois. Le trafic IP, représentant la somme des données qui transitent sur Internet, constitue le flux essentiel des échanges numériques à travers le monde. Les experts prévoient un triplement de ce volume dans les cinq prochaines années.

Parallèlement, de nouvelles applications en ligne ont surgi. Par exemple, le streaming vidéo de haute qualité en ligne (Netflix, Amazon Prime Video, Disney+), tout comme les plateformes de partage de vidéos (TikTok, Vimeo, YouTube), imposent des exigences de bande passante et de latence minimale considérables. Ces services en ligne ont révolutionné la manière dont nous consommons et partageons du contenu multimédia, tout en mettant à l'épreuve les infrastructures des réseaux de communication.

Plus récemment, l'émergence des réseaux mobiles de 5ème et de 6ème génération n'a fait qu'accentuer l'explosion de l'utilisation des données. Ces progrès technologiques se traduisent par des vitesses de transmission plus rapides et une connectivité plus fiable, offrant

aux utilisateurs un accès fluide à une multitude de services en ligne ainsi qu'à des applications nécessitant une importante consommation de données, dans un monde de plus en plus connecté et axé sur la mobilité.

Selon une récente étude [4], en une seule minute dans le monde numérique, nous sommes confrontés à un volume important de données à faire transiter, nécessitant une infrastructure de réseau capable de les acheminer efficacement. Cette réalité souligne l'importance d'avoir des réseaux robustes et hautement performants.

Par exemple, toutes les 60 secondes, nous sommes confrontés à une multitude de données :

— 241,2 millions d'e-mails envoyés.
— 18,8 millions de messages SMS transmis.
— 2,4 millions de recherches sur Google.
— 694 000 heures de vidéos visualisées.
— 347 222 tweets partagés.
— 271 309 applications téléchargées sur iOS et Android.

Dans ce contexte en constante évolution, la transmission de volumes importants de données est devenue primordiale pour répondre à divers besoins et applications.

Cette évolution technologique rapide a entraîné des changements importants dans de nombreux domaines, notamment dans celui des réseaux de communication, qui ont dû s'adapter pour répondre aux exigences croissantes des utilisateurs.

Cependant, il est de plus en plus évident que les méthodes traditionnelles de mise en réseau, souvent rigides et statiques, sont insuffisantes pour répondre de manière efficace et flexible aux besoins des utilisateurs dans des environnements réseau en constante évolution.

Par conséquent, les gestionnaires de réseaux sont confrontés à des défis de plus en plus complexes lorsqu'il s'agit de gérer et de maintenir leur infrastructure. Au cœur de leurs préoccupations se trouve la quête d'une optimisation des temps de réponse aux requêtes et de la satisfaction des diverses exigences des utilisateurs, qu'elles soient d'origine humaine ou applicative.

C'est dans ce contexte que s'inscrit la présente thèse, qui se propose d'étudier les enjeux et les défis de l'optimisation des réseaux de communication pour faire face aux évolutions rapides de l'utilisation d'Internet et des technologies associées.

Dans cette optique, notre attention se portera principalement sur les réseaux mobiles, qui ont connu une croissance exponentielle au cours des dernières années et ont émergé en tant qu'acteurs majeurs de la connectivité mondiale.

Il convient de souligner que, parallèlement à l'avènement des smartphones et des tablettes, les utilisateurs sont devenus de plus en plus dépendants des réseaux mobiles pour accéder à une multitude de services en ligne, notamment les réseaux sociaux, la messagerie instantanée et le streaming.

Par conséquent, il s'avère impératif de développer une compréhension de ces réseaux et de les optimiser afin de répondre de manière adéquate à la demande croissante des utilisateurs en matière de connectivité, tout en assurant une qualité de service (QoS) optimale.

0.2 Problématique

De nos jours, les réseaux de communication jouent un rôle vital dans la connectivité mondiale et la diffusion d'informations à travers une multitude d'applications et de services.

Toutefois, le paysage numérique évolue à un rythme effréné, générant des demandes toujours plus exigeantes de la part des utilisateurs en matière de qualité de service (QoS) : bande passante élevée, latence réduite et disponibilité constante.

Face à cette évolution rapide, les réseaux traditionnels se trouvent confrontés à des lacunes majeures qui les empêchent de satisfaire pleinement ces nouvelles exigences.

La problématique qui se dessine au cœur de cette thèse se concentre sur la nécessité impérative de repenser les architectures de réseaux de communication afin de surmonter les limitations intrinsèques des approches conventionnelles.

Alors que les réseaux traditionnels ont été conçus pour répondre à des besoins qui étaient autrefois considérés comme adéquats, ils montrent désormais leurs limites face à la réalité d'un monde interconnecté et en constante évolution.

L'une des principales lacunes des réseaux traditionnels réside dans leur manque de flexibilité et d'adaptabilité. Les architectures fixes et hiérarchiques, conçues pour des charges de travail prévisibles, peinent à gérer la variabilité des demandes d'aujourd'hui.

Les applications gourmandes en bande passante, telles que le streaming vidéo haute qualité et la réalité augmentée/virtuelle, requièrent une répartition de la bande passante dynamique et une faible latence pour offrir une expérience utilisateur optimale.

De même, la prolifération des objets connectés et l'essor de l'Internet des Objets (IoT) imposent des contraintes nouvelles en termes de gestion de trafic et de connectivité, auxquelles les réseaux traditionnels peinent à répondre de manière efficace.

Une autre problématique majeure concerne la sécurité et la confidentialité des données échangées. Les réseaux traditionnels, souvent centralisés, sont vulnérables aux attaques et aux compromissions. Les exigences croissantes en matière de protection des données personnelles et sensibles imposent de repenser les mécanismes de sécurité en profondeur, en intégrant des approches novatrices de chiffrement, d'authentification et de contrôle d'accès.

Face à ces enjeux, la thèse se propose d'explorer une approche originale axée sur la conception et la mise en œuvre de réseaux de communication résilients et adaptatifs, capables de répondre de manière proactive et dynamique aux besoins changeants des utilisateurs. Cette démarche impliquera l'intégration de concepts avancés tels que le Software-Defined Networking (SDN) et le Machine Learning (ML) pour offrir des solutions flexibles, sécurisées et évolutives.

En résumé, cette thèse vise à adresser la problématique cruciale de l'évolution des réseaux de communication pour répondre aux attentes grandissantes des utilisateurs dans un monde en perpétuelle transformation. En explorant des approches novatrices et interdisciplinaires, elle cherche à repenser les fondements mêmes des réseaux traditionnels, en vue de proposer des infrastructures résilientes, adaptables et sécurisées, capables de relever les défis de la demande croissante et diversifiée des utilisateurs modernes.

0.3 Objectif

L'objectif fondamental de cette thèse est de créer une approche originale et intégrée qui exploite synergiquement les avantages du Software-Defined Networking (SDN) et du Machine Learning (ML) pour surmonter les limites des réseaux de communication traditionnels.

Dans un contexte où les exigences des utilisateurs évoluent rapidement et où la performance du réseau est cruciale pour une variété d'applications, notre travail vise à concevoir un framework générique pour la création d'un réseau intelligent et autonome.

Ce framework vise spécifiquement à réduire l'intervention humaine et à créer un environnement où le réseau s'auto-optimise et s'auto-orchestre en tenant compte des informations contextuelles de son environnement.

La diversité des applications modernes constitue un élément central de nos objectifs. Par exemple, alors que les domaines de la conduite autonome, de l'industrie 4.0 et des applications XR (réalité étendue), telles que la chirurgie collaborative à distance, requièrent une quasi-absence de latence et une fiabilité élevée, les secteurs liés à la NB-IoT (*Narrowband Internet of Things*, en anglais) et aux machines agricoles nécessitent une couverture en profondeur et une faible consommation d'énergie [5].

Dans ce contexte, cette thèse propose la nécessité d'un réseau de communication end-to-end autogéré, auto-orchestré et auto-optimisé, y compris au niveau des nuages (*Clouds*, en anglais), afin de répondre à ces exigences.

En plus de l'approche du Software-Defined Networking et de la virtualisation des fonctions réseau (VNF), le Machine Learning est attendu pour jouer un rôle crucial dans la prochaine génération de réseaux, notamment les environnements 5G et 6G.

La Figure 1 illustre le rôle attendu de la technologie SDN dans le réseau mobile 5G [5]. Tandis que les plans d'orchestration, de gestion et de contrôle assurent une fonctionnalité optimale des fonctions réseau virtualisées (VNF) au niveau utilisateur, le contrôleur SDN gère le réseau sous-jacent pour interconnecter efficacement ces fonctions réseau. Cette conception permet d'assurer que les équipements des utilisateurs, tels que les smartphones, les tablettes et les ordinateurs portables, puissent se connecter au réseau et accéder aux services de manière transparente et efficace.

Ainsi, le document propose un framework générique nommé *DLO-SFC*, qui combine les techniques d'apprentissage profond (deep learning) et l'optimisation réseau pour fournir des configurations optimales rapides pour toute configuration et problématique d'orchestration réseau.

Au cœur de cette démarche se trouve la volonté de fournir des solutions concrètes et efficaces pour la gestion et la configuration des réseaux de communication modernes. Notre framework se positionne comme une réponse proactive aux limitations des approches traditionnelles, en proposant des solutions intelligentes et adaptatives capables de s'ajuster en continu aux conditions variables du réseau.

Dans cette perspective, notre objectif global est de contribuer significativement dans le progrès des réseaux de communication en proposant un ensemble de solutions novatrices et performantes. Celles-ci reposent sur une intégration synergique du SDN (Software-Defined Networking) et du ML (Machine Learning).

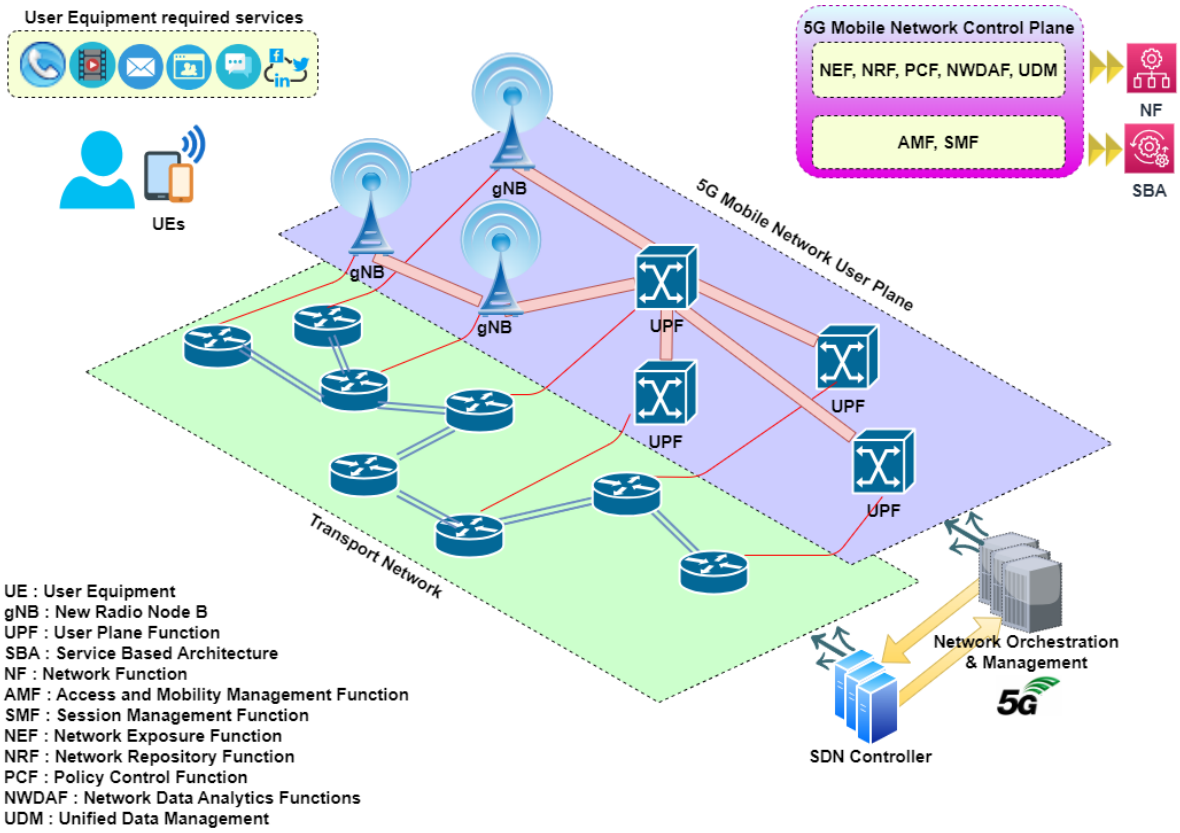


FIGURE 1 – Réseau mobile 5G

En somme, cette thèse s'efforce de façonner un avenir où les réseaux de communication répondent avec agilité et efficacité aux demandes en constante évolution, tout en maintenant une performance optimale.

0.4 Organisation de la thèse

Cette thèse est structurée en cinq chapitres, chacun explorant un aspect particulier des réseaux SDN et de l'intégration de l'apprentissage automatique :

Partie 1 : État de l'art

Chapitre 1 : Réseaux définis par logiciel et optimisation du routage multi-chemins

Ce chapitre examine les fondements des réseaux définis par logiciel (SDN), leur évolution historique, ainsi que les techniques d'optimisation du routage multi-chemins.

Chapitre 2 : Utilisation de l'apprentissage automatique dans les réseaux définis par logiciel (SDN)

Ce chapitre explore l'utilisation de l'apprentissage automatique dans les réseaux SDN, depuis ses fondements théoriques jusqu'à ses applications pratiques.

Partie 2 : Contributions

Chapitre 3 : Framework d'optimisation basé sur l'IA

Ce chapitre présente un framework d'optimisation basé sur l'intelligence artificielle développé dans le cadre de cette thèse. Il est conçu pour automatiser la configuration des réseaux SDN de manière efficace et fiable.

Chapitre 4 : Routage multi-chemins intelligent

Ce chapitre propose une approche originale de routage multi-chemins intelligent dans le contexte des réseaux définis par logiciel (SDN), visant à optimiser l'allocation des ressources réseau tout en garantissant une qualité de service optimale.

Chapitre 5 : Étude expérimentale

Ce chapitre présente une étude expérimentale approfondie, évaluant les performances des contributions proposées à travers des simulations et des analyses comparatives. Cette partie vise à valider empiriquement les approches proposées et à identifier leurs forces et leurs limitations dans des scénarios réalistes.

Chapitre 5.5 : Conclusion et Perspectives

Ce dernier chapitre marque la fin du travail présenté et résume les résultats obtenus. La conclusion générale met en avant l'approche proposée, qui combine le réseau défini par logiciel (SDN) et l'Intelligence Artificielle (IA) pour permettre une gestion automatisée et flexible des réseaux. Les tests menés ont montré une nette amélioration de l'agilité, de la flexibilité et de l'évolutivité des réseaux, ainsi qu'une réduction significative des coûts d'opération (OpEx).

Ce chapitre offre également un aperçu des diverses perspectives qui pourraient être explorées dans le cadre de ce travail, notamment l'amélioration de l'approche de virtualisation proposée, la mise en place de tests à plus grande échelle et l'exploration de nouveaux cas d'utilisation. Ces perspectives pourraient aider à étendre les résultats obtenus dans cette étude et à explorer davantage l'approche proposée pour la gestion des réseaux.

Première partie

Partie 1 : État de l’art

Chapitre 1

Réseaux définis par logiciel et optimisation du routage multi-chemins

La science est un train que le
mécanicien ne peut arrêter.
Frédéric Dard (1921-2000)

1.1 Introduction

Les réseaux informatiques ont connu une transformation significative au cours des décennies, évoluant des architectures traditionnelles vers des approches plus modernes et flexibles [6] pour répondre aux demandes croissantes des applications et d'exigences des utilisateurs. Au cœur de cette révolution se trouve le Software-Defined Networking (SDN), ou réseaux définis par logiciel. SDN est une solution innovante qui offre une approche centralisée en matière de gestion des réseaux.

Ce présent chapitre explore les concepts fondamentaux des réseaux définis par logiciel, mettant en lumière leur évolution, leur histoire, les composants clés, et les avantages qu'ils offrent en termes de flexibilité, d'automatisation renforcée et de réduction des coûts [7].

Dans la Section 1.2, nous abordons l'univers du SDN, décrivant comment cette technologie a émergé comme une réponse aux limitations des réseaux traditionnels. La centralisation du contrôle, la programmabilité du réseau, et une gestion plus intelligente caractérisent le SDN, offrant ainsi une alternative dynamique aux configurations statiques et aux interventions manuelles fastidieuses.

Le parcours historique du SDN présenté révèle son origine dans les recherches des années 1990 sur la programmabilité des réseaux. L'OpenSig et l'initiative Active Networking [8] ont joué des rôles clés en explorant la séparation du plan de contrôle et du plan de données. Cependant, c'est avec l'avènement d'OpenFlow en 2008 que cette idée a pris une dimension pratique dans l'industrie [9].

Dans la partie dédiée au concept du SDN, nous examinons de près l'architecture du SDN, caractérisée par la séparation claire entre le plan de contrôle et le plan de données. Nous explorons également des composants essentiels tels que les contrôleurs SDN, qui jouent un rôle

central dans cette architecture, ainsi que des environnements types de SDN, illustrés à travers des exemples concrets tels que le contrôleur SDN ONOS, la virtualisation avec Mininet, la communication Sud avec OpenFlow, et l'utilisation du switch logiciel Open vSwitch (OvS).

La Section 1.3 du chapitre se concentre sur la virtualisation des fonctions réseau (NFV), qui représente une autre innovation majeure dans le domaine des réseaux. L'initiative NFV vise à virtualiser les fonctions réseau traditionnelles, offrant une flexibilité accrue et des économies substantielles[10]. Nous explorons les Virtualized Network Functions (VNF) et la notion de Chaîne de Fonctions de Service (SFC) qui ont émergé dans le contexte de NFV, soulignant ainsi leur rôle crucial dans la transformation des réseaux de communication.

La Section 1.4 du chapitre se tourne vers les techniques d'optimisation, en mettant l'accent sur les approches heuristiques et la programmation linéaire en nombres entiers. Nous classifions les différentes techniques d'optimisation connues, et explorons les méthodes spécifiques applicables aux domaines de l'optimisation discrète et continue.

Enfin, la Section 1.5 de ce chapitre présente certains travaux de recherche portant sur l'optimisation du routage multi-chemins dans les réseaux SDN. En examinant la littérature existante, nous identifions des travaux pertinents et positionnons notre contribution dans ce contexte, visant à proposer un nouveau framework agile qui combine l'optimisation du réseau et les techniques d'apprentissage automatique pour configurer efficacement les réseaux SDN tout en garantissant fiabilité et qualité de service.

1.2 Réseaux définis par logiciel (SDN)

Les réseaux informatiques ont subi une évolution importante au fil des décennies, passant des architectures de réseau traditionnelles vers des approches plus modernes et flexibles. Cette évolution a été en grande partie stimulée par la nécessité d'adapter rapidement les infrastructures réseau aux exigences croissantes des applications et des utilisateurs. Dans ce contexte, le Software-Defined Networking (SDN) est considéré comme une solution révolutionnaire pour faire face à ces défis.

Les réseaux traditionnels, tels que nous les connaissons historiquement, étaient basés sur des configurations rigides, des protocoles de communication statiques et une configuration manuelle. Ces réseaux nécessitent des interventions réseau IP classiques manuelles fréquentes pour configurer, mettre à jour et résoudre les problèmes. De surcroît, ils présentaient souvent une complexité structurelle élevée et des coûts de maintenance considérables [2].

Cependant, l'arrivée du SDN a marqué le début d'une nouvelle ère dans le domaine des réseaux, en introduisant une approche révolutionnaire, caractérisée par la centralisation du contrôle, la programmabilité du réseau et une gestion plus intelligente. En outre, il propose une interface ouverte pour la configuration et la gestion des réseaux, favorisant ainsi une flexibilité exceptionnelle et une automatisation renforcée, tout en contribuant à réduire les coûts [7].

Dans leur étude, les auteurs [6] ont mis en avant les avantages offerts par le SDN par rapport aux dispositifs de réseau traditionnels. Dans ce cadre, le SDN simplifie la gestion et la configuration du réseau, permet une réponse rapide aux besoins changeants des utilisateurs, améliore les performances du réseau, facilite la mise en œuvre de la Qualité de Service (QoS) pour un flux de données régulé, permet la réutilisation de matériel existant pour des

économies de coûts, offre une vue consolidée du réseau de l'organisation pour une administration efficace, et simplifie la gestion en permettant des modifications de configurations réseau sans perturber l'ensemble du réseau, grâce à un contrôle centralisé. Ainsi, ces avantages permettent à SDN d'offrir une solution attrayante pour la gestion et l'optimisation des réseaux.

Le Tableau 1.1 présente une comparaison entre les caractéristiques du réseau traditionnel et du SDN, mettant en évidence les avantages du SDN en termes de programmabilité, de flexibilité et de coût. Les différences significatives incluent une interface ouverte, une configuration automatique, un support réseau optimisé, une complexité structurelle réduite, une facilité de dépannage et un coût de maintenance inférieur dans le cas du SDN [2].

TABLE 1.1 – Comparaison entre le Réseau Traditionnel et SDN [2]

Caractéristique	**Réseau Traditionnel**	**SDN**
Approche	Traditionnelle	Réseau Virtuel
Contrôle de réseau	Distribué	Centralisé
Programmabilité	Non programmable	Programmable
Interface	Fermée, fournie par le réseau traditionnel	Ouverte, fournie par SDN
Plans de données et de contrôle	Situés sur le même plan	Séparés
Configuration	Manuelle	Automatique
Gestion du trafic	Les paquets circulent dans une seule direction, sans priorisation	Accorde une priorité à certains paquets réseau tout en bloquant d'autres
Coût	Elevé	Réduit
Complexité structurelle	Forte	Faible
Diagnostic et dépannage	Difficile	Facile
Coût de maintenance	Supérieur	Inférieur

1.2.1 Histoire de SDN

Le SDN, malgré son apparence de technologie en plein essor, est en réalité le fruit d'une longue histoire de recherches et d'initiatives dans les milieux universitaires et industriels pour rendre les réseaux plus programmables et flexibles.

Au milieu des années 1990, le concept de réseaux programmables a émergé dans le domaine des technologies de communication. Ce concept innovant a permis des ajustements et des modifications dans la logique de contrôle des dispositifs réseau, offrant ainsi une flexibilité dans l'administration et la configuration des réseaux. Deux acteurs notables ont joué un rôle décisif dans le développement de cette notion : le groupe de travail OpenSig (Open Signaling) et l'initiative Active Networking [8].

L'OpenSig se concentrait particulièrement sur les réseaux ATM (Asynchronous Transfer Mode), qui se distinguaient par une séparation entre le plan de contrôle et le plan de données. Cette séparation permettait d'exploiter un potentiel considérable en termes de personnalisation et d'optimisation du réseau. L'une des caractéristiques les plus marquantes de l'approche

OpenSig était la mise en œuvre d'une interface ouverte pour la signalisation entre les deux plans, offrant ainsi une opportunité d'interopérabilité et de flexibilité sans précédent.

Parallèlement, l'initiative Active Networking a apporté une contribution significative en exposant les ressources des nœuds du réseau via une API réseau. Cette démarche a ouvert de nouvelles perspectives en matière de personnalisation des fonctionnalités et de gestion dynamique des flux de données.

Poularakis et al. [11] proposent une taxonomie de l'histoire du SDN en trois phases, qui illustrent les progrès réalisés dans cette technologie :

— **Phase 1 (milieu des années 1990)**
Le programme de recherche sur les réseaux actifs [12] a proposé deux approches principales pour apporter de la programmabilité dans le réseau : les capsules et les éléments de réseau programmables. Les capsules sont une approche qui consiste à placer du code dans les paquets, appelés "in-band", afin qu'il soit exécuté aux nœuds transitoires du réseau. Quant aux éléments de réseau programmables, ils permettent de placer directement le code à exécuter aux nœuds grâce à des méthodes "out-of-band".

— **Phase 2 (début des années 2000)**
Des efforts clés pour découpler les plans de données et de contrôle du réseau ont été lancés et de nouvelles architectures ont été proposées. Ces efforts étaient principalement motivés par l'augmentation de l'échelle des réseaux et le besoin d'un contrôle flexible basé sur des vues logiques centralisées de l'état du réseau.

— **Phase 3 (2008-présent)**
L'apparition de la spécification OpenFlow [9] a accéléré la tendance aux interfaces ouvertes et à SDN. Un switch OpenFlow utilise un paradigme de correspondance-action. Plusieurs champs de l'en-tête des paquets entrants sont examinés pour des correspondances avec diverses règles prédéfinies. Les correspondances réussies déclenchent des "actions" qui sont des règles de traitement des paquets à utiliser pour le traitement du paquet correspondant.

Dans les premières années 2010, une nouvelle génération de solutions liées à SDN a introduit la notion de programmabilité de la couche de données. De cette manière, les utilisateurs du réseau ont la possibilité d'écrire du code qui détermine les caractéristiques des en-têtes des paquets et établit la logique de correspondance et d'analyse sans être restreints par les spécifications du switch/OpenFlow. Ceci est principalement stimulé par l'utilisation de langages de programmation spécifiquement conçus pour le plan de données, notamment le langage de programmation open source *P4* [1], qui a été introduit en 2013.

L'histoire du SDN démontre que l'idée de rendre le réseau programmable et de séparer les concepts de contrôle et de plan de données existe depuis de nombreuses années. Cependant, ce n'est qu'avec l'avènement d'OpenFlow en 2008 que cette idée a commencé à être réellement mise en pratique dans l'industrie [11].

Aujourd'hui, le SDN est largement utilisé dans une variété d'applications, notamment le cloud computing, les réseaux d'entreprise et les centres de données.

1. `https://opennetworking.org/p4/`

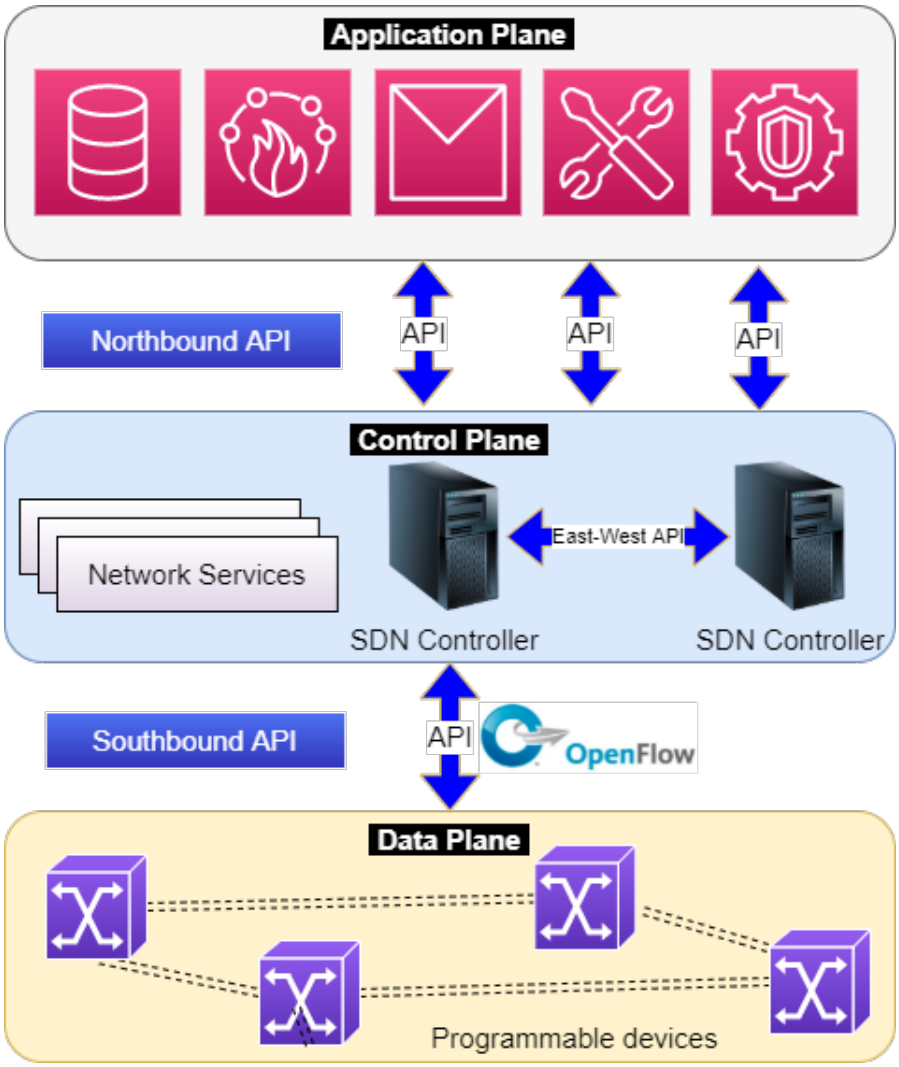


FIGURE 1.1 – Architecture SDN

1.2.2 Concept SDN

Le Réseau Défini par Logiciel (SDN, acronyme de Software Defined Network en anglais) est une approche qui apporte une innovation significative dans le domaine des réseaux informatiques. Elle repose sur la séparation du plan de contrôle et du plan de données.

Comme illustré dans la Figure 1.1, la technologie SDN fait appel à divers termes que nous allons expliciter ci-après :

— **Plan de données**
Egalement connu sous le nom de *plan d'infrastructure*, il constitue la couche la plus basse. Il est composé d'équipements réseau, tels que des routeurs, des pare-feu, des switchs ou bien des systèmes de détection d'intrusions [13]. Ce composant prend les mesures nécessaires pour acheminer les paquets en se basant sur la configuration fournie par le plan de contrôle. Le plan de données contient des informations relatives au réseau et gère le flux de données dans le réseau [14].

— **Plan de contrôle**
Ce composant est souvent considéré comme le cerveau de l'architecture réseau SDN, car il abrite le contrôleur centralisé responsable de la gestion des dispositifs réseau,

appelé *contrôleur SDN*. Il définit et supervise les tables de flux, la logique de transmission au niveau du plan de données, ainsi que leur programmation [13]. Le plan de contrôle interagit en permanence avec le plan de données au moyen d'une interface de programmation (API sud, ou "southbound API") et par l'utilisation du protocole OpenFlow, qui est devenu une norme universelle au sein des réseaux SDN [15]. Le contrôleur SDN gère le plan de contrôle en intégrant divers services et applications.

— **Plan d'application**
Il s'agit de la couche supérieure de l'architecture SDN qui regroupe les différentes applications réseau. Parmi les exemples d'applications, on peut citer les applications d'équilibrage de charge et les pare-feu. Les applications peuvent accéder aux informations d'état du réseau requises par le biais des interfaces Nord des contrôleurs SDN (API nord, ou "northbound API"). La couche d'application utilise ces informations pour personnaliser les politiques réseau et les intentions en fonction des besoins de service [13]. Il est à noter que les informations fournies via les interfaces Nord sont présentées de manière abstraite, dissimulant les détails techniques de l'infrastructure réseau sous-jacente. Cette abstraction réseau facilite la programmation du réseau, simplifie les tâches de contrôle et de gestion [11].

— **Northbound interface**
Dans le cadre des réseaux définis par logiciel (SDN), **l'interface Nord**, par le biais de l'API Nord, établit une connexion entre le plan de contrôle, représenté par le contrôleur SDN, et le plan d'application. Cette liaison permet à l'application de définir les ressources nécessaires au réseau, notamment le stockage et la bande passante, et le réseau alloue ensuite ces ressources en fonction des exigences [14]. Cette approche offre une perspective abstraite du réseau, ce qui favorise l'innovation, l'automatisation et la gestion des réseaux SDN [11].

— **Southbound interface**
L'interface Sud, incluant l'API Sud, établit une connexion entre le plan de contrôle et le plan de données, permettant ainsi aux nœuds du plan de données de transmettre l'état du réseau au plan de contrôle et de recevoir des directives de contrôle sous forme de règles de flux [11]. Cette interface simplifie la gestion et la communication entre les composants clés du réseau, tout en permettant aux dispositifs du plan de données de détecter la topologie du réseau et de suivre les flux de données. OpenFlow est un exemple d'interface Sud largement utilisée, mais il existe également plusieurs autres protocoles d'interface Sud, dont certains sont propriétaires [14].

— **Eastbound/Westbound interfaces**
Les interfaces Est/Ouest permettent à plusieurs contrôleurs SDN de communiquer entre eux. Cela est utile dans les réseaux de grande envergure où un seul contrôleur ne peut pas gérer tous les flux de données. Les interfaces est/ouest permettent également d'accroître la fiabilité du réseau en cas de défaillance d'un contrôleur. Chaque contrôleur assume généralement la responsabilité d'un sous-ensemble de switchs. Les contrôleurs SDN doivent communiquer entre eux pour fournir une vue globale du réseau aux applications de couche supérieure [11].

La technologie SDN intègre également un Système d'Exploitation Réseau (NOS, ou "Networking Operating System") qui facilite la distribution du plan de contrôle entre plusieurs contrôleurs SDN. ONOS [2], un projet open-source soutenu par l'Open Networking Foundation (ONF) [3], figure parmi les logiciels populaires existants qui correspondent à cette

2. `https://opennetworking.org/onos/`
3. `https://opennetworking.org/`

vision du NOS.

L'architecture du SDN, telle qu'elle est représentée dans la Figure 1.1, se caractérise par une segmentation claire entre le plan de contrôle et le plan de données. Cette séparation offre une flexibilité et une agilité inégalées dans la gestion des réseaux, tout en permettant l'intégration de services avancés et de solutions d'optimisation. En séparant le contrôle des flux de données du fonctionnement des équipements réseaux, le SDN ouvre de nouvelles perspectives pour l'administration et la personnalisation des réseaux, en s'adaptant aux exigences évolutives des applications [11].

En résumé, le SDN avec son architecture distincte et son utilisation du protocole OpenFlow, représente une avancée majeure dans le domaine des réseaux. Il offre la possibilité de reconfigurer et de gérer les réseaux de manière plus efficace. De plus, la présence d'un Système d'Exploitation Réseau (NOS), comme ONOS, offre une gestion centralisée et distribuée du contrôle, assurant ainsi l'agilité et la fiabilité des réseaux SDN.

1.2.3 Contrôleurs SDN

Les contrôleurs représentent le noyau fondamental de l'architecture SDN (Software-Defined Networking). Ils constituent un élément essentiel de cet écosystème dynamique et diversifié, où une multitude de solutions sont disponibles pour les experts du domaine. Parmi les principaux contrôleurs SDN *open source* se distinguent des noms tels que Beacon, POX, FloodLight, OpenDayLight (ODL), ONOS, Iris, RYU, Maestro et NOX, reconnus comme des références notables dans le domaine des contrôleurs OpenFlow [16] [17].

Le développement de ces contrôleurs a fait appel à différents langages de programmation, incluant divers choix tels que C, C++, Java et Python. Dans certains cas, l'intégralité de l'architecture du contrôleur a été conçue en utilisant un seul langage, tandis que, pour bon nombre de ces contrôleurs, plusieurs langages ont été harmonieusement combinés au sein de leur noyau et de leurs modules. Cette synergie de langages hétérogènes découle généralement d'une quête d'optimisation de l'allocation de mémoire, de la garantie d'une compatibilité multi-plateforme, et, avant tout, obtenir les meilleures performances dans des situations particulières [17].

Le Tableau 1.2 offre un aperçu des contrôleurs SDN les plus répandus en licence *open source*, qui se basent sur OpenFlow comme protocole de communication pour la gestion des switchs. *NOX* est considéré comme le premier contrôleur SDN. Cependant, il est important de noter que la littérature répertorie de nombreuses autres solutions de contrôleur SDN, y compris celles sous licence propriétaire, ainsi que leurs différentes déclinaisons [18].

Enfin, le choix du contrôleur SDN dépendra des exigences spécifiques du réseau ainsi que des objectifs visés en termes de performances. Ce choix peut être influencé à la fois par la disponibilité des ressources appropriées et par la plateforme d'exploitation adéquate.

1.2.4 Environnement type de SDN

Dans le contexte du domaine des réseaux définis par logiciel (SDN), plusieurs composants clés jouent un rôle essentiel dans la mise en œuvre et la gestion efficace des réseaux.

TABLE 1.2 – Contrôleurs SDN Open Source

Controleur	**Plateforme**	**Année de création**	**Langage de programmation**	**Licence Open Source**	**Éditeur**
Beacon[19]	Windows, MacOS, Linux	2010	Java	GPL 2.0	Stanford University (USA)
Floodlight[20]	Windows, MacOS, Linux	2012	Java	Apache 2.0	Big Switch Networks (USA)
IRIS[21][22]	Linux	2013	Java	Apache 2.0	ETRI (Corée)
Maestro[23]	Windows, MacOS, Linux	2009	Java	LGPL 2.1	Rice University (USA)
NOX[24]	Linux	2008	C++	GPL 3.0	Nicira (USA)
ONOS[25]	Windows, MacOS, Linux	2013	Java	Apache 2.0	ONF (USA)
OpenDaylight[26]	Windows, MacOS, Linux	2013	Java	EPL 1.0	Linux Foundation
POX[27]	Windows, MacOS, Linux	2011	Python	Apache 2.0	Stanford University (USA)
Ryu[28]	MacOS, Linux	2012	Python	Apache 2.0	NTT (Japon)

Parmi ces éléments, nous allons explorer le contrôleur SDN ONOS (Open Network Operating System), la mise en place de réseaux virtuels avec Mininet, la communication Sud avec OpenFlow, et l'utilisation du commutateur logiciel Open vSwitch (OvS) [29]. Ces composants représentent des éléments fondamentaux dans l'écosystème SDN.

Contrôleur SDN : ONOS

ONOS, acronyme de "Open Network Operating System", se positionne en tant que plateforme de contrôle SDN distribuée et ouverte. Ce projet, soutenu par l'Open Networking Foundation, est déployé en tant que logiciel open source sous licence Apache 2.0 [30].

ONOS peut être considéré comme un système d'exploitation réseau ouvert à l'usage de la communauté SDN. Il permet aux applications de gérer les ressources réseau, de créer des itinéraires de transfert de données, de superviser et de programmer les switchs réseau, tout en fournissant une vision globale du réseau. Le principal objectif de ONOS est de répondre aux impératifs des réseaux à grande échelle en matière de performances, de disponibilité et de scalabilité [25].

Virtual network : Mininet

Mininet se présente comme un émulateur de réseau. Il permet de créer un environnement comprenant des machines virtuelles, des switchs, des contrôleurs et des liaisons. Grâce à sa prise en charge complète d'OpenFlow, Mininet offre une grande flexibilité pour la mise en place de routages personnalisés et l'implémentation du Réseau Défini par Logiciel (SDN) [31].

En tant que logiciel de simulation de réseaux virtuels sur des machines physiques, Mininet permet de tester des applications, des protocoles et des configurations réseau. Son adoption généralisée dans les domaines du développement, de la recherche et de l'enseignement en informatique réseau en fait un outil incontournable.

Enfin, Mininet est un logiciel open source sous licence BSD, développé et maintenu par une communauté de bénévoles [31].

SouthBound Communication : OpenFlow

OpenFlow, devenu une norme universellement reconnue au sein des réseaux SDN, est un protocole réseau standard élaboré et publié par l'Open Networking Foundation (ONF) [4]. OpenFlow est un exemple d'interface Sud [9], permettant au contrôleur SDN de dicter au switch la manière dont le trafic réseau doit être géré.

Plus précisément, le contrôleur SDN peut créer des règles de flux spécifiant le cheminement des paquets. Ces règles jouent un rôle essentiel dans la mise en œuvre de divers services réseau, tels que la gestion de la qualité de service (QoS), la sécurité et la gestion du trafic [32].

Software Switch : Open vSwitch (OvS)

Open vSwitch (OVS) est un logiciel de commutation virtuelle multicouche distribué sous la licence open source Apache 2.0. Il occupe une place prépondérante dans le domaine de la virtualisation réseau et est largement adopté dans la technologie SDN. OVS est spécifiquement conçu pour répondre à l'impératif croissant d'une gestion efficace des ressources dans l'environnement virtuel [29].

FIGURE 1.2 – Logo d'Open vSwitch de la *Linux Foundation*

La Figure 1.2 illustre le logo d'*Open vSwitch* tel qu'adopté par la *Linux Foundation* [5].

OVS est conçue pour permettre une automatisation efficace du réseau grâce à des extensions programmables, tout en supportant des interfaces et protocoles de gestion standard tels que NetFlow, sFlow, SPAN, RSPAN, CLI, LACP et 802.1ag [33].

4. `https://opennetworking.org/`
5. `https://www.openvswitch.org/`

Enfin, OVS est une solution de commutation virtuelle flexible et puissante, adaptée à une large gamme de scénarios de réseau, de la virtualisation à la surveillance avancée du trafic.

1.3 Virtualisation des fonctions réseau (NFV)

À la fin de l'année 2012, plus de vingt des plus importants fournisseurs de services de télécommunications au niveau mondial, parmi lesquels AT&T, BT, Deutsche Telekom et NTT, se sont réunis au sein de l'Industry Specification Group (ISG) de l'European Telecommunications Standards Institute (ETSI) dans le but de définir la Virtualisation des Fonctions Réseau (Network Functions Virtualization, ou NFV en anglais).

Depuis lors, l'initiative NFV a suscité un intérêt considérable, avec une participation accrue d'opérateurs de réseaux et de fournisseurs de technologies issus de l'ensemble de l'industrie des télécommunications [34].

L'objectif principal de l'initiative NFV réside dans la résolution des enjeux opérationnels et financiers inhérents à la gestion des dispositifs exclusifs et propriétaires qui sont actuellement en service au sein des infrastructures de télécommunication.

Cette démarche vise à permettre aux opérateurs d'accroître considérablement leur flexibilité opérationnelle, d'accélérer le déploiement de nouvelles prestations, tout en réalisant des économies à la fois sur les dépenses d'exploitation (OpEx) et les dépenses d'investissement (CapEx) [34].

NFV est une approche révolutionnaire dans le domaine des réseaux de communication, visant à virtualiser les fonctions réseau traditionnelles en dissociant le logiciel de l'infrastructure matérielle sous-jacente. Cela permet d'exécuter ces fonctions réseau dans des environnements virtualisés sur des serveurs standard de l'industrie, offrant ainsi des avantages tels que la réduction des coûts d'équipement et une plus grande flexibilité dans le déploiement et l'exploitation des services réseau [35].

En somme, le NFV vise à simplifier les réseaux en remplaçant les équipements matériels dédiés par des logiciels virtualisés sur des serveurs génériques. Les principaux avantages du NFV sont [10] :

Déploiement rapide des services : Il simplifie l'ajout de nouveaux services en utilisant des ressources virtualisées et des logiciels, accélérant ainsi le déploiement et la mise à jour des services.

Réduction des coûts de construction du réseau : En regroupant les éléments du réseau dans des dispositifs génériques et en optimisant l'utilisation des ressources, le NFV diminue les coûts de déploiement et d'exploitation du réseau.

Efficacité accrue de l'exploitation et de la maintenance du réseau : Grâce à une gestion automatisée et centralisée, le NFV améliore l'efficacité opérationnelle et réduit les coûts de maintenance.

Création d'un écosystème ouvert : Contrairement aux réseaux télécoms traditionnels fermés, le NFV favorise l'ouverture du réseau aux développeurs tiers en utilisant des standards matériels et logiciels, encourageant ainsi l'innovation.

1.3.1 Fonctions réseau virtualisées (VNF)

Dans le contexte de NFV, les fonctions réseau traditionnelles, telles que les pare-feu, les IDS (Intrusion Detection Systems) et les proxys, sont transformées en Virtualized Network Functions (VNF), ou fonctions réseau virtualisées. Les VNF représentent les composantes essentielles de NFV, car ce sont les éléments spécifiques qui sont exécutés dans des environnements virtualisés sur des serveurs standard de l'industrie [35].

Les VNF permettent de réaliser la vision de NFV en offrant une plus grande flexibilité lors du déploiement des services réseau, une réduction des dépenses en équipements et une meilleure utilisation des ressources réseau.

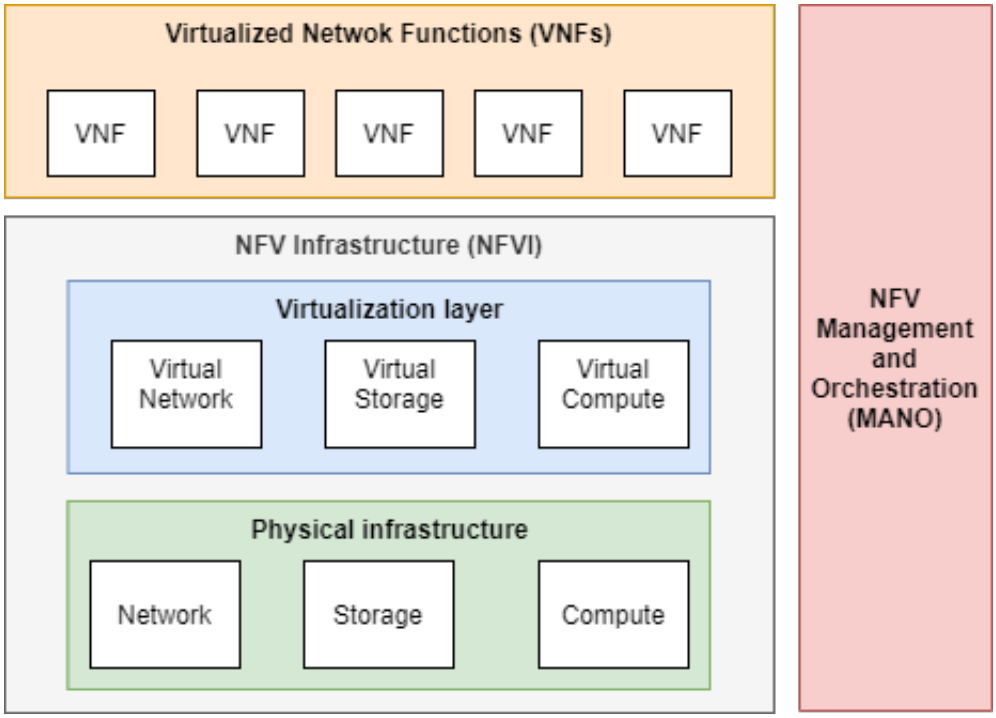


FIGURE 1.3 – Architecture de la NFV

La Figure 1.3 présente une vue d'ensemble de l'architecture de la NFV, telle que définie par l'organisme de normalisation ETSI (European Telecommunications Standards Institute) [36]. Cette représentation met en lumière trois domaines fondamentaux au sein de la NFV :

(i) Les Fonctions Réseau Virtualisées (VNF) incarnent la concrétisation logicielle des fonctions réseau pouvant être déployées et exécutées au sein du NFVI.

(ii) L'Infrastructure de Virtualisation des Fonctions Réseau (NFVI) couvre une variété de ressources physiques et leur capacité à être virtualisées. Le NFVI est chargé d'assurer le déploiement et l'exécution des VNFs.

(iii) La Gestion et l'Orchestration de la Virtualisation des Fonctions Réseau (MANO) englobent l'orchestration et la gestion du cycle de vie des ressources physiques et/ou logicielles nécessaires à la virtualisation de l'infrastructure, ainsi que la gestion du cycle de vie des VNFs.

En résumé, NFV représente l'approche générale qui permet de virtualiser les fonctions réseau, tandis que VNF est le résultat concret de cette virtualisation, à savoir les fonctions

réseau elles-mêmes qui ont été virtualisées. NFV permet la création et le déploiement des VNF, contribuant ainsi à améliorer l'efficacité et la flexibilité des réseaux.

1.3.2 Chaînage de fonctions service (SFC)

La Chaîne de Fonctions de Service, ou Service Function Chain (SFC) en anglais, a été définie dans le RFC7665 [37]. Elle représente un ensemble structuré de fonctions de service abstraites, accompagné de contraintes d'ordonnancement qui doivent être appliquées aux paquets, aux trames et/ou aux flux sélectionnés suite à une opération de classification.

Le chaînage fonctionnel est une méthode qui consiste à solliciter séquentiellement des fonctions élémentaires telles que la NAT (Network Address Translation), un pare-feu ou un IPS (Intrusion Prevention Systems), au sein d'une séquence prédéfinie. Cette approche permet de créer des flux de traitement personnalisés, où chaque fonction de service est activée dans un ordre spécifique pour répondre de manière précise aux exigences.

L'ordre implicite n'est pas rigoureusement linéaire, en raison d'une architecture qui autorise les SFC à se diviser en plusieurs branches, tout en préservant la souplesse nécessaire pour que l'ordre d'application des fonctions de service puisse varier dans des circonstances spécifiques. Cette flexibilité permet d'adapter la configuration des fonctions de service en fonction des besoins et des exigences particulières du réseau.

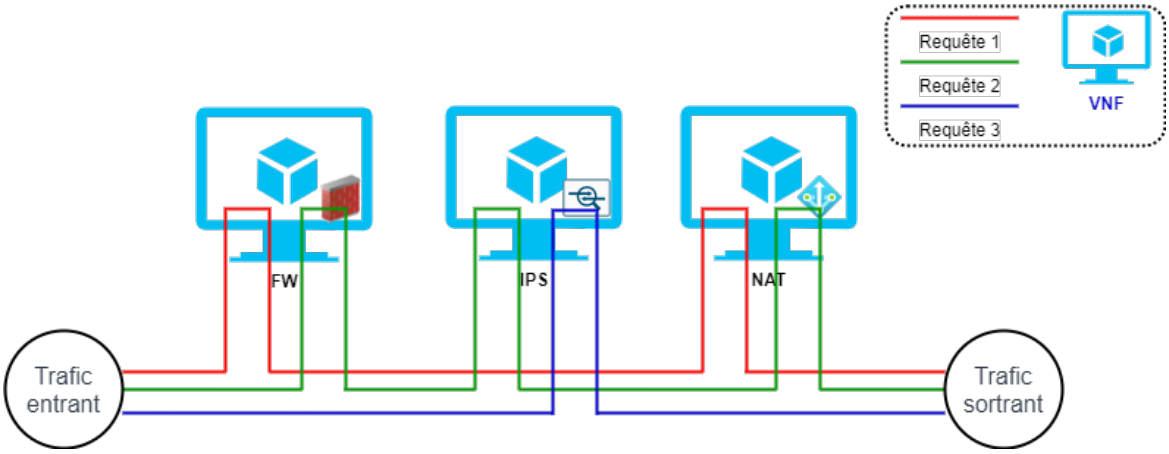


FIGURE 1.4 – Exemple d'utilisation du SFC

La Figure 1.4 présente un exemple d'utilisation du SFC au sein d'un réseau de télécommunication, incorporant des fonctions telles que le système de prévention des intrusions (IPS), le pare-feu (FW) et la translation d'adresse réseau (NAT). Ces fonctions jouent un rôle crucial dans la sécurisation, la gestion du trafic et l'optimisation des performances au sein du réseau, garantissant ainsi une expérience de communication fiable et de haute qualité pour les utilisateurs finaux.

Comme illustré, la requête 1 a pour objectif d'accéder aux services web depuis le serveur, nécessitant uniquement un pare-feu de base et une translation d'adresse réseau (NAT). À cet effet, le contrôleur SDN met en place une chaîne de fonctions de service (SFC) spécifique pour la requête 1 : [FW, NAT], où le trafic est dirigé uniquement à travers le pare-feu et la NAT.

Quant à la requête 2, elle concerne l'accès à des données sensibles à partir du serveur. Dans ce contexte, le contrôleur SDN élabore une chaîne de SFC distincte pour la requête 2 : [FW, IPS et NAT], permettant au trafic de passer par l'ensemble des fonctions réseau nécessaires à cette demande particulière.

Enfin, la requête 3 parcourt uniquement l'IPS en utilisant la chaîne : [IPS] [38].

1.3.3 Synthèse

La virtualisation des fonctions réseau (NFV - Network Function Virtualization) représente un modèle architectural novateur et évolutif dans le domaine des réseaux informatiques. Elle tire parti de la technologie de virtualisation pour restructurer l'infrastructure des réseaux, en séparant les fonctionnalités des nœuds réseau de leur infrastructure matérielle. La VNF permet une plus grande flexibilité et agilité dans la gestion des réseaux.

Ce concept novateur donne naissance aux fonctions réseau virtuelles (VNF - Virtual Network Functions), des services réseau hébergés sur des machines virtuelles (VM). Ces VNFs peuvent inclure des pare-feux, des routeurs, des IDS et sont déployés de manière dynamique selon les besoins spécifiques du réseau.

Cette flexibilité permet aux opérateurs de réseaux de configurer facilement les fonctions réseau en réponse aux évolutions du trafic et des besoins des services.

Une séquence organisée de plusieurs VNFs, pour acheminer et gérer le flux de données, est connue sous le nom de chaîne de fonctions de service (SFC - Service Function Chain). Cette chaîne de fonctions de service permet aux opérateurs réseau de définir des politiques de routage personnalisées, optimisant ainsi le parcours du trafic selon des critères spécifiques tels que la qualité de service ou la sécurité.

Enfin, le déploiement de SFC à l'aide du SDN et du NFV favorisera une implémentation plus économique et flexible des réseaux 5G de nouvelle génération [38].

1.4 Techniques d'optimisation

Dans cette section, nous nous intéresserons sur les différentes techniques d'optimisation existantes dans la littérature spécialisée.

Plus précisément, nous nous intéresserons aux approches heuristiques ainsi qu'à la programmation linéaire en nombres entiers (*Integer Linear Programming*, en anglais) dans le cadre de nos travaux de thèse.

1.4.1 Problème d'optimisation

Un problème d'optimisation est une quête visant à trouver le maximum ou le minimum, également appelé optimum, d'une fonction donnée. Cela implique l'exploration parmi un ensemble de solutions possibles, dénommé espace de décision ou espace de recherche, afin de déterminer la ou les solutions qui minimisent ou maximisent une fonction mesurant la qualité de ces solutions. Cette fonction est communément appelée fonction objectif ou fonction coût [39].

Soit X un ensemble de solutions possibles et f la fonction objectif à minimiser (ou à maximiser) qui va de l'ensemble X vers l'ensemble des nombres réels $\mathbb{R}$, tel que :

$$f : X \to \mathbb{R}$$

Le problème consiste donc à trouver la meilleure solution possible, c'est-à-dire l'optimum $x^* \in X$, avec $f(x^*)$ minimal (ou maximal).

1.4.2 Classification des techniques d'optimisation

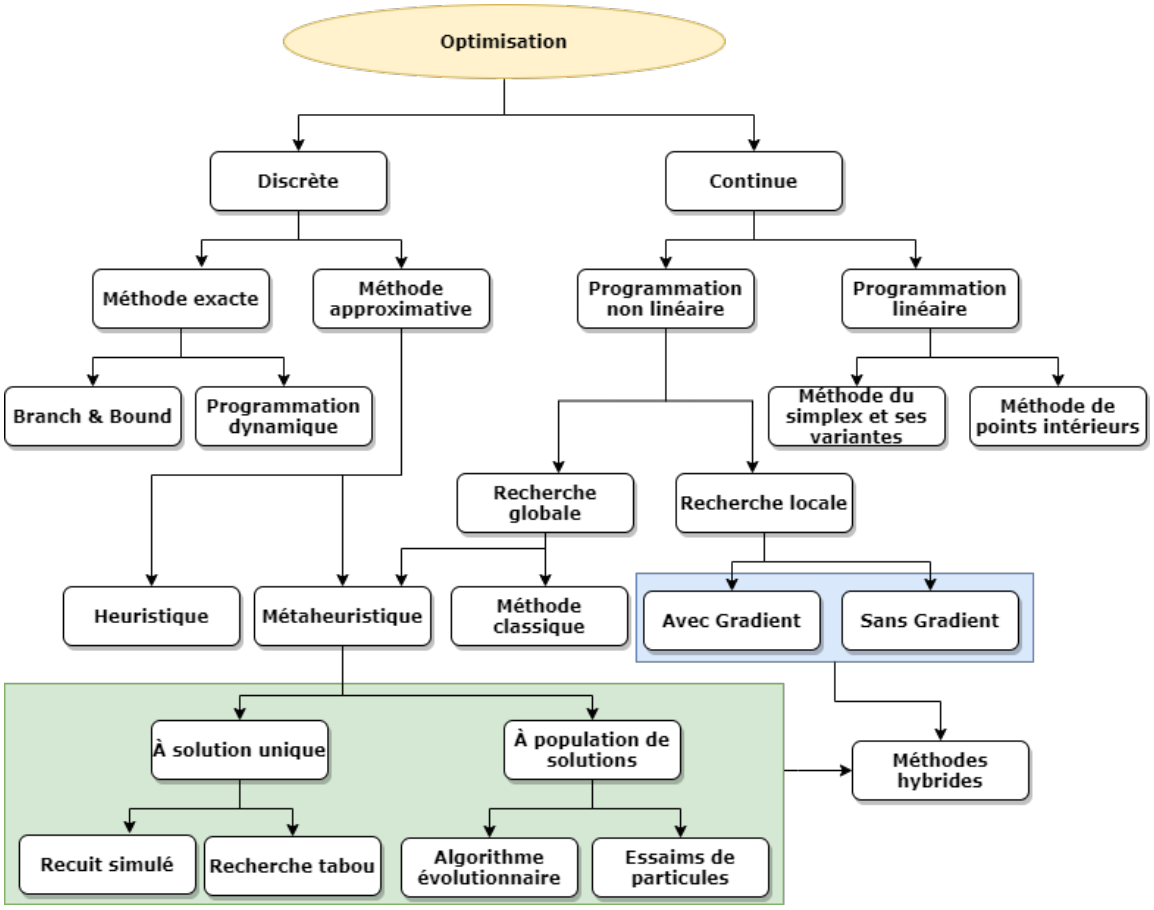


FIGURE 1.5 – Classification des techniques d'optimisation

Lorsqu'on aborde la résolution d'un problème d'optimisation, cela nécessite une exploration variée des méthodes disponibles. La Figure 1.5 offre une représentation visuelle et une classification de ces différentes méthodes, permettant ainsi de mieux appréhender la diversité des approches utilisées dans ce domaine [39][40].

Les problèmes d'optimisation, de par leur nature même, requièrent une analyse méthodique et une évaluation rigoureuse des outils disponibles pour parvenir à des solutions efficaces.

La pluralité des méthodes présentées dans la figure englobe une gamme étendue de techniques, chacune ayant ses propres avantages, limites et applications spécifiques.

Dans la Figure 1.5, on distingue initialement deux catégories qui se démarquent : l'optimisation discrète (aussi appelée combinatoire) et l'optimisation continue. Cette distinction initiale repose sur la nature des "espaces" où les variables de décision évoluent, marquant ainsi la séparation entre le domaine discret et continu, une distinction fondamentale en mathématiques, impactant considérablement les méthodes applicables.

1.4.3 Optimisation discrète

Les problèmes d'**optimisation discrète** (ou combinatoire) de taille raisonnable peuvent être résolus de manière exacte par des méthodes précises qui parcourent méthodiquement l'espace des combinaisons pour trouver la solution optimale.

Cependant, la plupart du temps, le nombre de combinaisons possible des valeurs des variables est immense, rendant l'énumération exhaustive inefficace comme méthode de recherche de solution.

Pour contenir cette explosion combinatoire, ces approches structurent l'espace des combinaisons en arbre, recourent à des techniques d'élagage pour réduire cet espace, et utilisent des heuristiques pour déterminer l'ordre d'exploration.

Malgré cela, les techniques de filtrage et les heuristiques ne parviennent pas toujours à réduire suffisamment la complexité combinatoire, rendant certaines instances de problèmes impossibles à résoudre en un temps acceptable par ces méthodes exhaustives. Parmi les **méthodes exactes**, on retrouve les méthodes de séparation et évaluation, telles que les méthodes de **Branch & Bound**, ainsi que la **programmation dynamique**.

Lorsque les capacités de calcul sont restreintes ou lorsque des défis complexes d'une ampleur considérable se présentent, il est fréquent d'opter pour des **approches approximatives** afin d'obtenir une solution de qualité raisonnable. Dans ces situations, le choix se fait parfois entre une **heuristique spécialisée** dédiée au problème spécifique ou une **métaheuristique** [39].

L'utilisation d'**heuristiques** s'avère nécessaire lorsque la résolution exacte des problèmes combinatoires de grande ampleur est souvent impossible. Les heuristiques sont des méthodes empiriques spécifiquement adaptées à un problème donné. Elles reposent généralement sur des approches « *gloutonnes*, choisissant à chaque étape la solution immédiate jugée optimale. Bien que simples et rapides car elles ne nécessitent pas d'itérations, ces méthodes ne garantissent pas la qualité finale du résultat[40].

Cependant, une **métaheuristique** représente une approche empirique globale de recherche qui peut être appliquée à divers problèmes d'optimisation[40].

Les métaheuristiques peuvent être classées en deux grandes catégories en fonction de la manière dont elles explorent l'espace de recherche :

Les métaheuristiques à solution unique

Egalement connues sous le nom de méthodes de trajectoire, elle débutent avec une solution initiale unique et s'éloignent graduellement de celle-ci en parcourant une trajectoire dans l'espace de recherche. Cette catégorie englobe principalement la **méthode de descente**, le **recuit simulé**, la **recherche tabou**, la méthode **GRASP**, la **recherche à voisinage variable**, la **recherche locale itérée** ainsi que leurs déclinaisons.

Les métaheuristiques à population de solutions

Ces métaheuristiques progressent au fil des itérations en améliorant un ensemble de solutions. Cette catégorie comprend les algorithmes évolutionnaires, une famille d'algorithmes découlant de la théorie de l'évolution de Charles Darwin (1859), ainsi que les algorithmes d'optimisation par essaims de particules. Ces derniers, tout comme les algorithmes évolutionnaires, tirent leur inspiration d'analogies avec des phénomènes biologiques naturels.

1.4.4 Optimisation continue

Dans le domaine de l'**optimisation continue** (appelée aussi optimisation paramétrique), une première distinction se fait entre les cas linéaires (relevant notamment de la **programmation linéaire**) et les cas **non linéaires**, qui englobent les défis de l'optimisation complexe.

Cependant, nous retrouvons également la **programmation linéaire mixte** qui inclut les problèmes linéaires impliquant des variables, dont certaines sont continues tandis que d'autres sont entières ou binaires [40].

Programmation linéaire

La programmation linéaire constitue une méthode essentielle pour résoudre des problèmes d'optimisation, s'étant érigée comme l'une des approches les plus répandues en ce domaine.

Elle est couramment employée pour résoudre des problèmes d'optimisation en recherche opérationnelle, couvrant divers domaines tels que le routage des flux, la planification, l'allocation et la gestion des ressources, entre autres.

Un programme linéaire se compose de trois éléments fondamentaux [41] :

1. Une fonction objectif : visant à minimiser ou maximiser un coût.
2. Un ensemble de variables de décision : chaque variable est dotée d'un coefficient dans la fonction objectif, exprimé en coût.
3. Un ensemble de contraintes linéaires, exprimées sous forme d'égalités et/ou d'inégalités : chaque contrainte restreint les valeurs des variables.

Dans ce qui suit, nous présentons un exemple d'application de la programmation linéaire en nombres entiers pour modéliser le problème du plus court chemin entre deux points, en tenant compte de contraintes de capacité inhérentes [41].

Cette démarche, bien que la résolution par la programmation linéaire en nombres entier puisse ne pas être la plus efficiente, offre une approche méthodologique permettant de saisir les subtilités de la modélisation des contraintes.

Considérons le graphe G défini comme $G = (V, E, W)$, où il représente une requête émanant d'une source $s \in V$ et devant être acheminée vers une destination d (également $d \in V$). Cette requête entraîne une utilisation de la bande passante, représentée par la quantité $w'_{s,d}$. Ainsi, la requête est modélisée de la manière suivante : $(s, d, w'_{s,d})$.

Les paramètres utilisés dans cette modélisation sont présentés dans le Tableau 1.3.

TABLE 1.3 – Paramètres de modélisation du problème du plus court chemin.

Variable	Description
V	Un ensemble de nœuds.
E	Un ensemble de liens.
W	La bande passante de communication entre les différents nœuds dans V.
$G(V, E, W)$	Un graphe qui représente la topologie du réseau.
$w_{i,j}$	La capacité de la bande passante du lien $(i, j) \in E$, où $w_{i,j} \in W$ et $i, j \in V$.
$w'_{s,d}$	La bande passante requise pour la liaison entre la source s et la destination d, où $s, d \in V$.
$(s, d, w'_{s,d})$	Modélisation de la requête, où $s, d \in V$.

La **variable** de décision $x_{i,j}$ est représentée par l'état de l'utilisation du lien $(i, j) \in E$ entre les nœuds i et j, où $x_{i,j} \in \{0, 1\}$. Les valeurs de x sont soit 0, soit 1, comme mentionné dans l'Equation 1.1.

$$x_{i,j} = \begin{cases} 1 & \text{Si le lien } (i, j) \text{ est utilisé} \\ 0 & \text{Sinon} \end{cases} \tag{1.1}$$

Notre objectif principal réside dans la détermination du plus court chemin entre les noeuds i et j, tout en garantissant le respect de la capacité $w_{i,j}$ de chaque liaison $(i, j) \in E$.

A cet effet, la **fonction objectif** consiste à minimiser le nombre de liens utilisés et est formalisée dans l'Equation 1.2

$$min \sum_{(i,j)\in E} x_{i,j} \tag{1.2}$$

Cependant, les contraintes linéaires sont exprimées sous la forme de contraintes de conservation des flux, contraignant ainsi le déplacement du flux du nœud s vers le nœud d. L'Equation 1.3 expose ces contraintes de conservation des flux pour chaque nœud $i \in V$.

$$\sum_{(i,j)\in E} x_{i,j} - \sum_{(j,i)\in E} x_{j,i} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = d \\ 0 & \text{else} \end{cases} \tag{1.3}$$

Pour ce qui est des contraintes liées à la capacité des liens, l'Équation 1.4 définie les limitations imposées à chaque lien $(i, j) \in E$. Ainsi, elle exclut l'utilisation de tout lien dont la capacité n'est pas suffisante pour assurer le transit du flux de la requête.

$$w'_{i,j} \cdot x_{i,j} \leq w_{i,j} \tag{1.4}$$

Programmation non linéaire

La programmation non linéaire vise à optimiser des fonctions objectives non linéaires tout en respectant des contraintes. Elle est appliquée lorsque les relations entre les variables ne suivent pas une forme linéaire [40].

Dans la forme de minimisation de la fonction objective $f(x)$, le programme non linéaire est formulé de façon générale dans l'Equation 1.5, où $x \in \mathbb{R}^n$.

$$\underset{x \in \mathbb{R}^n}{min} f(x) \text{ sous réserve } \begin{cases} G_i(x) \leq 0, i \in \{1, 2, \ldots, p\} \\ H_j(x) = 0, j \in \{1, 2, \ldots, q\} \end{cases} \tag{1.5}$$

Avec n, p et q des entiers positifs, l'Équation 1.5 représente un problème d'optimisation à n variables x, soumis à p contraintes d'inégalité G et q contraintes d'égalité H.

Programmation linéaire mixte

La programmation linéaire mixte s'applique aux problèmes linéaires complexes où la nature des variables varie entre réelles, entières ou binaires. Cette méthode trouve une utilité significative dans de nombreux problèmes concrets qui peuvent être formulés sous la forme d'un problème linéaire mixte [40].

En ce qui concerne les méthodes de résolution, la programmation linéaire mixte se base principalement sur l'algorithme du simplexe. Cet algorithme, fondamental en optimisation linéaire, examine itérativement les solutions envisageables pour converger vers une solution optimale.

La formulation standard d'un problème linéaire impliquant des variables mixtes est donné dans l'Équation 1.6.

$$\underset{x}{min} f^T x \text{ sous réserve } \begin{cases} A \cdot x = b, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m \\ x_i \geq 0, i \in I \cup R \end{cases} \tag{1.6}$$

La fonction objectif est définie par $f^T x$, avec f représentant un vecteur de constantes, et x désignant le vecteur des variables inconnues. Il est à noter que x appartient à l'espace vectoriel $\mathbb{R}^n$, où n représente le nombre de variables du problème.

Les indices I sont associés aux variables entières, tandis que les indices R désignent les variables continues (ou réelles) positives.

1.5 Optimisation du routage multi-chemins dans les réseaux SDN

La mise en oeuvre efficace de stratégies de routage multi-chemin dans les réseaux SDN a suscité un intérêt croissant dans la littérature, en vue d'améliorer divers aspects des performances du réseau, tels que la qualité de service (QoS), la fiabilité et l'efficacité opérationnelle.

Dans cette section, nous examinons plusieurs recherches importantes qui se sont intéressées à cette problématique.

Egilmez et al. [42] ont présenté une solution qui exploite *OpenFlow* pour effectuer un routage QoS optimal pour la diffusion en continu de vidéos. Cette solution utilise plusieurs contrôleurs SDN distribués pour optimiser la classification du trafic et l'allocation des chemins, améliorant ainsi le débit du réseau.

Sharma et al. [43] ont proposé une solution visant à garantir la QoS et le SLA [6] (*contrôleurs dynamiques (DCAP)*, en anglais), en utilisant des interfaces et des protocoles SDN, notamment *OpenFlow* et *Open vSwitch Database (OVSDB)* [7]. Le framework proposé est mis en œuvre dans des scénarios de systèmes autonomes (*autonomous system*, en anglais)) sur le contrôleur SDN *Floodlight* au sein du banc d'essai *OpenFlow* européen *Ofelia*.

Pour réduire les temps de réponse et les coûts de maintenance dans les centres de données, Wang et al. [45] ont formulé le problème d'attribution des contrôleurs dynamiques (*dynamic controller assignment problem (DCAP)*, en anglais) en tant qu'optimisation en ligne.

Tomovic et Radusinovic dans [46] ont élaboré un framework de contrôle SDN visant à résoudre les problèmes de congestion réseau afin d'améliorer tant le débit que la qualité de service, au sein des réseaux des fournisseurs de services Internet (*ISP*). La solution repose sur un modèle d'optimisation bi-objectif qui harmonise l'optimisation du routage et la minimisation du nombre de mises à jour de la table de flux.

Par ailleurs, Xiaolong et al. [47] ont introduit *MTSS* (*multi-path traffic scheduling mechanism*), un mécanisme de planification du trafic multi-chemins basé sur SDN pour résoudre le problème de la congestion du réseau et améliorer la qualité de service dans les centres de données en nuage (Cloud).

Celenlioglu et Mantar [48] ont proposé un modèle de routage et de gestion des ressources pour les réseaux intra-domaines basés sur SDN, en tenant compte de multi-chemins préétablis avec une réservation de ressources pour améliorer l'évolutivité du routage et réduire le temps d'admission, assurant ainsi la qualité de service souhaitée.

Yan et al. [49] ont proposé *HiQoS*, une solution QoS multi-chemins basé sur le contrôleur SDN open-source *Floodlight*, pour réduire le délai et augmenter le débit, assurant ainsi les performances de la QoS souhaitée. En même temps, il s'appuie sur une version modifiée de l'algorithme de Dijkstra. *HiQoS* met en œuvre une transmission multi-chemins pour résoudre rapidement le problème des chemins défaillants, en considérant le chemin optimal pour chaque flux entre les nœuds en surveillant l'état du réseau en temps réel.

Sahhaf et al. [50] ont également introduit une approche similaire qui trouve un schéma de routage multi-chemins adaptatif basé sur des intervalles de temps, pour sélectionner les chemins dynamiquement avec une bande passante maximale et une disponibilité élevée.

Hussain et al. [51] ont proposé un protocole de planification multi-chemins dynamique (DMSP) en tant qu'algorithme de planification dans l'environnement SDN pour faciliter la gestion des liens dans un réseau de centre de données en utilisant le contrôleur *Floodlight*.

6. L'accord de niveau de service (SLA) définit le niveau de service convenu entre un fournisseur de services et un client [44].

7. Base de données utilisée pour la configuration des instances OVS (RFC 7047).

Basit et al. [52] exploitent une coordination intercouches entre les Fournisseurs de Services Internet (FSI), et utilisent une stratégie de transfert multi-chemins basée sur SDN pour atteindre une allocation optimale des ressources et augmenter la fiabilité du réseau.

Dans le domaine de la grille informatique (grid computing), Huang et al. [53, 54] ont utilisé l'Algorithme de Dijkstra pour proposer le protocole *GridFTP* qui vise à augmenter le taux de transfert de données.

Étant donné la séparation du plan de données et du plan de contrôle dans l'architecture SDN, Fu et Wu [55] ont présenté les avantages de l'utilisation d'un algorithme de routage multi-chemins disjoint pour l'équilibrage de charge en considérant différents modèles de graphes réseau.

Dans les systèmes de stockage distribués (DSS), Guillen et al. [56] ont proposé une approche hybride qui exploite à la fois les capacités de routage multi-chemins basé sur SDN, pour offrir un débit réseau élevé et une utilisation équilibrée des ressources.

Guan et al. [57] ont introduit un framework multipath superposé, appelé réseau superposé de services définis par logiciel (SDSON), pour orchestrer les nœuds superposés et fournir une transmission garantie de la qualité de service (QoS). Le SDSON a été conçu pour faire face à la mobilité fréquente des utilisateurs.

Hiryanto et al. [58] ont proposé un programme linéaire mixte en nombres entiers ainsi qu'un algorithme heuristique appelé *M-GMSU*, pour maximiser les économies d'énergie. Cela se réalise en mettant à niveau un réseau existant à travers une implémentation de routage multi-chemins basé sur la technologie SDN.

D'autre part, Mahmoudi et al. [59] ont introduit la méthode *SDN-DVFS* (*dynamic voltage frequency scaling*) pour équilibrer équitablement la charge du trafic sur plusieurs serveurs en exploitant le SDN. La solution proposée prend en compte la surcharge de chaque machine virtuelle (VM), l'efficacité de la machine hôte et la charge créée par chaque utilisateur.

Cependant, contrairement aux résultats obtenus dans notre travail de thèse, ces travaux sont insuffisants pour parvenir à une allocation optimale des ressources, dans un environnement multi-chemins et dynamique qui implique la mobilité des utilisateurs.

Dwarakanathan et al. [60] ont proposé un module conçu pour garantir une haute disponibilité de la Qualité de Service (QoS), prenant en considération les changements fréquents au sein du réseau. La solution proposée vise à garantir la bande passante réseau tout en réduisant les coûts.

En revanche, l'objectif de notre travail de thèse est de fournir une configuration optimale en tirant parti à la fois de l'optimisation du réseau et des techniques d'apprentissage automatique.

Yoon et Kamal [61] ont proposé une solution pour minimiser la consommation d'énergie dans les centres de données basés sur SDN en tenant compte de l'état des hôtes et des commutateurs, garantissant ainsi la QoS souhaitée. Par contre, nous visons, dans le cadre de la thèse, à fournir une configuration optimale permettant de réaliser des économies de coûts et d'énergie, tout en tenant également compte de la fiabilité du réseau.

Pour atteindre une QoS optimale dans les centres de données SDN, Tariq et Bassiouni [62] ont proposé un algorithme basé sur l'algorithme de Dijkstra, qui permet de contrôler la réservation de la bande passante dans Multipath TCP sur les réseaux OBS (*Optical Burst Switching*).

En tirant parti des architectures SDN et NFV, Wang et al. [63] ont introduit un schéma de virtualisation de réseau multipath qui permet la planification des ressources réseau en sélectionnant et en répartissant les flux sur le réseau multi-chemins.

Pour récapituler les travaux de recherche mentionnés dans cette section, une comparaison a été réalisée, comme présenté dans le Tableau 1.4. Ce tableau synthétise les objectifs principaux, les méthodes employées, ainsi que les technologies et protocoles spécifiques associés à chaque étude.

En comparaison avec les travaux mentionnés précédemment, l'objectif de notre travail, dans le cadre de la thèse, est de proposer un nouveau framework agile qui tire parti à la fois de l'optimisation du réseau et des techniques d'apprentissage automatique (*Machine Learning*, en anglais), pour configurer les réseaux basés sur SDN afin de réduire les coûts, tout en garantissant la fiabilité du réseau et la qualité de service (QoS).

En outre, la conception principale de notre solution vise à fournir un paradigme de configuration réseau rapide capable de capturer les changements du réseau et de fournir une configuration optimale.

1.6 Conclusion

En conclusion, le chapitre sur les réseaux définis par logiciel (SDN) et l'optimisation du routage multi-chemins offre une vision des évolutions majeures dans le domaine des réseaux informatiques.

L'émergence du SDN a marqué une transition significative des architectures traditionnelles vers une approche plus flexible et programmable [6], répondant ainsi aux exigences croissantes des applications et des utilisateurs.

La centralisation du contrôle, la programmabilité du réseau et la gestion intelligente ont propulsé le SDN comme une solution révolutionnaire, offrant des avantages tels que la simplification de la gestion, la réponse rapide aux besoins changeants, l'amélioration des performances et la réduction des coûts [7].

L'histoire du SDN remonte aux années 1990, avec des initiatives telles que le groupe OpenSig et l'Active Networking [8], établissant les bases conceptuelles de la séparation entre le plan de contrôle et le plan de données. Le développement ultérieur a conduit à l'avènement d'OpenFlow en 2008, marquant le début de la mise en pratique de la programmabilité des réseaux [9].

La virtualisation des fonctions réseau (NFV) est également abordée, soulignant l'importance de la dissociation du logiciel et de l'infrastructure matérielle pour optimiser la gestion des fonctions réseau[10]. Les fonctions réseau virtualisées (VNF) et la Chaîne de Fonctions de Service (SFC) élargissent les horizons de la virtualisation, offrant flexibilité et économies de coûts.

La section sur les techniques d'optimisation explore les approches heuristiques et la programmation linéaire mixte, mettant en lumière les défis des problèmes d'optimisation dans le contexte des réseaux. Cette section classe les méthodes en fonction de leur applicabilité aux problèmes discrets ou continus.

Enfin, la section sur l'optimisation du routage multi-chemins dans les réseaux SDN met en évidence des travaux de recherche pertinents, soulignant l'importance de cette approche pour améliorer la qualité de service, la fiabilité et l'efficacité opérationnelle des réseaux.

Ces travaux de recherche servent de fondement à notre proposition de framework agile intégrant optimisation réseau et techniques d'apprentissage automatique pour configurer les réseaux SDN de manière efficiente.

En somme, ce chapitre présente les principes essentiels pour la compréhension des concepts fondamentaux liés aux réseaux définis par logiciel (SDN) et à l'optimisation du routage multi-chemins.

TABLE 1.4 – Travaux de recherche sur le routage multi-chemins dans les réseaux SDN.

Auteurs	**Objectif principal**	**Méthode/Technologie**
Egilmez et al. [42]	Routage QoS pour vidéos	OpenFlow, SDN distribué
Sharma et al. [43]	QoS et SLA avec SDN	OpenFlow, OVSDB , Floodlight
Wang et al. [45]	Attribution contrôleurs dynamiques	Optimisation en ligne DCAP
Tomovic et Radusinovic [46]	Résoudre les problèmes de congestion	Modèle d'optimisation bi-objectif
Xiaolong et al. [47]	Résoudre congestion réseau	Multi-trajet MTSS
Celenlioglu et Mantar [48]	Routage intra-domaines SDN	Multi-chemins préétablis
Yan et al. [49]	Assurer performances de la QoS souhaitée	Floodlight, Dijkstra modifié
Sahhaf et al. [50]	Routage multi-chemins	Adaptatif, basé sur temps
Hussain et al. [51]	Planification multi-chemins dynamique	DMSP, Floodlight
Basit et al. [52]	Allocation optimale des ressources	Coordination intercouches FSI
Huang et al. [53, 54]	Augmenter le taux de transfert de données	Algorithme Dijkstra, Grid computing
Fu et Wu [55]	Equilibrage de charge réseau	Algorithme de routage multi-chemins disjoint
Guillen et al. [56]	Augmentation du débit et équilibrage des ressources	Approche hybride pour systèmes de stockage distribués
Guan et al. [57]	Garantir QoS	Framework SDSON, multi-path superposé
Hiryanto et al. [58]	Économies d'énergie	Programme linéaire mixte en nombres entiers, algorithme heuristique M-GMSU
Mahmoudi et al. [59]	Equilibrer la charge du trafic	SDN-DVFS
Dwarakanathan et al. [60]	Module haute disponibilité QoS	Changements fréquents
Yoon et Kamal [61]	Minimisation de la consommation d'énergie	État hôtes et commutateurs
Tariq et Bassiouni [62]	QoS optimale dans les centres de données	Algorithme Dijkstra, contrôle bande passante, multi-chemins TCP
Wang et al. [63]	Planification des ressources réseau	NFV, schéma de virtualisation de réseau multi-chemins

Chapitre 2

Utilisation de l'apprentissage automatique dans les réseaux définis par logiciel

Le futur appartient à ceux qui croient à
la beauté de leurs rêves.
Eleanor Roosevelt (1884-1962)

2.1 Introduction

L'intégration de l'apprentissage automatique (*Machine Learning*, en anglais) dans les réseaux définis par logiciel (SDN) représente un domaine de recherche en pleine expansion, offrant de nouvelles perspectives pour l'optimisation des performances réseau, la prise de décision intelligente et l'automatisation des tâches de gestion. Ce chapitre explore l'évolution de l'intelligence artificielle (IA) et de l'apprentissage automatique, depuis leurs fondements théoriques jusqu'aux dernières avancées, en mettant l'accent sur leur application dans le domaine des réseaux définis par logiciel.

La première section de ce chapitre se penche sur les origines de l'intelligence artificielle, remontant aux travaux pionniers de Warren McCulloch et Walter Pitts dans les années 1940 jusqu'aux développements contemporains tels que l'apprentissage profond (*Deep Learning*, en anglais). En parcourant les différentes phases historiques de l'IA, nous observons l'évolution des modèles et des algorithmes qui ont façonné ce domaine interdisciplinaire.

La deuxième section explore les fondements de l'apprentissage automatique, définissant ses principaux concepts et techniques, notamment l'apprentissage supervisé, non supervisé, semi-supervisé et par renforcement. Nous examinons également les métriques d'évaluation utilisées pour évaluer les performances des modèles d'apprentissage automatique, fournissant ainsi un cadre méthodologique pour appréhender les études ultérieures.

La troisième section se concentre sur l'apprentissage profond, une discipline de l'apprentissage automatique qui a récemment suscité un intérêt croissant en raison de ses performances remarquables dans des domaines tels que la vision par ordinateur, la reconnaissance

vocale et le traitement du langage naturel. Par la suite, nous explorons une architecture spécifique de réseaux neuronaux profonds : les réseaux de neurones convolutifs (CNN).

Enfin, la quatrième section et dernière section examine les différentes études et approches qui intègrent l'apprentissage automatique (ML) dans les réseaux définis par logiciel (SDN) afin d'améliorer la gestion du trafic, la sécurité du réseau, la qualité de service (QoS) et l'efficacité énergétique. En explorant ces travaux, nous mettons en lumière les défis et les opportunités liés à l'utilisation de l'apprentissage automatique dans les réseaux définis par logiciel, tout en identifiant les pistes de recherche futures dans ce domaine en constante évolution.

2.2 Intelligence artificielle

2.2.1 Définition

Dans la littérature, différentes définitions sont attribuées à l'intelligence artificielle (IA). Selon le dictionnaire en ligne *Larousse*, l'IA est définie comme un "ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine" [1].

Une autre définition provient du dictionnaire de *Cambridge* sur l'intelligence artificielle, qui la décrit comme "l'étude de la manière de produire des ordinateurs ayant certaines des qualités de l'esprit humain, telles que la capacité de comprendre le langage, reconnaître des images, résoudre des problèmes et apprendre" [2].

En résumé, l'IA vise à reproduire les capacités cognitives humaines au sein des machines ou, de manière plus générale, des systèmes informatiques.

2.2.2 Histoire de l'intelligence artificielle

Dans ce qui suit, nous présentons les phases cruciales de l'histoire de l'intelligence artificielle depuis l'année 1943, mettant en évidence les évolutions marquantes qui ont façonné ce domaine passionnant à travers le temps [64].

Phase 1 : 1943-1956

Les fondements de l'intelligence artificielle (IA) trouvent leurs racines dans les travaux pionniers de Warren McCulloch et Walter Pitts, datant de 1943. Ils ont élaboré un modèle de neurones artificiels basé sur la physiologie neuronale, la logique propositionnelle de Russell et Whitehead, ainsi que la théorie de calcul de Turing. Leur proposition incluait des concepts tels que l'allumage et l'extinction des neurones, avec la capacité d'apprentissage.

En 1950, Marvin Minsky et Dean Edmonds ont construit le premier ordinateur à réseau neuronal, le SNARC, utilisant 3000 tubes à vide. Minsky a ensuite étudié la computation universelle dans les réseaux neuronaux à Princeton. Des travaux préliminaires, tels que les

1. https ://www.larousse.fr/
2. https ://dictionary.cambridge.org/dictionary

programmes de jeu de dames de Christopher Strachey et Arthur Samuel en 1952, ont également contribué à l'émergence de l'IA.

Alan Turing a joué un rôle crucial en présentant le test de Turing, l'apprentissage machine, les algorithmes génétiques et l'apprentissage par renforcement dans son article de 1950. Il a envisagé la création d'une IA de niveau humain par le biais de l'apprentissage plutôt que de la programmation manuelle.

En 1955, John McCarthy a organisé un atelier à Dartmouth, rassemblant des chercheurs intéressés par les automates, les réseaux neuronaux et l'intelligence artificielle. Bien que l'atelier n'ait pas conduit à des percées majeures, il a marqué le début de la collaboration entre chercheurs comme Allen Newell, Herbert Simon, et d'autres. Le Logic Theorist (LT), un système de démonstration de théorèmes mathématiques présenté par Newell et Simon, a été une réalisation notable de cette période. Malgré des débuts modestes, ces travaux jetèrent les bases pour les développements futurs de l'IA.

Phase 2 : 1952-1969

Après le succès de LT, Newell et Simon ont créé le General Problem Solver (GPS), qui imitait les protocoles de résolution de problèmes humains. Le succès du GPS a conduit à la formulation de l'hypothèse du système de symboles physiques, affirmant que tout système intelligent manipule des structures de données composées de symboles.

IBM a également contribué avec des programmes d'IA, tels que le Geometry Theorem Prover en 1959 de Herbert Gelernter et le travail révolutionnaire d'Arthur Samuel sur le jeu de dames, utilisant l'apprentissage par renforcement. En 1958, John McCarthy a défini le langage Lisp et proposé l'Advice Taker, un programme représentant la connaissance générale du monde. Marvin Minsky, avec McCarthy, a contribué à l'IA au MIT, se concentrant sur la représentation logique.

Le travail mené au MIT sous la supervision de Marvin Minsky, où des étudiants se sont penchés sur des problèmes spécifiques regroupés sous le terme de "micromondes" (*micro-worlds*, en anglais). Ces micromondes impliquaient des tâches nécessitant de l'intelligence pour être résolues. Divers programmes, tels que SAINT en 1963, ANALOGY en 1968 et STUDENT en 1967, ont été développés pour résoudre des problèmes comme l'intégration de calculs, les analogies géométriques et les problèmes algébriques.

Les réseaux neuronaux de McCulloch et Pitts ont également prospéré, avec des travaux sur la représentation collective de concepts par Winograd et Cowan en 1963, ainsi que des avancées dans les perceptrons par Widrow et Rosenblatt en 1962. Le théorème de convergence du perceptron a démontré sa capacité à ajuster ses connexions afin de s'aligner avec n'importe quel ensemble de données d'entrée, pour autant que cette correspondance soit possible.

Phase 3 : 1966-1973

Les premiers systèmes d'IA ont souvent échoué face à des problèmes plus complexes en raison de deux principales raisons. Tout d'abord, ces systèmes étaient souvent basés sur une "introspection informée" imitant la façon dont les humains accomplissent une tâche, plutôt

que sur une analyse approfondie de la tâche elle-même et des solutions requises. Deuxièmement, il y avait un manque de compréhension de la complexité de nombreux problèmes traités par l'IA, avec des approches initiales basées sur l'essai et l'erreur.

Avant le développement de la théorie de la complexité computationnelle, on croyait à tort que l'augmentation de l'échelle des problèmes pouvait être résolue simplement par un matériel plus rapide et de plus grandes mémoires. Cependant, cette illusion a été rapidement dissipée, notamment dans le domaine de la résolution de théorèmes, où les chercheurs ont rencontré des difficultés à prouver des théorèmes impliquant plus d'une douzaine de faits.

L'idée d'une puissance de calcul illimitée s'est également reflétée dans les premières expériences d'évolution artificielle, qui ont échoué malgré l'optimisme initial. L'incompréhension de l'*explosion combinatoire* a été critiquée dans le rapport Lighthill en 1973, conduisant même à la réduction du soutien à la recherche en IA par le gouvernement britannique.

Une autre difficulté est survenue en raison des limitations fondamentales des structures utilisées pour générer un comportement intelligent. Par exemple, les perceptrons, une forme simple de réseau neuronal, étaient limités dans leur capacité à représenter des informations complexes. Bien que les recherches sur les réseaux neuronaux aient connu un déclin après les travaux de Minsky et Papert en 1969, les nouveaux algorithmes d'apprentissage par rétropropagation ont ultérieurement revitalisé la recherche dans ce domaine dans les années 1980 et 2010, bien qu'ils aient été développés initialement dans d'autres contextes au début des années 1960.

Phase 4 : 1969-1986

L'intelligence artificielle au cours des premières décennies cherchaient à résoudre des problèmes généraux en enchaînant des étapes de raisonnement élémentaires, mais elles étaient limitées face à des problèmes complexes. Une alternative émergeait en utilisant des connaissances spécifiques au domaine pour faciliter le raisonnement. Le programme DENDRAL, développé en 1969 à Stanford, axé sur la spectroscopie de masse moléculaire, illustrait cette approche en incorporant des connaissances spécifiques plutôt que des principes fondamentaux.

Le projet de programmation heuristique (HPP) en 1971 à Stanford, suivi par le système MYCIN pour le diagnostic d'infections sanguines, démontrait l'efficacité des systèmes experts basés sur des connaissances spécialisées. Ces systèmes ont été considérés comme réussis en raison de leur capacité à intégrer des connaissances spécifiques au domaine dans des règles pratiques.

L'avènement de systèmes experts commerciaux, tels que R1 chez Digital Equipment Corporation en 1982, a marqué une croissance significative de l'industrie de l'IA. Les succès initiaux ont été observés dans des domaines tels que la configuration des commandes pour les nouveaux systèmes informatique, avec des économies considérables pour les entreprises.

L'industrie de l'IA a connu une croissance exponentielle dans les années 1980, avec des investissements massifs et la création de nombreux systèmes experts, robots, et logiciels spécialisés. Cependant, cette période a été suivie par ce que l'on appelle "l'hiver de l'IA", caractérisé par des échecs, des difficultés et des promesses non tenues. Les obstacles comprenaient l'inefficacité des méthodes de raisonnement face à l'incertitude et l'incapacité des systèmes à apprendre de l'expérience.

Phase 5 : 1986-2001

Dans les années 1980, plusieurs groupes ont redéveloppé l'algorithme de rétropropagation, initialement conçu dans les années 1960, pour l'appliquer à divers problèmes d'apprentissage en informatique et en psychologie. La diffusion généralisée des résultats en 1986 a suscité un grand enthousiasme. Ces modèles, appelés connexionnistes, étaient perçus comme des concurrents directs des approches symboliques de Newell et Simon ainsi que de l'approche logiciste de McCarthy.

Les modèles connexionnistes, ressurgis dans les années 1980 et 2010, semblent mieux capables de traiter la complexité du monde réel en formant des concepts internes de manière fluide et imprécise. Ils ont également la capacité d'apprendre à partir d'exemples, ajustant leurs paramètres pour s'adapter aux données et améliorer leurs performances futures.

L'intelligence artificielle (IA) a adopté une approche plus scientifique face à la fragilité des systèmes experts. Cette nouvelle approche intègre la probabilité, l'apprentissage automatique, et les résultats expérimentaux.

Les années 1970 ont vu l'essai de nombreuses architectures ad hoc et fragiles, tandis que les années 1980 ont été marquées par la domination des modèles cachés de Markov (HMM). Les HMM reposent sur une théorie mathématique rigoureuse et sont entraînés sur de grands corpus de données, assurant une performance robuste. Cela a conduit à des applications industrielles et grand public généralisées dans la reconnaissance vocale et la reconnaissance de caractères manuscrits.

En 1988, des liens importants ont été établis entre l'IA et d'autres domaines tels que les statistiques, la recherche opérationnelle et la théorie de la décision. L'introduction des réseaux bayésiens par Judea Pearl a renforcé l'acceptation de la probabilité et de la théorie de la décision en IA. Rich Sutton a également contribué en connectant l'apprentissage par renforcement à la théorie des processus de décision de Markov (MDP), ouvrant la voie à des applications en robotique et en contrôle de processus.

La nouvelle appréciation de l'IA pour les données, la modélisation statistique, l'optimisation et l'apprentissage automatique a conduit à la réunification de sous-domaines tels que la vision par ordinateur, la robotique, la reconnaissance vocale, les systèmes multi-agents et le traitement du langage naturel.

Phase 6 : 2001-2011

Les avancées significatives dans la puissance des systèmes informatiques et la création du World Wide Web, ont facilité la formation de vastes ensembles de données, communément appelés **big data**. Ces ensembles comprennent des quantités massives d'images, de vidéos, de mots, ainsi que des données de réseaux sociaux, etc.

Ces développements ont conduit à la conception d'algorithmes d'apprentissage spécialement adaptés à ces ensembles de données volumineux. Dans de nombreux cas, la majorité des exemples dans ces ensembles ne sont pas étiquetés, mais avec une taille suffisamment grande, des algorithmes appropriés peuvent atteindre une précision élevée dans des tâches telles que la désambiguïsation du sens des mots. En 2001, Banko et Brill ont souligné que la taille de l'ensemble de données semble être plus déterminante pour les performances que l'ajustement de l'algorithme.

Le phénomène se répète dans les tâches de vision par ordinateur en 2007, où des ensembles de données massifs ont contribué à des avancées significatives, comme dans le cas du remplissage des lacunes dans les photographies.

L'importance du big data et l'orientation vers l'apprentissage automatique ont revitalisé l'intérêt commercial pour l'intelligence artificielle. Il est à souligner le rôle du big data dans la victoire du système Watson d'IBM dans le jeu questions-réponses télévisé américain "Jeopardy !" en 2011.

Phase 7 : 2011 à présent

En 2011, l'**apprentissage profond** (*deep learning*, en anglais), une sous-discipline de l'apprentissage automatique basée sur des réseaux de neurones artificiels profonds, a connu son véritable essor. D'abord dans la reconnaissance vocale puis dans la reconnaissance d'objets visuels. En 2012, lors de la compétition ImageNet, un système créé par le groupe de Geoffrey Hinton a démontré des performances exceptionnelles, dépassant les systèmes antérieurs.

Depuis lors, l'apprentissage profond a surpassé les performances humaines dans certaines tâches telles que la vision, la reconnaissance vocale, la traduction automatique, le diagnostic médical et les jeux. Ces succès ont suscité un vif intérêt pour l'IA parmi diverses parties prenantes. Cependant, l'apprentissage profond nécessite un matériel puissant, avec des algorithmes exécutant des opérations massivement parallèles. Malgré les progrès, il demeure dépendant de grandes quantités de données d'entraînement et de techniques algorithmiques spécifiques.

En 2017, le programme informatique AlphaGo, basé sur l'apprentissage automatique, a battu le champion du monde de jeu de go[65]. Ce dernier a déclaré que "l'intelligence artificielle est bien trop puissante et ne pourra jamais être battue par un humain".

En 2014, les réseaux antagonistes génératifs (generative adversarial networks ou GANs, en anglais), un type de modèles de machine learning, ont permis de générer des images avec un fort degré de réalisme [66].

Enfin, l'avancement de l'intelligence artificielle se poursuit grâce à des recherches portant sur des modèles toujours plus vastes et des applications étendues.

2.3 Machine learning

2.3.1 Définition

L'apprentissage automatique (*Machine Learning*, en anglais) est une sous-discipline de l'intelligence artificielle, qui se définit comme le processus de résolution d'un problème pratique qui implique (i) la collecte d'un ensemble de données et (ii) la construction algorithmique d'un modèle statistique fondé sur ces données. L'hypothèse sous-jacente est que ce modèle statistique est ensuite utilisé de manière effective pour résoudre le problème pratique en question[67].

En termes plus simples, une autre définition indique que l'apprentissage automatique représente le domaine d'étude visant à permettre à l'ordinateur d'apprendre de manière autonome[1].

2.3.2 Classification de l'apprentissage automatique

L'apprentissage automatique peut être classé en quatre groupes généraux :

1. l'apprentissage supervisé (*supervised learning*, en anglais),
2. l'apprentissage non supervisé (*unsupervised learning*, en anglais),
3. l'apprentissage semi-supervisé (*semi-supervised learning*, en anglais),
4. l'apprentissage par renforcement (*reinforcement learning*, en anglais).

La Figure 2.1 propose une classification des algorithmes employés dans le domaine de l'apprentissage automatique [1]. Elle est structurée de manière hiérarchique, plaçant les quatre catégories principales d'apprentissage automatique au sommet, tandis que les différents algorithmes correspondants sont disposés en dessous.

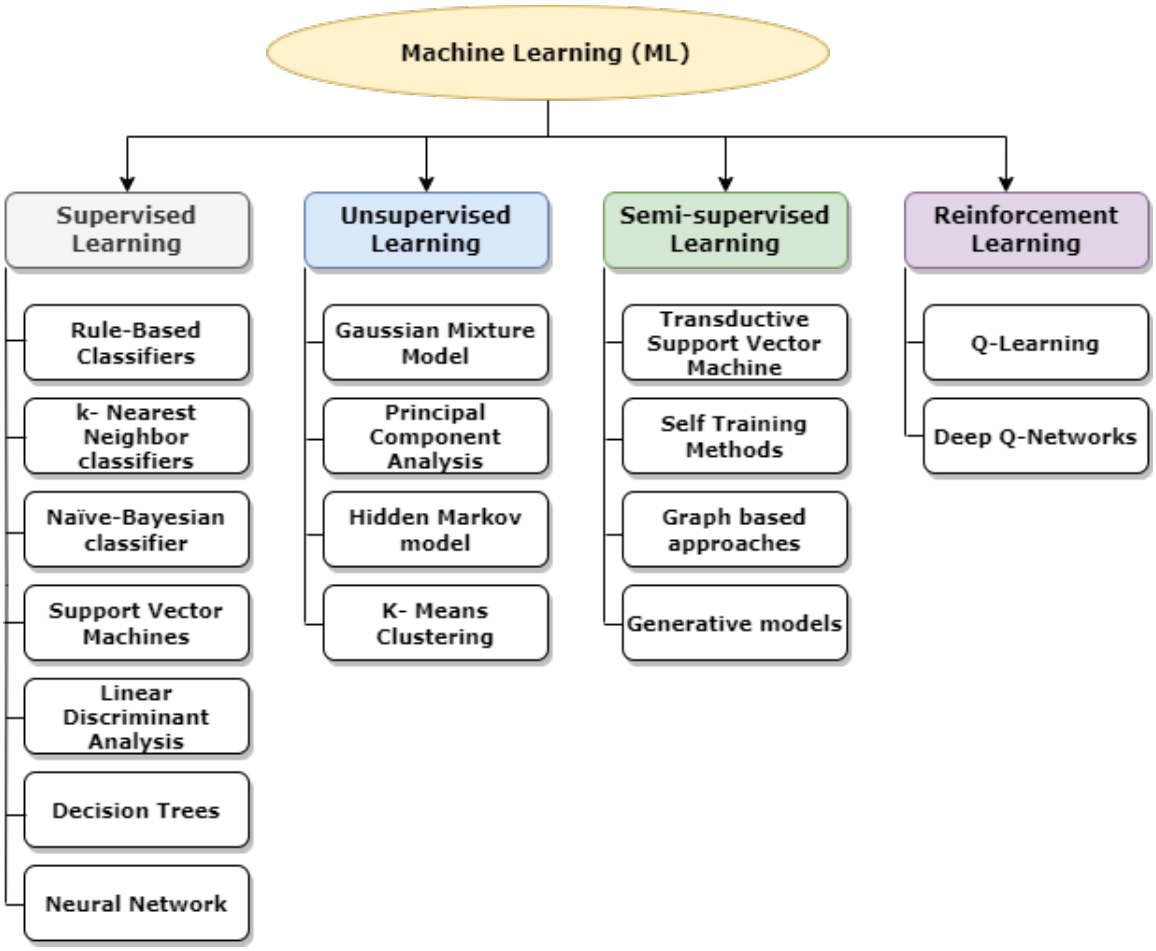


FIGURE 2.1 – Classification de l'apprentissage automatique[1]

Apprentissage supervisé

L'apprentissage supervisé utilise des entrées connues et leurs sorties correspondantes. Le terme "supervisé" fait référence à la présence d'une supervision ou d'une orientation pendant

le processus d'apprentissage. Les données d'entraînement utilisées (*Dataset*) pour former le modèle comprennent à la fois les entrées (*Features*) et les sorties souhaitées (*Label*) [68].

À titre d'exemple, les entrées pourraient consister en des images capturées par une caméra, chacune associée à une sortie spécifiant la catégorie du véhicule, telle que "voiture" ou "moto", entre autres.

Les algorithmes d'apprentissage supervisé les plus courants sont : k-Nearest Neighbour (k-NN), Support Vector Machine (SVM), Neural Network (NN), et Bayesian [1].

L'apprentissage supervisé développe un modèle pour trouver des solutions à deux catégories de problèmes : régression et classification. Les modèles de régression prédisent des réponses continues, tandis que les modèles de classification sont utilisés pour prédire des réponses discrètes[68]. La Figure 2.2 illustre un exemple de Régression et de Classification, démontrant visuellement les résultats des modèles respectifs dans les domaines de la prédiction continue et de la catégorisation des données.

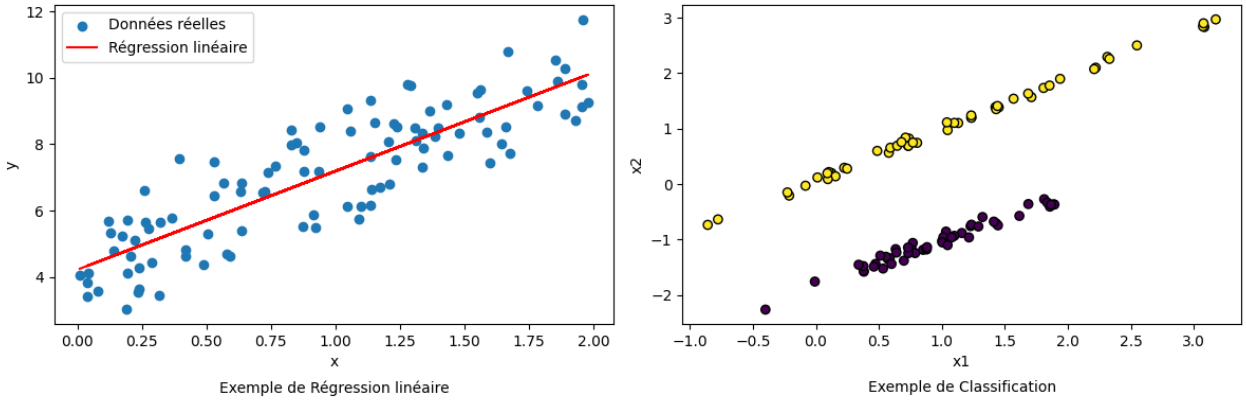


FIGURE 2.2 – Exemple de Régression et de Classification

Plus formellement, l'apprentissage supervisé repose sur les quatre (04) concepts suivants [64][1][69] :

1. **Ensemble de données (Dataset)**

   Soit un ensemble d'apprentissage de N paires d'exemples d'entrée-sortie (x_i, y_i), i de 1 à N.

   Les exemples (x_i, y_i) constituent le *Dataset*.

   y représente la cible (*Label*), c-à-dire la valeur que l'on souhaite prédire.

   x est défini comme le facteur (*feature*), influençant la valeur de y.

2. **Modèle et ses paramètres**

   Un modèle, en tant que fonction mathématique, est élaboré à partir du *Dataset*. Il peut prendre la forme d'un modèle linéaire ou d'un modèle non linéaire, comme illustré dans la Figure 2.3.

   Les paramètres sont les éléments modifiables du modèle qui sont ajustés pendant le processus d'apprentissage afin d'améliorer ses performances.

   Par exemple, dans la fonction $f(x) = ax + b$, les valeurs de a et b sont considérées comme des paramètres du modèle.

3. **Fonction Coût**
   La *Fonction Coût* mesure l'ensemble des erreurs, retournées par le modèle, par rapport au *Dataset*.
   L'objectif est de minimiser la *Fonction Coût* pour améliorer la précision des prédictions, ce qui signifie que le modèle devra produire de petites erreurs.
4. **Algorithme d'apprentissage**
   L'objectif principal de l'apprentissage supervisé consiste à déterminer les paramètres du modèle afin de minimiser la *Fonction Coût*.
   Pour ce faire, on a recours à un algorithme d'apprentissage, dont l'exemple le plus répandu est l'algorithme de *descente de gradient* (*Gradient Descent, en anglais*).

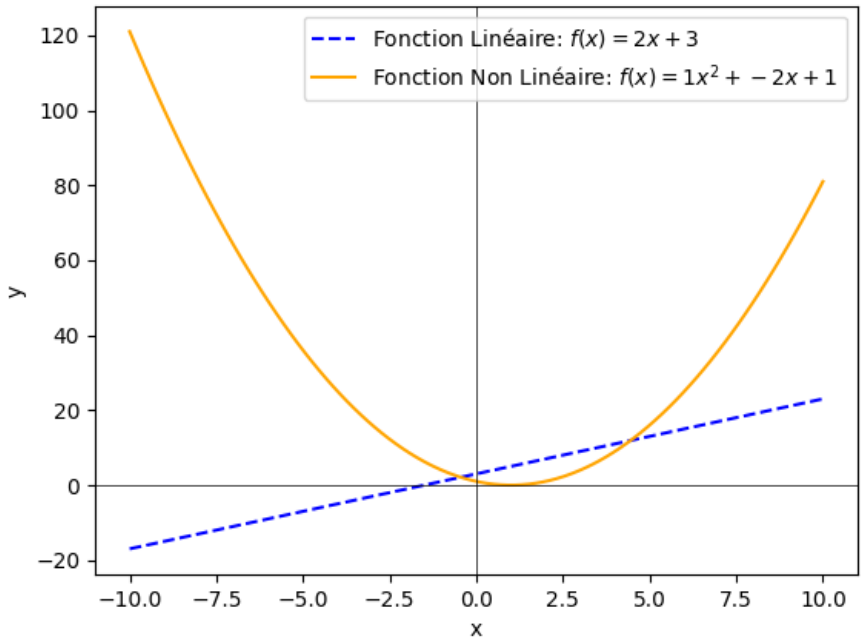


FIGURE 2.3 – Fonctions Linéaire et Non Linéaire

Apprentissage non supervisé

En apprentissage non supervisé, aucune donnée de sortie n'est fournie ; seuls les éléments d'entrée sont exploités dans le processus d'apprentissage.

On dispose ainsi d'un ensemble de données (*Dataset*) avec des entrées (x) sans leurs correspondantes sorties (y), le rendant ainsi non étiqueté (*Unlabelled*) avec les exemples (x_i), i de 1 à N[67].

La machine apprend alors à identifier des structures au sein des données (x) qui lui sont présentées.

L'algorithme opère une classification des éléments d'entrée en groupes désignés sous le terme de *Clusters*. Ainsi, tout nouvel élément d'entrée peut être assigné au groupe approprié [68].

Le *K-Mean Clustering* se distingue comme l'algorithme le plus largement adopté pour résoudre les problèmes de *Clustering*, consistant à regrouper des données en fonction de leurs structures communes.

Apprentissage semi-supervisé

L'apprentissage semi-supervisé fusionne les approches supervisée et non supervisée pour élaborer une fonction de mapping à partir d'entrées de données comprenant à la fois des éléments étiquetés et non étiquetés [68].

Son objectif principal est de classer une partie des données non étiquetées en tirant parti des informations disponibles dans l'ensemble de données étiquetées (*Labeled Dataset*) [1].

Transductive Support Vector Machine (TSVM) et les approches basées sur les graphes (*Graph based approaches*) font partie des algorithmes utilisés dans le domaine de l'apprentissage semi-supervisé.

Apprentissage par renforcement

L'apprentissage par renforcement consiste à acquérir la connaissance des relations entre les entrées et les sorties grâce à un système de récompense *(Reward)*. L'algorithme apprend en interagissant avec son environnement.

Grâce à ce processus, l'agent autonome peut ajuster ses actions en fonction des récompenses reçues, perfectionnant ainsi sa capacité à atteindre ses objectifs de manière optimale [68].

Un exemple d'algorithme d'apprentissage par renforcement est le *Q-Learning*.

2.3.3 Évaluation des performances des modèles d'apprentissage automatique

Des métriques d'évaluation sont utilisées dans le contexte de l'apprentissage automatique pour évaluer la performance des modèles. Pour la classification, on fait souvent référence à des métriques telles que la matrice de confusion, la précision et le rappel ainsi que le *F-score*. En ce qui concerne la régression, les principales métriques d'évaluation comprennent l'erreur quadratique moyenne racine (*Root Mean Squared Error (RMSE)*) ainsi que les quantiles d'erreur [67][68].

Matrice de confusion

La mesure matrice de confusion (*confusion matrix*, en anglais) est une table qui résume les prédictions du modèle de classification en comparaison avec les exemples appartenant à différentes classes dans un Dataset. Dans un problème de classification binaire, la matrice de confusion peut prendre deux valeurs possibles : classe positive et classe négative, avec les cas suivants :

— Vrais positifs : Les exemples de la classe positive correctement prédits par le modèle.

— Vrais négatifs : Les exemples de la classe négative correctement prédits par le modèle.
— Faux positifs : Les exemples qui appartiennent à la classe négative, mais ont été incorrectement prédits comme appartenant à la classe positive.
— Faux négatifs : Les exemples qui appartiennent à la classe positive, mais ont été incorrectement prédits comme appartenant à la classe négative.

Considérons une illustration dans un problème de classification binaire où un modèle formé pour reconnaître si un email est un spam ou non. Le Tableau 2.1 affiche la matrice de confusion pour notre exemple.

TABLE 2.1 – Matrice de confusion

Etiquettes réelles	**spam (prédit)**	**non spam (prédit)**
spam	42	4
non spam	21	338

La matrice de confusion pour cet exemple aurait deux catégories principales : les vrais positifs (42 exemples correctement classés en tant que "spam"), les vrais négatifs (338 exemples correctement classés en tant que "non spam"), ainsi que les faux positifs (21 exemples incorrectement classés comme "spam") et les faux négatifs (4 exemples incorrectement classés comme "non spam").

Enfin, la matrice de confusion est un outil essentiel pour calculer deux autres mesures de performance importantes : la précision et le rappel.

Précision

La mesure de précision (*precision*, en anglais) est une métrique évaluant la proportion d'instances pertinentes (positives) parmi toutes les instances classées comme positives.

En d'autres termes, la précision représente le nombre d'éléments vrais positifs divisé par le nombre total d'éléments classés comme positifs (incluant vrais positifs et faux positifs). Sa formule est la suivante :

$$\text{Précision} = \frac{\text{Vrais positifs}}{\text{Vrais positifs} + \text{Faux positifs}}$$

Rappel

La mesure de rappel (*recall*, en anglais) est le rapport entre le nombre de vrais positifs et la somme des vrais positifs et des faux négatifs dans le Dataset. Sa formule est représentée comme suit :

$$\text{Rappel} = \frac{\text{Vrais positifs}}{\text{Vrais positifs} + \text{Faux négatifs}}$$

F-score

Le *F-score*, également appelé *F1-score* ou *F Measure*, est une mesure qui représente la moyenne harmonique entre la précision et le rappel. Elle atteint sa valeur optimale à 1 (correspondant à une précision et un rappel parfaits) et son point le moins favorable à 0. Le calcul de la mesure F s'effectue de la manière suivante :

$$\text{F-score} = 2 \times \frac{\text{precision} \times \text{rappel}}{\text{precision} + \text{rappel}}$$

Erreur quadratique moyenne racine

L'erreur quadratique moyenne racine (*Root Mean Squared Error (RMSE)*), mesure la moyenne des erreurs quadratiques entre les valeurs prédites par le modèle ($\hat{y}_i$) et les valeurs réelles observées (y_i).

En considérant n comme le nombre total d'observations, la métrique RMSE est calculée comme suit :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Une valeur basse du *RMSE* indique une meilleure performance du modèle, signifiant que les prédictions sont plus proches des valeurs réelles.

Quantiles d'erreur

La métrique *RMSE* calcule la moyenne des erreurs dans l'ensemble des données. Cependant, en présence de valeurs aberrantes (*outliers*, en anglais) dans le *Dataset*, leur impact sur la valeur moyenne peut être considérable. Pour contrer cette influence, il est possible d'utiliser une métrique plus robuste connue sous le nom de quantiles d'erreurs (*Quantiles of errors*, en anglais).

Cette approche prend en compte l'Erreur Médiane Absolue en Pourcentage (*Median Absolute Percentage Error (MAPE)*), offrant ainsi une évaluation moins sensible aux effets disproportionnés des valeurs aberrantes.

La formulation suivante exprime la médiane des pourcentages absolus d'erreur (*MAPE*) entre les valeurs réelles (Y_i) et les valeurs prédites ($\hat{Y}_i$) :

$$\text{MAPE} = \text{Median}\left(\frac{\left|Y_i - \hat{Y}_i\right|}{Y_i} \times 100\right)$$

Il est à noter qu'un score MAPE plus bas indique une meilleure précision du modèle.

Prenons un exemple pour illustrer le *MAPE*, en considérant un ensemble de données avec les valeurs figurant dans le Tableau 2.2.

TABLE 2.2 – Exemple illustrant le *MAPE*

Prévisions	**Valeurs réelles**	**Valeurs prédites**	**Erreur (%)**
1	200	160	$\frac{\lvert 200-160 \rvert}{100} = 20\%$
2	50	70	$\frac{\lvert 50-70 \rvert}{50} = 40\%$

Par conséquent, la moyenne du *MAPE* serait la somme des erreurs divisée par le nombre d'observations, ce qui donne $(20\% + 40\%)/2$, soit *MAPE* = 30%.

2.4 Deep Learning

2.4.1 Définition

L'apprentissage profond (*Deep Learning*, en anglais) est une approche de l'apprentissage automatique et repose sur l'utilisation de réseaux neuronaux artificiels afin de reproduire le processus d'apprentissage observé dans le cerveau humain [68][67].

En général, *"Deep Learning"* et *"Deep Neural Network Learning"* sont utilisés de la même manière, car ils font référence au même concept.

La Figure 2.4 offre une représentation visuelle de la classification de *Deep Learning* dans le domaine de l'Intelligence Artificielle (IA) [66].

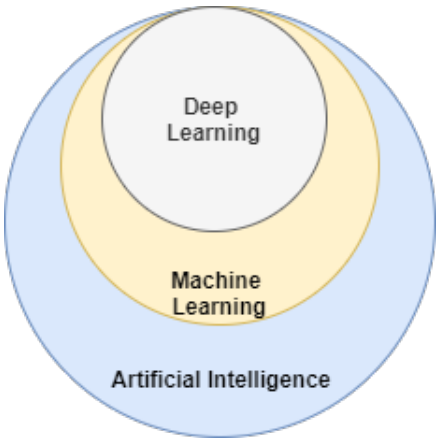


FIGURE 2.4 – Deep Learning et l'IA

Cette approche est couramment employée dans des domaines tels que la reconnaissance d'objets, la reconnaissance faciale, l'analyse des sentiments, ainsi que l'analyse du langage naturel, entre autres.

Au cours de ces dernières années, le *Deep Learning* a connu une croissance significative et sa popularité a augmenté, principalement en raison de l'avènement d'ordinateurs plus puissants, de l'accès à des ensembles de données plus volumineux et du développement de techniques facilitant l'entraînement de réseaux plus complexes.

2.4.2 Classification de Deep Learning

Le domaine du *Deep Learning* peut être classé en deux catégories : les modèles discriminatifs profonds et les modèles génératifs. Les modèles discriminatifs profonds englobent trois approches principales : les réseaux neuronaux récurrents (RNN), les réseaux neuronaux convolutifs (CNN) et les réseaux neuronaux profonds (DNN). Il convient de noter que les réseaux neuronaux profonds font référence à des *Perceptrons Multicouches* (MLP) comportant un nombre de couches supérieur à trois [70].

Quant aux modèles génératifs, ils se déclinent en quatre approches distinctes : les machines de Boltzmann restreintes (RBM), les machines de Boltzmann profondes (DBM), les réseaux de croyances profondes (DBN) et les autoencodeurs profonds (DAE) [70].

Comme mentionné précédemment, la Figure 2.5 présente les deux principales catégories d'algorithmes d'apprentissage profond. Chacune de ces catégories met en lumière les différentes techniques utilisées dans ce domaine.

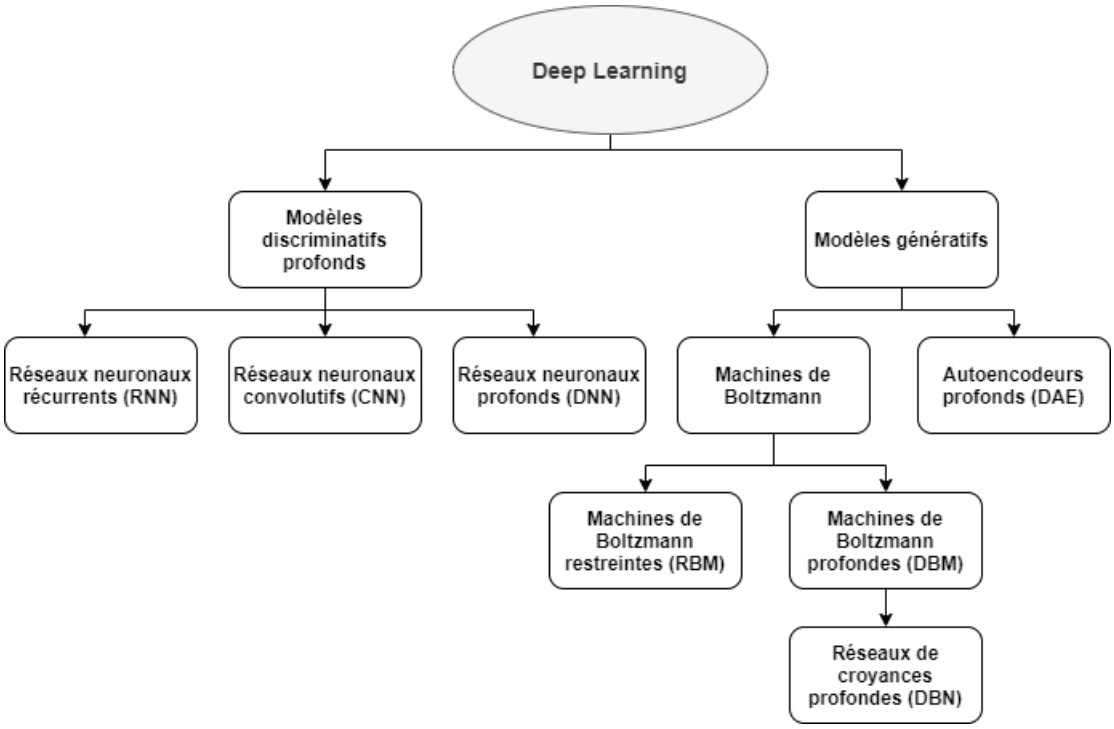


FIGURE 2.5 – Types de modèles de *Deep Learning*

2.4.3 Réseaux de Neurones

Perceptron simple

Le *perceptron* simple est l'un des premiers modèles de neurone artificiel développés. L'idée fondamentale du *perceptron* était de s'inspirer du fonctionnement du réseau neuronal humain afin de résoudre certaines problématiques.

Son innovation a considérablement propulsé le domaine de l'intelligence artificielle, réalisant des progrès spectaculaires. Cette avancée est souvent qualifiée de révolution du *Machine Learning.*

Le *perceptron* représente le réseau neuronal artificiel le plus élémentaire, qui est composé d'un seul neurone. C'est un modèle de classification binaire qui peut réaliser une séparation linéaire entre deux ensembles de points [71].

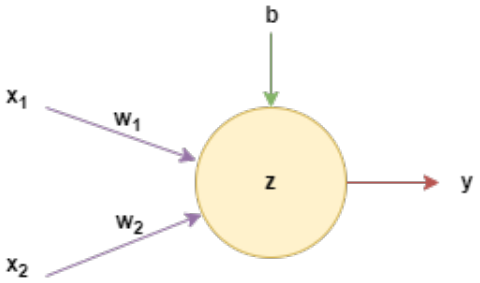


FIGURE 2.6 – Perceptron

La Figure 2.6 illustre le principe de fonctionnement du *perceptron* pour un neurone à deux entrées x_1 et x_2, tandis que w_1 et w_2 représentent les poids associés, et b est le biais. Le petit cercle dans la figure représente le neurone.

Le neurone effectue un calcul linéaire des entrées x_1 et x_2, comme présenté par l'Équation 2.1, suivie de l'application d'une fonction d'activation. Cette dernière produit la sortie y du *perceptron* sous forme d'une fonction de décision.

$$z = w_1.x_1 + w_2.x_2 + b \tag{2.1}$$

Dans le cas de la prédiction directe d'une étiquette binaire, 0 ou 1, on utilise une fonction de seuil (*threshold*, en anglais), représentée dans l'Equation 2.2.

$$y = \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{sinon} \end{cases} \tag{2.2}$$

Pour améliorer ce modèle de *perceptron*, nous devrons prédire la probabilité d'appartenance à une classe spécifique. Dans ce cas, nous utilisons une fonction d'activation telle que la fonction Sigmoïde (Logistique) $\sigma(z)$. Cette fonction est représentée par l'Equation 2.3.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

Dans le contexte des réseaux de neurones artificiels, diverses fonctions d'activation sont employées, parmi lesquelles les plus courantes sont exposées dans la Sous-section 2.4.3.

De manière générale, la formule générique du perceptron est représentée par l'Equation 2.4, où $w_i, b \in \mathbb{R}$ et $i \in \mathbb{N}$.

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b \tag{2.4}$$

L'objectif du *perceptron* est d'ajuster les poids w_i et le biais b afin d'obtenir un modèle optimal, ce qui constitue la phase d'apprentissage ou d'entraînement.

Pour cela, une fonction coût (***Cost Function***, en anglais) est définie afin de minimiser les erreurs entre les valeurs prédites et les valeurs réelles du *Dataset*. Par exemple, la fonction coût *Log Loss* peut être utilisée, dont la formule est donnée dans l'Equation 2.5.

$$L = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i \log(\sigma(z_i)) + (1-y_i)\log(1-\sigma(z_i))\right) \tag{2.5}$$

où m est le nombre d'échantillons dans le *Dataset*, y_i est la valeur réelle de la classe pour l'échantillon i. Enfin, $\sigma(z_i)$ est la probabilité prédite de l'échantillon i, le résultat de la fonction d'activation Sigmoide dans notre exemple précédent.

Pour minimiser la fonction coût, on utilise un algorithme d'optimisation la *Descente de Gradient*, qui ajuste itérativement les paramètres du modèle dans la direction du gradient de la fonction coût.

Les Équations 2.6 et 2.7 représentent la formule de la *Descente du Gradient* pour la fonction de coût *Log Loss* respectivement pour les poids et le biais.

$$\frac{\partial L}{\partial w_j} = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i - \sigma(z_i)\right)x_{ij} \tag{2.6}$$

$$\frac{\partial L}{\partial b} = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i - \sigma(z_i)\right) \tag{2.7}$$

où $L(w)$ est la fonction coût *Log Loss*, w_j est le paramètre (poids) à mettre à jour, m est le nombre d'échantillons dans le *Dataset*, y_i est la valeur réelle de la classe pour l'échantillon i, $\sigma(z_i)$ est la sortie prédite de l'échantillon i (résultant de la fonction d'activation sigmoïde), x_{ij} est la valeur de l'entrée correspondant au poids w_j pour l'échantillon i.

Ensuite, la méthode de la *Descente de Gradient* sera employée pour ajuster les paramètres du modèle en suivant la direction de la fonction coût la plus basse, conformément aux Equations 2.8 et 2.9.

$$w_i = w_i - \alpha\frac{\partial L}{\partial w_i} \tag{2.8}$$

$$b = b - \alpha\frac{\partial L}{\partial b} \tag{2.9}$$

α est le taux d'apprentissage (***Learning Rate***, en anglais). C'est un hyperparamètre essentiel dans l'algorithme d'apprentissage, car elle influence considérablement le processus d'optimisation. Si elle est trop élevée, l'algorithme risque d'osciller autour de la solution optimale voire de diverger. En revanche, une valeur trop faible entraînera une convergence extrêmement lente.

Le processus d'apprentissage débute par une initialisation aléatoire des paramètres, suivi de la définition du nombre maximal d'itérations (***Epochs***, en anglais). À chaque itération, les paramètres sont ensuite mis à jour en suivant la direction du gradient (ou la dérivée) de la fonction coût par rapport à ces paramètres. Ainsi, les paramètres sont ajustés dans le sens qui réduit la valeur de la fonction coût.

En fait, pour chaque itération jusqu'à atteindre le nombre maximal d'itérations, et pour chaque exemple dans le *Dataset*, on procède comme suit : prédire une valeur, calculer l'erreur entre la prédiction et la valeur réelle (fonction coût), calculer le gradient, et enfin mettre à jour les paramètres w_i et b.

La Figure 2.7 résume le processus itératif par lequel les poids et le biais sont ajustés pour minimiser l'erreur de prédiction.

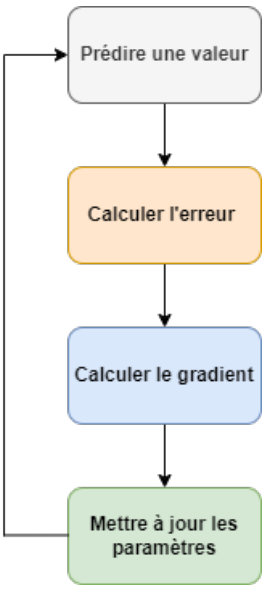


FIGURE 2.7 – Etapes d'apprentissage

Perceptron multicouche

Le perceptron est adapté à la résolution de problèmes de classification binaire simples. Cependant, il ne peut pas résoudre des problèmes plus complexes. Les Perceptron multicouche, ou les réseaux de neurones multicouches (*MultiLayer Perceptron (MLP)*), ont été développés pour surmonter ces contraintes en permettant l'apprentissage de représentations plus complexes [72].

Les réseaux de neurones multicouches représentent un modèle d'apprentissage profond. Ils sont constitués de couches d'entrées, de couches cachées et d'une couche de sortie. La Figure 2.8 illustre l'architecture d'un perceptron multicouches à trois couches.

La couche d'entrée est constituée des variables d'entrée ainsi que du premier neurone, tandis que les couches cachées sont formées de neurones intermédiaires. Enfin, la couche de sortie est représentée par le dernier neurone, qui calcule la valeur résultante de la prédiction.

A la différence du perceptron simple, le processus d'apprentissage du perceptron multicouche se déroule en deux étapes : (i) Propagation avant (***Forward Propagation***) et (ii) Propagation arrière (***Back Propagation***).

Propagation avant :

Pour chaque neurone j situé dans la couche cachée l, la somme pondérée z_j^l est calculée selon l'Equation 2.10, où x_i représente les valeurs d'entrée, w_{ij}^l les poids et b_j^l le biais.

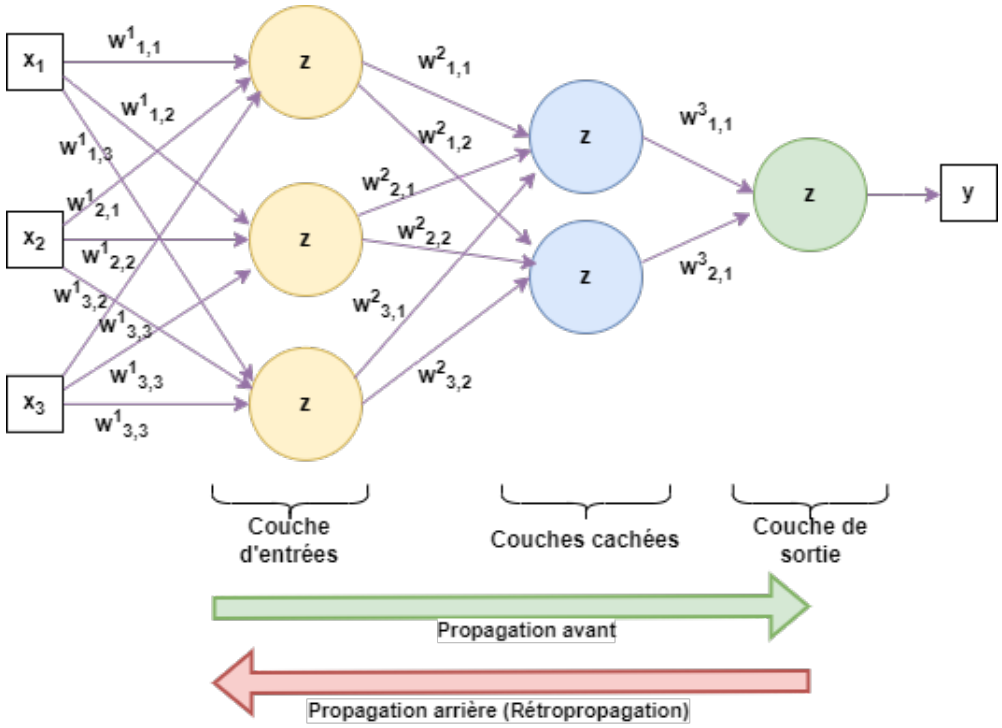


FIGURE 2.8 – Perceptron multicouches

$$z_j^l = \sum_i w_{ij}^l x_i + b_j^l \tag{2.10}$$

Ensuite, une fonction d'activation σ est appliquée à z_j^l pour obtenir la sortie y_j^l du neurone j, comme présenté dans l'équation 2.11.

$$y_j^l = \sigma(z_j^l) \tag{2.11}$$

Ce processus de calcul est répété pour chaque neurone dans chaque couche cachée jusqu'à ce que la couche de sortie soit atteinte.

Propagation arrière :

L'erreur de prédiction est calculée en comparant la sortie prédite avec la valeur réelle à l'aide d'une fonction coût L.

Ensuite, nous procédons à l'ajustement des paramètres du modèle. Pour chaque poids w_{ij}^l et chaque biais b_j^l, les deux étapes suivantes sont effectuées :

Tout d'abord, le gradient de la fonction de coût est calculé par rapport à ces paramètres.

Par la suite, les poids et les biais sont mis à jour en utilisant un algorithme d'optimisation (ex. la Descente de Gradient), comme illustré dans les équations 2.12 et 2.13, où α représente le taux d'apprentissage.

$$w_{ij}^l = w_{ij}^l - \alpha \frac{\partial L}{\partial w_{ij}^l} \tag{2.12}$$

$$b_j^l = b_j^l - \alpha \frac{\partial L}{\partial b_j^l} \tag{2.13}$$

Fonctions d'activation

Dans le domaine du *Deep Learning*, diverses fonctions d'activation sont utilisées. Parmi les plus fréquemment rencontrées, on distingue notamment [64] :

1. La fonction **logistique**, aussi connue sous le nom de fonction **sigmoïde**, est employée dans le contexte de la régression logistique :

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

2. La fonction **tangente hyperbolique (*tanh*)**, dont la plage de valeurs est comprise entre $(-1, 1)$:

$$\tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1}$$

3. La fonction **unité linéaire rectifiée** (*rectified linear unit*, **ReLU**) :

$$\text{ReLU}(z) = \begin{cases} 0 & \text{si } z < 0 \\ z & \text{sinon} \end{cases}$$

Si la valeur d'entrée z est positive, la sortie sera z, sinon, elle sera zéro.

4. La fonction **softplus**, qui est une version légère de la fonction ReLU (*Rectified Linear Unit*) :

$$\text{softplus}(z) = \log(1 + e^{z})$$

Pour fournir une représentation visuelle, la Figure 2.9 illustre les diverses fonctions d'activation utilisées dans les réseaux de neurones.

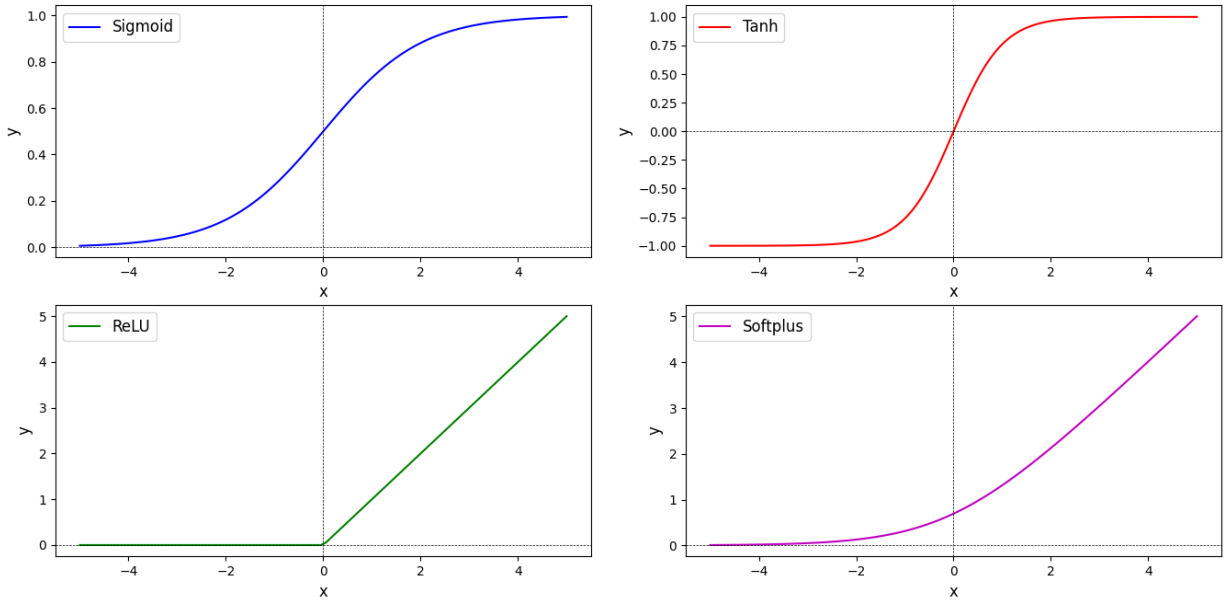


FIGURE 2.9 – Différentes fonctions d'activation

2.4.4 Réseaux de neurones convolutifs (CNN)

Les perceptrons multicouches (MLP) sont limités lorsqu'il s'agit de traiter des données de grande taille en raison de l'augmentation exponentielle du nombre de connexions entre les neurones.

En revanche, les réseaux de neurones convolutifs (*Convolutional Neural Networks (CNN)*), inspirés par le cortex visuel des animaux, restreignent le nombre de connexions entre un neurone et les neurones des couches adjacentes. Cette limitation réduit considérablement la complexité du modèle à entraîner, ce qui rend les *CNN* plus performants pour le traitement de données de grande taille tout en maintenant des performances élevées.

Les réseaux de neurones convolutionnels, présentent un algorithme d'apprentissage supervisé. Ils sont largement utilisés dans les domaines de la vision par ordinateur et de la détection d'objets [64][66].

Les réseaux de neurones convolutionnels sont composés de multiples couches de convolution et de regroupement (***pooling***, en anglais), suivies par une couche entièrement connectée (***fully-connected***, en anglais). Cette dernière résulte de l'aplatissement (***flattening***, en anglais) des données dans une forme adaptée au *Perceptron multicouche*.

Les couches de convolution appliquent des filtres à des données d'entrée à l'aide de plusieurs noyaux (***kernel***, en anglais), tandis que les couches de regroupement réduisent la dimension des données tout en préservant les caractéristiques essentielles [73].

Convolution

En général, l'opération de convolution implique l'application d'un filtre, appelé ***kernel***, à des données représentées sous forme de matrice.

Illustrée dans la Figure 2.10, une opération de convolution est effectuée sur une matrice de taille 4×4, en appliquant un *kernel* de dimension 3×3. Cette technique implique la somme des produits élément par élément des valeurs d'un bloc, de la même taille que le *kernel*, des données d'entrée, avec ceux se trouvant aux mêmes positions dans le *kernel*.

Le *kernel* se déplace sur l'entrée (***input***) en effectuant des déplacements de gauche à droite et de haut en bas, selon ce que l'on appelle les ***strides***, qui correspondent à la taille du *pas*.

L'opération de convolution est symbolisée comme suit :

$$\textit{input} \otimes \textit{kernel}$$

Pour un *kernel* de taille $n \times m$ avec les poids $w_{i,j}$ et un *input* ayant les éléments $x_{i,j}$, les résultats de l'opération de convolution sont exprimés par l'Équation 2.14.

$$\textit{input} \otimes \textit{kernel} = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{i,j} \cdot x_{i,j} \tag{2.14}$$

Le résultat de cette opération est ensuite intégré dans ce que l'on appelle une *sortie convoluée*.

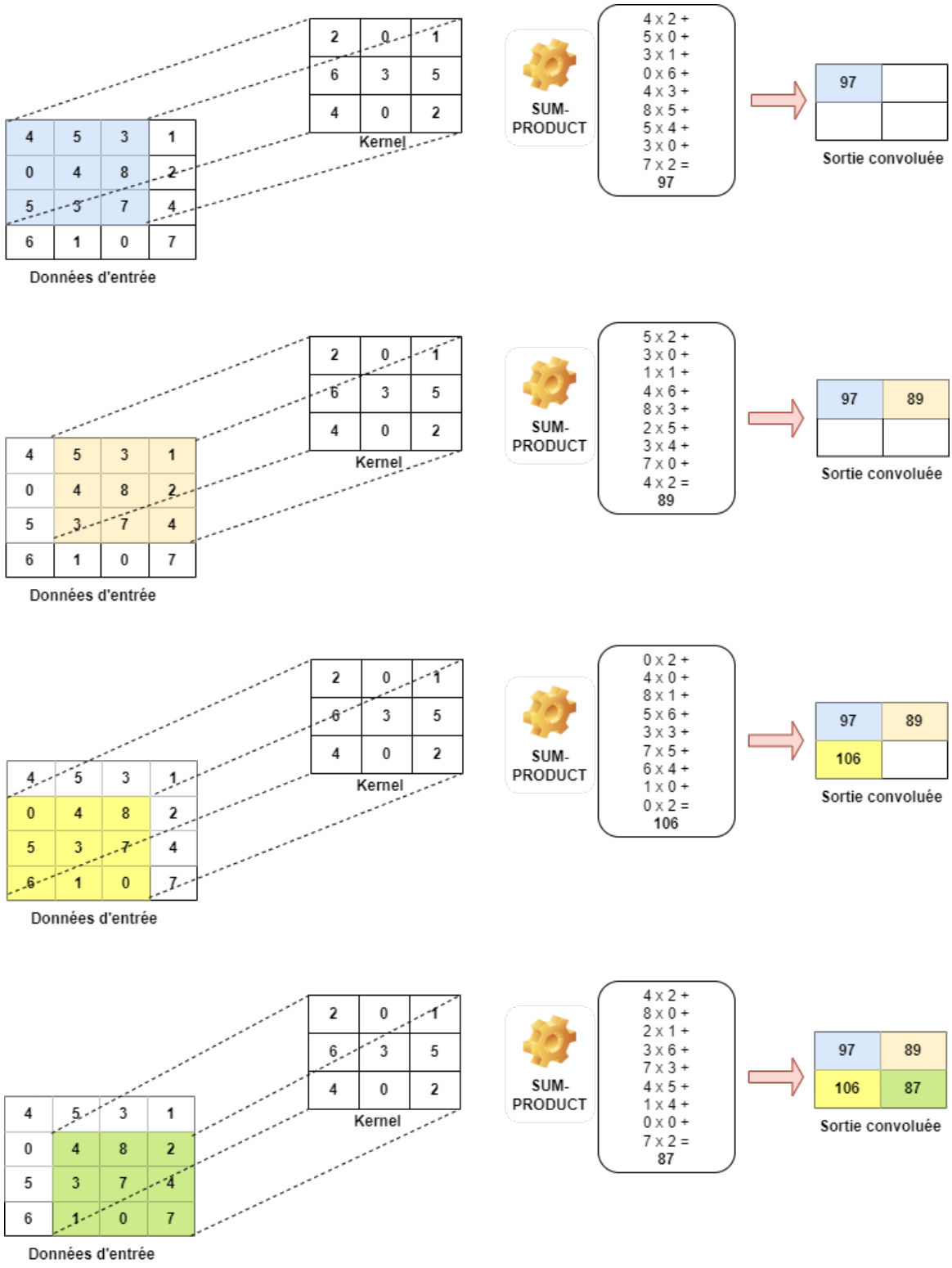


FIGURE 2.10 – Opération de convolution

Dans le contexte des réseaux de neurones convolutifs, lors de l'opération de convolution sur une entrée, nous appliquons les principes fondamentaux des réseaux de neurones artificiels.

Cela signifie que nous ajoutons un biais à chaque convolution et appliquons ensuite une

fonction d'activation. Ainsi, ce processus peut être représenté par une formule, comme illustré dans l'Equation 2.15, où σ désigne une fonction d'activation.

$$y = \sigma(input \otimes kernel + b) \tag{2.15}$$

La Figure 2.11 illustre de façon synthétique le processus de création d'une couche de convolution (***Convolution layer*** en anglais). Le résultat de cette opération est également appelé carte de caractéristiques de sortie (***feature map***, en anglais) ou encore ***activation map***.

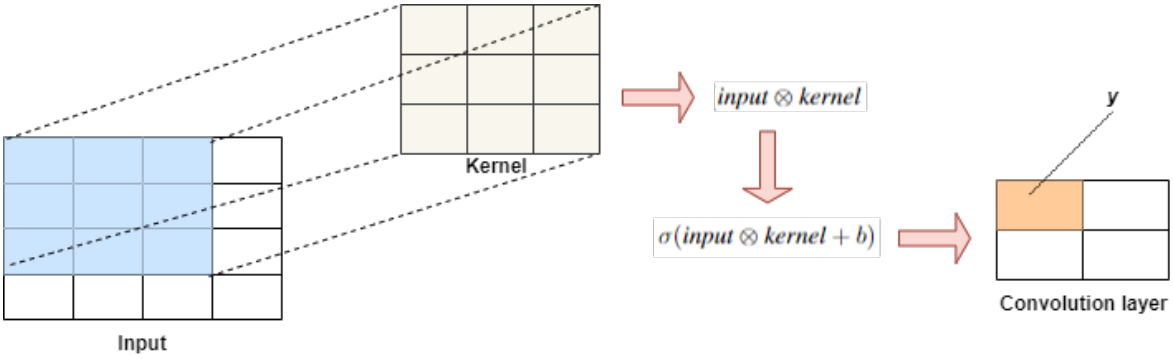


FIGURE 2.11 – Couche de convolution

Il convient de noter que pour un *kernel* de taille $n \times m$, le nombre de paramètres pour une couche de convolution est de $n \times m + 1$.

Remplissage (*Padding*)

L'opération de remplissage (***padding***, en anglais) est un procédé utilisé pour contrôler la taille de la sortie des couches de convolution et préserver la résolution spatiale des données d'entrée à travers le processus de traitement.

Cette technique implique l'ajout de zéros autour de l'entrée (ou des *feature map*) afin de maintenir la taille souhaitée pour les sorties de convolution.

La Figure 2.12 présente un exemple de padding appliqué à une entrée de données.

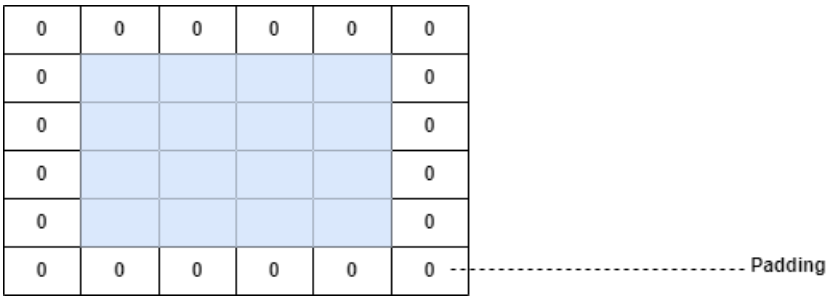


FIGURE 2.12 – Opération de *padding*

Regroupement (*Pooling*)

L'objectif principal du *pooling* vise à diminuer la taille des données tout en préservant leurs caractéristiques essentielles. Cette réduction contribue à rendre les calculs plus efficaces et à accélérer le processus d'apprentissage [74].

Pour ce faire, l'entrée est divisée en cellules régulières, et seule la valeur maximale de chaque cellule est conservée (***Max pooling***). En option, une autre méthode consiste à prendre la valeur moyenne au lieu de la valeur maximale (***Average pooling***).

Ayant le même principe de déplacement que la convolution, le processus de *pooling* parcourt l'entrée (***input***) de gauche à droite et de haut en bas, en effectuant des déplacements conformes à la taille du *pooling*.

La Figure 2.13 illustre le principe de l'opération de *pooling* en utilisant la méthode "Max pooling" sur une entrée de taille 4×4, avec des cellules de dimension 2×2, en appliquant un pas de 2.

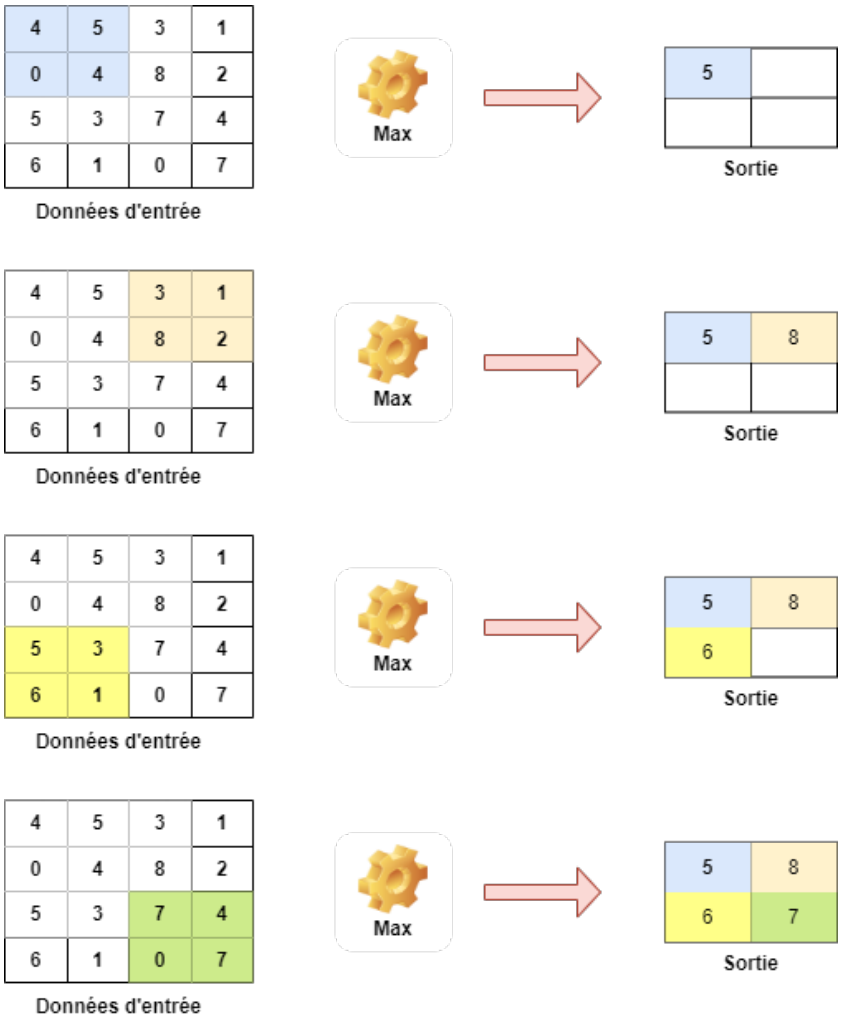


FIGURE 2.13 – Opération de *pooling*

Aplatissement (*Flattening*)

L'opération d'aplatissement est essentielle pour créer la dernière couche, également appelée couche entièrement connectée (*fully-connected*, en anglais). Cette couche permet d'appliquer le principe des réseaux de neurones artificiels en utilisant des couches de neurones interconnectées, comme cela est présenté dans les *Perceptrons multicouches* [73] [74].

La Figure 2.14 illustre la transformation d'une entrée de dimensions 3×3 en une forme aplatie de taille 9×1.

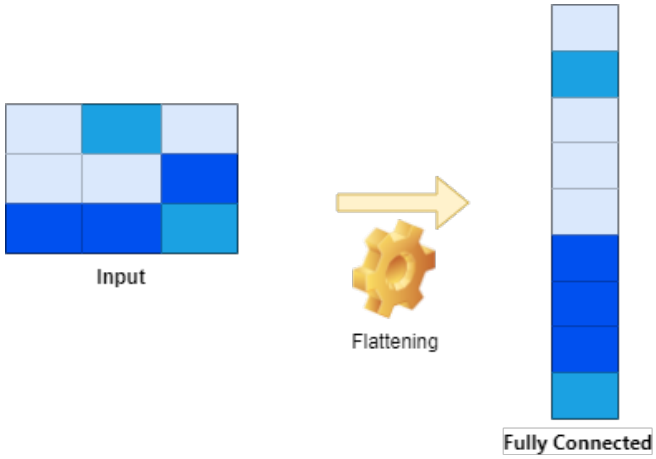


FIGURE 2.14 – Opération de *flattening*

Enfin, la Figure 2.15 présente un exemple d'architecture de réseaux de neurones convolutifs, composée de couches de convolution, de *pooling*, de *flattening*, et enfin le résultat de la prédiction.

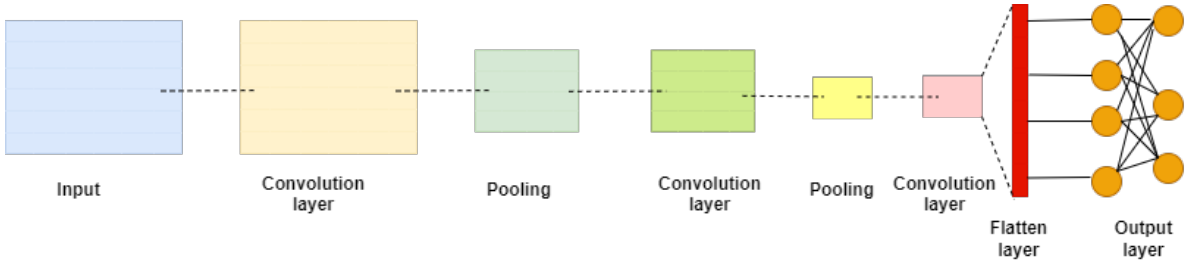


FIGURE 2.15 – Architecture d'un *CNN*

2.5 Utilisation du Machine Learning dans les réseaux définis par logiciel

Les performances croissantes des techniques d'apprentissage automatique (ML) peuvent rendre l'architecture basée sur SDN plus intelligente, révolutionnant ainsi la complexité inhérente aux architectures réseau traditionnelles. Par conséquent, certains aspects du SDN

peuvent bénéficier de cet avantage, tels que l'analyse du trafic basée sur le logiciel et le contrôle centralisé, offrant une gestion plus efficace et plus évolutive des réseaux.

Les techniques d'apprentissage automatique permettent, entre autres, d'optimiser la gestion du trafic réseau et d'atteindre de meilleures performances réseau tout en réduisant l'intervention humaine dans l'administration du réseau.

Pour répondre aux exigences de qualité de service (QoS) adaptées au trafic du réseau mobile 5G dans le cadre des technologies SDN, Chen et al. [75] ont étudié un modèle de réseau de neurones convolutif (*CNN*). Celui-ci catégorise les paquets ou flux en fonction des informations de charge utile, issues des différents types de trafic tels que mMTC, URLLC, eMBB et Internet. Toutefois, les auteurs ont exploré divers paramètres, notamment le nombre et la taille des noyaux ainsi que le taux de désactivation, afin d'identifier les valeurs optimales permettant d'atteindre une précision élevée et une latence faible.

Cheng et al. [76] ont proposé une méthode d'apprentissage augmenté par l'optimisation *(Learning Augmented Optimization*, en anglais), combinant des techniques d'apprentissage profond avec des algorithmes d'optimisation traditionnels inspirés des théories de stabilité de *Lyapunov*, afin de garantir l'efficacité de la segmentation du réseau 5G. Les auteurs ont développé un procédé permettant d'apprendre une solution de segmentation sécurisée à partir de données historiques et d'observations en temps réel, démontrant une amélioration pouvant atteindre jusqu'à 2,6 fois lors des simulations par rapport aux algorithmes conventionnels.

Cependant, notre approche tire parti de l'optimisation du réseau et des techniques d'apprentissage automatique pour proposer des configurations optimales. Ceci vise à réduire les coûts tout en assurant la fiabilité du réseau et le maintien de la qualité de service (QoS) dans un environnement basé sur SDN.

Dans leur étude sur les réseaux activés par SDN et NFV, Pei et al. [77] ont exploré une approche reposant sur l'apprentissage profond pour résoudre le défi de sélection et de chaînage des fonctions virtualisées de réseau (VNF). Ils ont formalisé ce problème sous la forme d'un modèle de programmation en nombres entiers binaires (BIP). En outre, ils ont développé un nouvel algorithme biphasé, nommé DL-TPA, qui tire parti des capacités de l'apprentissage profond, en se basant sur le modèle *Deep Belief Networks (DBNs)*. Cet algorithme utilise deux réseaux distincts, l'un dédié à la sélection des VNF et l'autre au chaînage des VNF, afin d'effectuer ces opérations de manière intelligente et efficiente pour les chaînes de services (SFC). Les résultats de simulations démontrent que l'utilisation de l'apprentissage profond permet d'obtenir efficacement les chemins de routage pour les demandes de SFC, tout en réduisant le temps nécessaire au calcul de ces itinéraires.

En revanche, Kim et al. [78] ont introduit une solution pour prédire l'utilisation des ressources des fonctions virtualisées de réseau (VNF), en utilisant des données de séquence des chaînes de services (SFC), afin de garantir la qualité de service (QoS) souhaitée et d'optimiser l'allocation des ressources. Leur approche repose sur l'utilisation d'un algorithme d'apprentissage profond, plus spécifiquement le réseau de neurones profond LSTM *(Long Short-Term Memory)*. Leur travail s'inscrit dans le contexte de la gestion de la virtualisation des fonctions réseau (NFV), explorant la manière dont la corrélation entre les VNF dans une chaîne de fonctions de service (SFC) peut être exploitée pour prédire la demande future en ressources d'un VNF.

Bouzidi et al. [79] ont proposé un schéma d'ingénierie du trafic dans le contrôleur SDN ONOS. Ce schéma vise à optimiser à la fois le routage des flux réseau et la prédiction du

trafic, dans le but de prévenir les problèmes de congestion réseau. Leur proposition repose sur l'utilisation du réseau de neurones profond LSTM *(Long Short-Term Memory)*.

Assefa et Özkasap [80] ont proposé un framework hybride basé sur l'apprentissage automatique, en tenant compte du trafic, pour économiser l'énergie dans un réseau utilisant la technologie SDN. Le framework proposé, appelé *HyMER*, combine les avantages des techniques d'apprentissage supervisé et par renforcement (*Reinforcement learning*, en anglais), en particulier le *Q-learning*. Son objectif principal est de réduire la consommation d'énergie en minimisant à la fois l'utilisation électrique des switchs et le nombre de liens actifs. Les résultats des expérimentations démontrent que *HyMER* parvient à établir un compromis entre les performances du réseau et son efficacité énergétique, en favorisant un routage économe en énergie.

Khairi et al. [81] ont proposé l'utilisation d'algorithmes d'apprentissage automatique, notamment les arbres de décision (DT), les machines à vecteurs de support (SVM), les arbres de décision extrêmement rapides (EFDT), ainsi qu'une approche hybride de décision-arbre/machine à vecteurs de support (DT-SVM), dans le but d'améliorer la détection et la classification des flux conflictuels au sein des réseaux définis par logiciel (SDN).

L'apprentissage automatique fait l'objet de nombreuses recherches pour améliorer la sécurité dans les réseaux basés sur le SDN. Par exemple, Banitalebi et al. [82] ont proposé une approche combinant les techniques d'apprentissage automatique et les méthodes statistiques pour renforcer la détection des attaques par déni de service distribué (DDoS) au sein des réseaux SDN. La solution proposée repose sur les arbres de décision *REPTree* (*Reduced Error Pruning Tree*, en anglais), des algorithmes d'apprentissage automatique supervisé.

De même, Sudar et al. [83] ont proposé une solution reposant sur les algorithmes d'apprentissage automatique Machine à Vecteurs de Support (SVM) (*Support Vector Machine*, en anglais) ainsi que les arbres de décision. Cette approche vise à détecter les attaques DDoS en examinant les caractéristiques fondamentales du trafic.

Dans une autre étude, Revathi et al. [84] ont avancé une approche qui fusionne des aspects des machines à vecteurs de support (SVM) avec une méthode de mémoire discrète et évolutive, dans le but de détecter les menaces DDoS.

Contrairement à ces solutions, notre framework exploite l'optimisation réseau et la méthodologie d'apprentissage profond (deep learning) pour garantir la qualité de service (QoS) et la fiabilité du réseau, tout en réduisant les coûts.

Afin de fournir une configuration réseau intelligente et autonome, Wang et al. [85] ont proposé un framework axé sur le réseau cognitif pour fournir une Qualité de l'Expérience (QoE) appropriée en utilisant des techniques de l'apprentissage automatique supervisé dans une architecture basée sur SDN. Le framework proposé s'auto-optimise et s'auto-orchestre en tenant compte des informations de surveillance. Ces informations sont recueillies à travers les interfaces SDN sud (SBI) pour obtenir l'état et les métriques du réseau, tandis que les interfaces nord (NBI) sont utilisées pour accéder à divers indicateurs clés de performance (KPI).

Rodríguez et al. [86] ont avancé une proposition visant à améliorer la Qualité de Service (QoS) des réseaux de communication, en optimisant le processus de planification des chemins à l'aide des principes d'apprentissage automatique appliqués aux réseaux définis par logiciel (SDN). Dans cette perspective, la planification des chemins est appréhendée comme un problème de classification multiple. Les résultats obtenus ont identifié le modèle

de machine à vecteurs de support (SVM) comme le classificateur optimal pour déterminer les trajets optimaux entre les points d'extrémité.

Pour synthétiser les travaux de recherche présentés dans cette section, le Tableau 2.3 met en lumière la diversité des approches et des techniques utilisées dans les différents articles pour traiter les problématiques de qualité de service et d'optimisation réseau dans les réseaux basés sur SDN.

Les solutions mentionnées s'appuient sur les techniques d'optimisation du réseau pour fournir des configurations optimales. Malheureusement, leur temps d'exécution est souvent élevé, ce qui pose un problème sérieux à leur capacité à s'adapter aux changements rapides du réseau.

D'autre part, les solutions reposant exclusivement sur les techniques d'apprentissage automatique (ML) offrent généralement des configurations avec des temps de réponse rapides, mais ces configurations ne sont pas toujours optimales.

Notre travail vise à combler cette lacune en proposant une interaction intelligente entre les techniques d'apprentissage automatique (ML) et d'optimisation, afin de fournir rapidement des configurations optimales.

Notre contribution se distingue de toutes les recherches précédemment citées, car nous introduisons un framework générique qui combine les techniques d'apprentissage automatique (ML) et d'optimisation pour fournir rapidement et efficacement une configuration optimale des réseaux activés par SDN. L'objectif principal de notre travail est de réduire les coûts tout en garantissant la fiabilité du réseau dans des délais courts.

2.6 Conclusion

La convergence de l'intelligence artificielle (IA) et des réseaux définis par logiciel (SDN) offre des opportunités sans précédent pour améliorer l'efficacité, la fiabilité et la sécurité des réseaux informatiques. L'histoire de l'intelligence artificielle, du machine learning et du deep learning révèle une évolution remarquable, marquée par des progrès significatifs et une influence croissante.

Depuis ses débuts modestes dans les années 1940 et 1950 avec les travaux pionniers de Warren McCulloch, Marvin Minsky, Alan Turing, et d'autres, l'intelligence artificielle a parcouru un long chemin. Les phases successives ont vu l'émergence de modèles et d'algorithmes de plus en plus complexes, culminant dans l'essor de l'apprentissage profond au cours des dernières décennies.

Le machine learning, en tant que discipline de l'IA, a connu une croissance exponentielle grâce à l'augmentation de la puissance de calcul, à l'abondance de données et aux progrès dans les algorithmes. Ses diverses approches, notamment l'apprentissage supervisé, non supervisé, semi-supervisé et par renforcement, ont révolutionné la façon dont les systèmes informatiques acquièrent des connaissances et prennent des décisions.

Le *Deep Learning*, en particulier, a émergé comme une branche dominante du machine learning, grâce à ses réseaux neuronaux profonds capables d'apprendre des représentations de données complexes. Ses succès dans des domaines tels que la vision par ordinateur, le

TABLE 2.3 – Travaux de recherche sur l'apprentissage automatique et les réseaux SDN

Auteurs	**Approche**	**Techniques de Machine Learning**	**Contribution**
Chen et al. [75]	Modèle de réseau de neurones pour catégoriser les paquets selon le trafic 5G.	Réseau de neurones convolutif (CNN).	Répondre aux exigences de qualité de service (QoS).
Cheng et al. [76]	Apprentissage augmenté par l'optimisation pour la segmentation du réseau.	Apprentissage Profond, Algorithmes d'optimisation traditionnels.	Amélioration de l'efficacité de la segmentation du réseau 5G.
Pei et al. [77]	Utilisation de l'apprentissage profond pour la sélection et le chaînage des fonctions virtualisées de réseau (VNF).	Réseaux de neurones profonds Deep Belief Networks (DBNs).	Proposition d'un algorithme biphasé pour des opérations intelligentes de sélection et de chaînage de VNF.
Kim et al. [78]	Prédiction de l'utilisation des ressources VNF basée sur les données de séquence SFC.	Réseaux de neurones profonds (LSTM).	Optimisation de l'allocation des ressources pour garantir la QoS dans les réseaux NFV.
Bouzidi et al. [79]	Schéma d'ingénierie de trafic pour optimiser le routage et la prédiction du trafic.	Réseaux de neurones profonds (LSTM).	Amélioration de la gestion du trafic et prévention des problèmes de congestion.
Assefa et Özkasap [80]	Framework HyMER pour favoriser un routage économe en énergie.	Apprentissage supervisé, Apprentissage par renforcement.	Réduction de la consommation d'énergie tout en maintenant les performances du réseau.
Khairi et al.	Détection et classification des flux conflictuels dans les réseaux SDN.	Arbres de décision (DT), Machines à vecteurs de support (SVM), Arbres de décision extrêmement rapides (EFDT), Approche hybride DT-SVM.	Amélioration de la détection des attaques et classification des flux conflictuels en utilisant différents algorithmes d'apprentissage automatique.
Banitalebi et al. [82]	Renforcer la détection des attaques DDoS dans les réseaux SDN.	Arbres de décision (REP-Tree), Méthodes statistiques	Renforcement de la détection des attaques DDoS.
Sudar et al. [83]	Détecter les attaques DDoS en analysant les caractéristiques du trafic.	Machines à vecteurs de support (SVM), Arbres de décision.	Amélioration de la détection des attaques DDoS.
Revathi et al. [84]	Détecter les menaces DDoS.	Machines à vecteurs de support (SVM), Mémoire discrète et évolutive.	Amélioration de la détection des menaces DDoS.
Wang et al. [85]	Utilisation d'un framework axé sur le réseau cognitif pour fournir une QoE appropriée.	Techniques de Machine Learning supervisé.	Auto-optimisation et auto-orchestration du réseau pour garantir la QoE en utilisant un framework basé sur le réseau cognitif et les techniques de ML.
Rodríguez et al. [86]	Optimisation du processus de planification des chemins.	Machines à vecteurs de support (SVM).	Amélioration de la QoS en optimisant la planification des chemins.

traitement du langage naturel et bien d'autres ont catalysé l'adoption de techniques d'apprentissage profond dans diverses applications, y compris les réseaux informatiques.

Dans le contexte spécifique des réseaux définis par logiciel, l'utilisation de techniques d'apprentissage automatique offre des avantages significatifs. Les algorithmes d'apprentissage automatique peuvent être appliqués pour optimiser la gestion du trafic, améliorer la qualité de service (QoS), renforcer la sécurité et réduire les coûts opérationnels des réseaux SDN. Des études récentes ont démontré l'efficacité de l'apprentissage profond dans des domaines tels que la segmentation de réseau, la sélection et le chaînage de fonctions réseau virtuelles (VNF), la prédiction de l'utilisation des ressources et la détection des attaques réseau.

Enfin, l'intégration de l'apprentissage automatique dans les réseaux définis par logiciel représente une évolution majeure dans le domaine des technologies réseau. En exploitant les capacités de l'intelligence artificielle, des réseaux informatiques plus intelligents, adaptatifs et sécurisés peuvent être réalisés, ouvrant la voie à un avenir où les réseaux sont non seulement des infrastructures de communication, mais aussi des entités autonomes capables de s'adapter aux besoins changeants et de prendre des décisions intelligentes en temps réel.

Deuxième partie

Partie 2 : Contributions

Chapitre 3

Framework d'optimisation basé sur l'IA

L'imagination est plus importante que le savoir.

Albert Einstein (1879-1955)

3.1 Introduction

Dans ce chapitre, nous introduisons un framework d'optimisation novateur basé sur l'intelligence artificielle (IA), désigné ***DLO-SFC***, abréviation de *Deep Learning Optimization for Service Function Chaining*. Ce framework s'inspire du concept ETSI ZSM (Zero touch network & Service Management) [87][88], pour assurer une automatisation en boucle fermée, facilitant ainsi une configuration efficace du réseau.

Cette solution offre une double opportunité : elle réduit à la fois les coûts opérationnels (OpEx) et les dépenses en capital (CapEx), tout en atténuant le risque d'erreur humaine. De plus, elle contribue de manière significative à la diminution de l'empreinte carbone, amplifiée par l'utilisation intensive d'énergies fossiles comme le gaz et le charbon dans les centres de données.

La première section de ce chapitre se focalise sur une présentation exhaustive du DLO-SFC, mettant en lumière sa conception, son architecture et ses principaux composants. Le DLO-SFC est pensé pour répondre à la demande croissante en automatisation intrinsèque des réseaux, en particulier dans le contexte des réseaux 5G/6G, en offrant des fonctionnalités d'auto-configuration, d'auto-surveillance, d'auto-réparation et d'auto-optimisation.

Le framework propose une approche unifiée en intégrant harmonieusement des algorithmes d'apprentissage profond, des techniques d'optimisation et des heuristiques innovantes, tous encapsulés dans une architecture modulaire et flexible.

Cette approche représente une rupture significative par rapport aux solutions préexistantes, qui tendent à traiter ces problématiques de manière compartimentée et séquentielle.

La deuxième section se concentre sur le déploiement pratique de la plateforme DLO-SFC, décrivant en détail son architecture micro-services basée sur Kubernetes, ainsi que les technologies clés utilisées pour la mise en œuvre réussie de la plateforme, notamment les

contrôleurs SDN basés sur ONOS, le système de surveillance Prometheus, le framework d'intelligence artificielle TensorFlow, le solveur d'optimisation Gurobi Optimizer et la bibliothèque Python NetworkX pour les heuristiques d'optimisation.

3.2 DLO-SFC : Framework d'optimisation basé sur le *Deep Learning*

Au fil de cette section, nous présenterons le framework DLO-SFC, abréviation de *Deep Learning Optimization for Service Function Chaining*. Celui-ci exploite les principes de l'apprentissage profond *(*Deep Learning) en vue de l'optimisation de la chaîne des fonctions de service *(service function chaining (SFC))* au sein des réseaux mobiles.

Les notations pertinentes, présentées dans ce chapitre pour décrire de manière cohérente la solution proposée, sont synthétisées et regroupées dans le Tableau 3.1.

TABLE 3.1 – Synthèse des notations utilisées

Notation	Description	Notation	Description
DLO-SFC	Deep Learning Optimization for Service Function Chaining, le framework proposé.	ND	Network Driver, assure la communication entre le framework DLO-SFC et l'environnement externe, constitué notamment des contrôleurs SDN.
Orchestrator	Composant central du DLO-SFC, assure une interaction intrinsèque avec l'ensemble des éléments du framework.	NMA	Network Monitoring Agent, est une fonctionnalité conçue pour surveiller à la fois le réseau et les changements dans sa topologie.
DLC	Deep Learning Component, utilise son réseau neuronal pour calculer une configuration optimale du réseau, offrant une solution basée sur l'IA.	HC	Heuristic Component, génère une configuration valide dans un délai raisonnable en utilisant un algorithme heuristique.
MC	Model Checker, l'algorithme chargé de vérifier la validité de la solution proposée par le DLC (solution IA).	OC	Optimisation Component, génère une configuration réseau optimisée en utilisant un algorithme d'optimisation.

3.2.1 Vue générale de la plateforme DLO-SFC

Les impératifs avancés par les réseaux cellulaires (5G/6G) engendrent une demande pressante envers une automatisation intrinsèque, déployée de manière exhaustive dans la gestion des infrastructures réseau ainsi que des services de bout en bout, garantissant des opérations d'auto-configuration, d'auto-surveillance, d'auto-réparation et d'auto-optimisation.

Dans cette optique, il est manifeste qu'une révision significative de l'architecture conventionnelle s'impose, une architecture conçue spécifiquement pour la réalisation d'automatismes en boucle fermée, et ce, dans un alignement optimisé avec les principes fondamentaux des algorithmes de l'intelligence artificielle et de l'apprentissage automatique, qui s'appuient sur les données empiriques disponibles en vue d'atteindre ces objectifs.

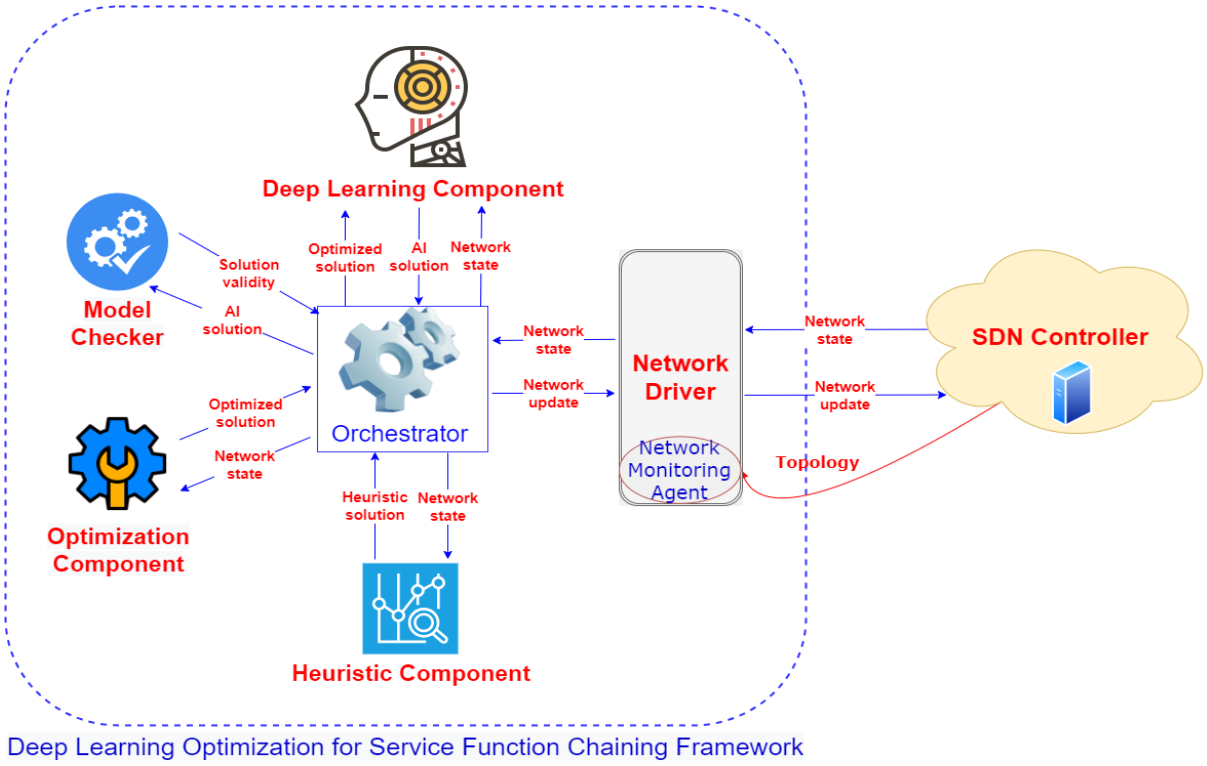


FIGURE 3.1 – Architecture du Framework d'optimisation *DLO-SFC*

Le framework DLO-SFC proposé vise à fournir un cadre d'architecture de bout en bout, qui tire parti des avancées inhérentes à l'apprentissage profond, afin d'automatiser la gestion intrinsèque du réseau en combinaison avec les divers services connexes.

Le framework communique de manière concertée avec les contrôleurs de Réseaux Définis par Logiciel (SDN), garantissant le recueil des données pertinentes relatives au réseau, pour extraire une nouvelle configuration optimale du réseau à l'aide d'algorithmes d'apprentissage profond.

La présente solution repose sur une convergence complète d'algorithmes pluridisciplinaires, fusionnant des paradigmes d'optimisation, des heuristiques ingénieuses et des techniques d'apprentissage profond, tous encapsulés au sein d'un seul et même édifice conceptuel.

Cette démarche se distingue nettement avec les solutions préexistantes qui tendent à traiter ces composantes de manière compartimentée et séquentielle.

Cette combinaison algorithmique permet une synergie émanant de l'interaction concertée de ces modèles, introduisant ainsi une approche originale qui englobe la complexité des défis posés par les réseaux 5G/6G et leurs exigences en matière d'automatisation complète.

3.2.2 Composants de la plateforme DLO-SFC

Le framework *DLO-SFC* s'articule autour de six éléments constitutifs majeurs, à savoir :

1. le Pilote du Réseau, *Network Driver (**ND**)* en anglais,

2. l'Orchestrateur, ***Orchestrator*** en anglais,
3. le Composant d'apprentissage profond, *Deep Learning Component (**DLC**) en anglais*,
4. le Contrôleur de Modèle, *Model Checker (**MC**) en anglais*,
5. le Composant Heuristique, *Heuristic Component (**HC**) en anglais*,
6. le Composant d'Optimisation, *Optimization Component (**OC**) en anglais*.

À cet égard, la Figure 3.1 illustre les divers composants du DLO-SFC, offrant ainsi un aperçu visuel de sa structure et de ses éléments constitutifs.

La dynamique de la collaboration entre ces divers composants inhérents au framework DLO-SFC se trouve illustrée par le diagramme de séquence proposée dans la Figure 3.2.

Le premier composant, ***Network Driver (ND)***, garantit la communication avec l'environnement externe. Cela englobe notamment l'interaction avec les orchestrateurs et les contrôleurs SDN, comme illustré dans les schémas présentés aux Figures 3.1 et 3.2. Ces contrôleurs SDN implémentent les décisions optimales générées par le framework DLO-SFC et surveillent l'état du réseau pour le compte de ce dernier.

Le composant ***Network Driver (ND)*** comprend également la fonction de l'*Agent de Surveillance du Réseau (ASR)*, abrégée en ***NMA*** pour *Network Monitoring Agent*. Cette fonction offre une opportunité considérable de mener une surveillance exhaustive, couvrant à la fois le réseau et les ressources de calcul associées. Elle facilite également la détection et le suivi des modifications dans la topologie du réseau (c'est-à-dire, l'état du réseau), traduisant ainsi l'évolution dynamique de son état.

Il est à noter que le composant *ND* jouit d'une capacité active de mise en œuvre des diverses décisions prises par le framework. Plus précisément, le *ND* est capable d'exécuter les résolutions prises en réponse aux défis et aux impératifs soulevés par l'environnement réseau en constante mutation. Cette approche proactive lui permet de mettre à jour le réseau conformément aux besoins évolutifs et aux exigences émanant du framework.

Le second composant, désigné sous l'appellation ***Orchestrator***, génère une configuration réseau efficiente, apte à cohabiter harmonieusement tout en se conformant à l'état dynamique propre au réseau, ainsi qu'aux indicateurs de performance désirés (KPI). Il constitue le composant central au sein du framework *DLO-SFC*, offrant une interaction intrinsèque avec l'ensemble des éléments de ce dernier, comme le montrent les Figures 3.1 et 3.2.

Grâce au composant *ND*, l'*Orchestrator* reçoit des informations relatives à l'état et à la topologie du réseau, puis il met en œuvre les décisions prises. Ces décisions sont appliquées au sein du réseau par le biais des contrôleurs SDN, via l'API Northboun.

L'Orchestrator doit être capable de s'adapter et de cohabiter afin de fournir des configurations optimales dans un délai raisonnable. Lors de la réception de l'état du réseau depuis le *ND*, l'Orchestrator opère une transmission simultanée de ce dernier en direction des composantes *OC* et *DLC*, comme illustré dans la Figure 3.2.

Il convient de noter que le composant *OC* manifeste une probabilité élevée dans la génération de configurations réalisables comparativement à *DLC*. Cependant, cette probabilité accrue s'accompagne d'un coût inévitable, à savoir le temps d'exécution.

Lorsque l'accent est résolument porté sur l'optimisation, il en résulte parfois des conséquences défavorables en raison d'un manque de synchronisation entre les modifications apportées au réseau et les prises de décision qui en découlent.

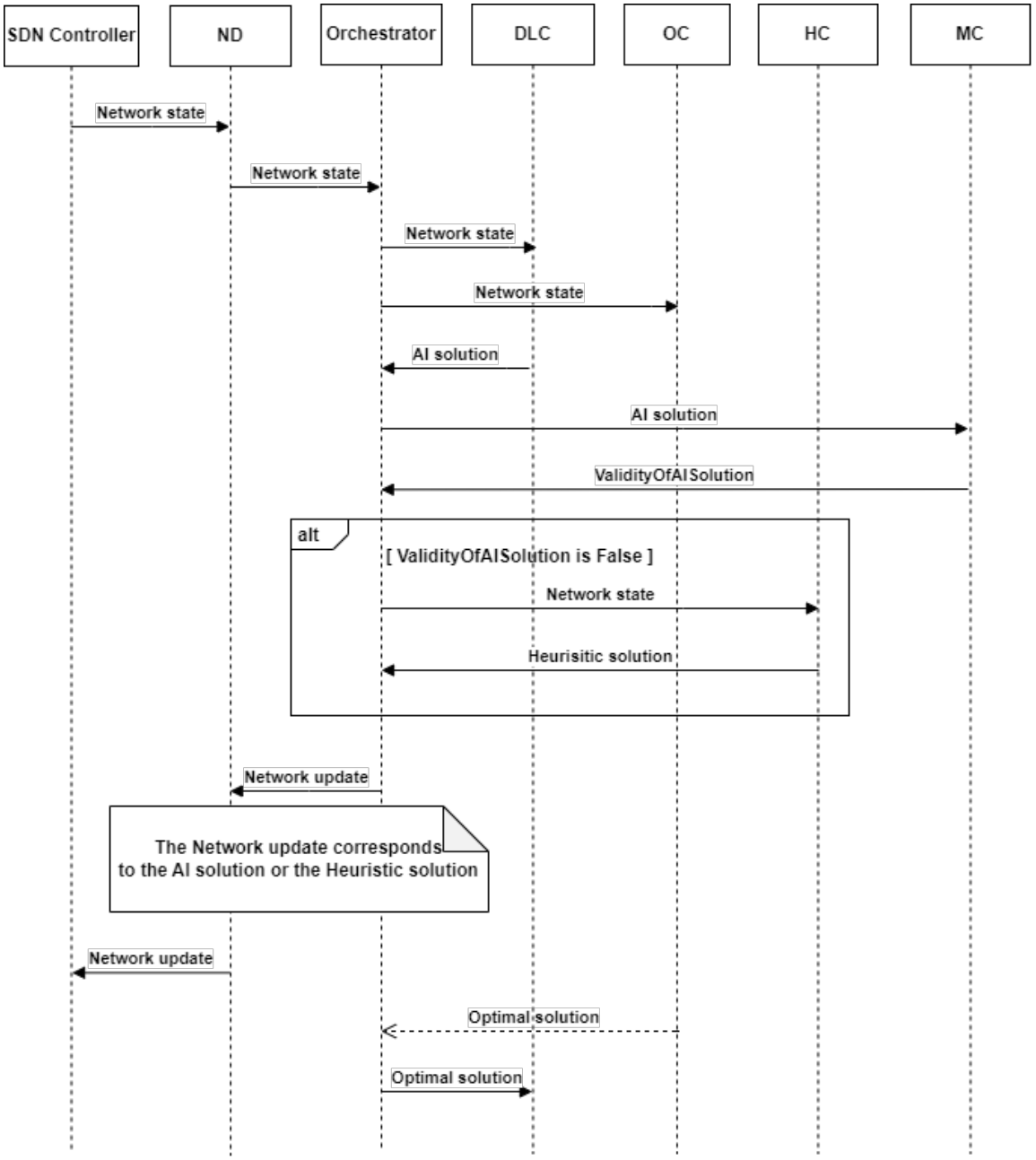


FIGURE 3.2 – Diagramme de séquence du framework DLO-SFC

Dans l'optique de répondre à cette problématique, l'*Orchestrator* établit une interaction intelligente entre les techniques sous-jacentes à l'apprentissage automatique (ML, plus précisément le Deep Learning) et les méthodes d'optimisation, en privilégiant principalement la première de ces modalités et en faisant appel à la seconde en dernier recours, de manière analogue à la fonction d'un "condensateur" (capacitor, en anglais).

L'*Orchestrator* transmet aux composants *DLC* et *OC* l'état du réseau, englobant à la fois la configuration topologique du réseau en question ainsi que les demandes spécifiques aux différents services, comme illustré dans la Figure 3.2. Ces deux composants constitutifs s'engagent dans une forme de compétition pour élaborer et soumettre la configuration la plus optimale, visant à satisfaire au mieux les contraintes inhérentes ainsi qu'aux objectifs préalablement définis.

Le *DLC* utilise son réseau neuronal en vue de la détermination d'une configuration réseau optimale (solution d'intelligence artificielle). Le *DLC* génère d'abord la configuration, procédant ensuite à sa transmission à l'*Orchestrator*, tel qu'illustré dans la Figure 3.2.

À ce stade, l'*Orchestrator* fait appel au ***Model Checker (MC)*** pour vérifier la validité de la configuration ainsi obtenue, comme clairement mis en lumière par la Figure 3.2.

À noter que la vérification de la validité d'une configuration prend généralement un temps polynomial, impliquant une série d'éléments tels que la vérification de la connectivité des chemins générés, l'évitement de la formation de boucles, et la conformité aux indicateurs clés de performance désirés (à savoir le délai et les allocations de bande passante).

Dans l'éventualité où la configuration proposée surmonte avec succès l'évaluation de la validation, l'*Orchestrator* applique les politiques générées via le composant *ND*.

Dans les autres cas, la configuration obtenue n'est pas valide. Cette situation est illustrée par la représentation graphique consignée dans la Figure 3.2. Dans ces circonstances particulières, l'*Orchestrator* peut invoquer l'***Heuristic Component (HC)*** en vue de générer une configuration valide en un temps raisonnable, plutôt que d'attendre passivement que l'*OC* puisse accomplir sa fonction, qui prend habituellement un certain temps.

Le *HC* utilise un algorithme heuristique afin de générer une configuration réseau (solution heuristique). Il fournit une configuration pratiquement optimale dans un délai raisonnable.

Il convient de noter que le *HC* peut également être invoqué en même temps que le *DLC* et l'*OC* afin d'économiser du temps, dans le cas où la configuration générée par le *DLC* se révèle invalide.

Les configurations découlant de l'*OC* sont envisagées pour être incluses au sein du jeu de données (*dataset*), en vue d'améliorer le modèle du *DLC*. Il est à relever que les configurations issues de l'*OC* seront d'abord utilisées dans le cas où elles sont générées dans un intervalle temporel jugé raisonnable.

Cette approche stratégique aide non seulement à l'adoption d'une configuration optimale, mais également à l'atténuation des risques de désynchronisation entre le processus de prise de décision et les changements enregistrés au niveau du plan de l'utilisateur.

La Figure 3.3 présente le diagramme d'activité du framework DLO-SFC. Ce diagramme offre une représentation visuelle du fonctionnement du framework, illustrant les différentes étapes et interactions entre ses composants.

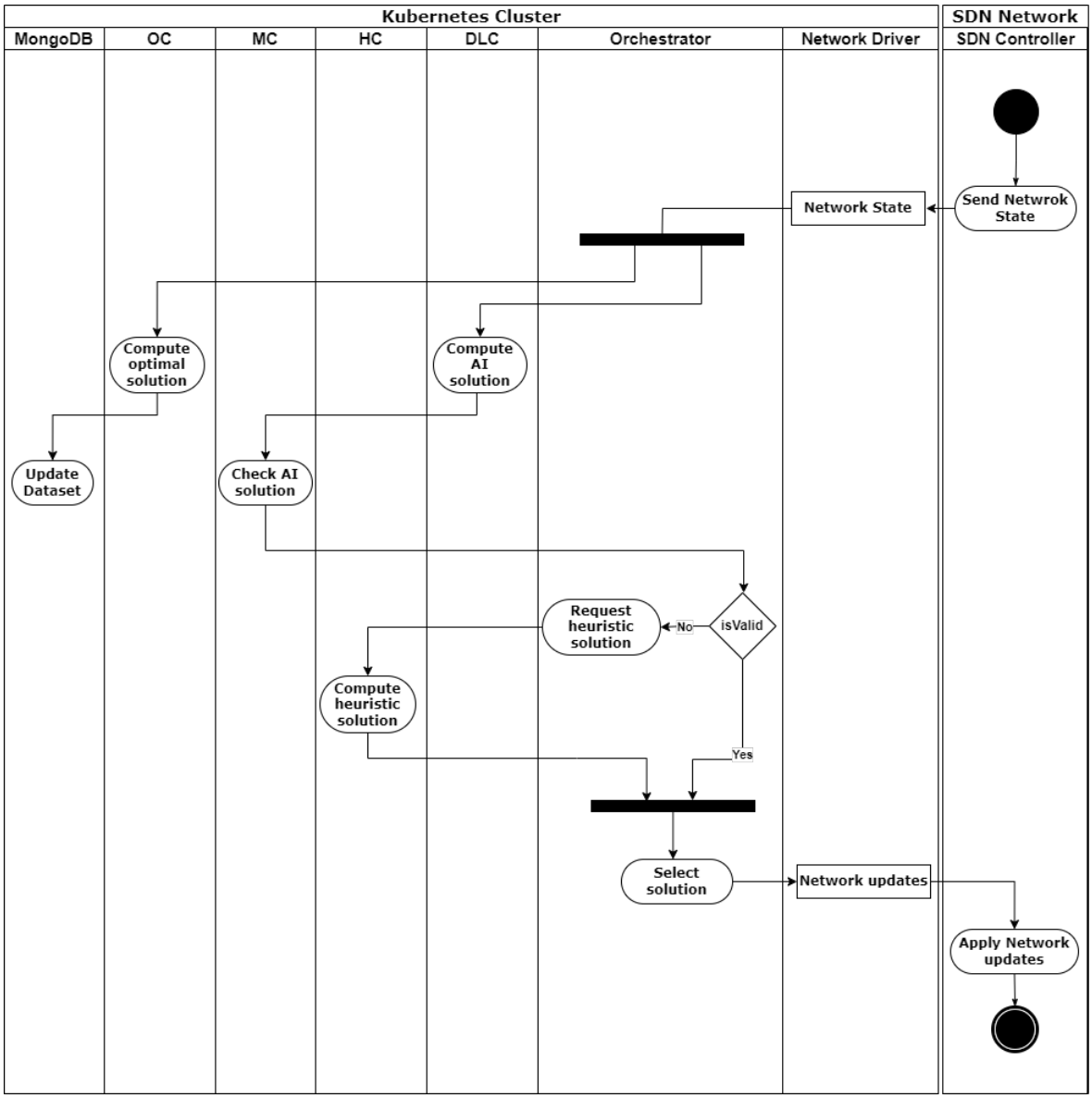


FIGURE 3.3 – Diagramme d'activité du framework DLO-SFC

Le *DLC* utilise un réseau neuronal profond en vue de générer une configuration réseau efficace, laquelle satisfait aux demandes des clients et répond aux contraintes inhérentes à la topologie du réseau.

Cette démarche trouve son fondement dans la modélisation du problème sous la forme d'un modèle de classification, de telle sorte que les étiquettes de classification sont produites au moyen du composant *OC*. En d'autres termes, l'ensemble de données *dataset* est généré en utilisant l'état du réseau reçu, ainsi que la configuration qui a été effectuée par l'*OC*.

Cependant, comme le framework *DLO-SFC* proposé est modulaire, nous avons la possibilité de changer le modèle d'intelligence artificielle utilisé par *DLC* sans altérer la fonctionnalité du framework.

Il convient de mentionner que le *DLC* constitue un champ de recherche important qui nécessite une exploration plus approfondie. À cet effet, une analyse plus détaillée et complète sera menée dans le chapitre suivant, permettant ainsi une meilleure compréhension des mécanismes sous-jacents et des implications pratiques inhérentes à notre approche.

Entre-temps, le *MC* vérifie la validité de la configuration qui a été générée par le *DLC*. Ce processus vise à garantir que la configuration réseau résultante est en parfaite conformité avec les contraintes préétablies ainsi qu'avec les indicateurs de performance (KPI) associés à l'état du réseau correspondant. En effet, une configuration réseau optimale, parfaitement harmonieux avec les impératifs du réseau en question, est acceptée et mise en œuvre par le biais du *ND* en synergie avec le contrôleur SDN en place.

Toutefois, si la validation de cette configuration optimale devait échouer, la stratégie alternative se déploie. Dans cette optique, la configuration générée par le *HC* ou l'*OC* serait appliquée à la place. Cette démarche trouve sa justification dans le souhait de prévenir tout désaccord entre les indicateurs de performance, ou KPIs, et d'assurer une concordance durable entre les paramètres opérationnels et les exigences propres au réseau en question.

Dans le chapitre suivant, nous élaborerons davantage sur le fonctionnement du *MC*. Au sein du framework *DLO-SFC*, nous avons adopté les approches *FPR* et *HPR*, telles que définies par Bagaa et al. [89], pour occuper les rôles respectifs de l'*OC* et du *HC*. Dans le chapitre à venir, une présentation détaillée de ces deux solutions sera effectuée, mettant en lumière leur pertinence dans le contexte de notre étude.

3.3 Déploiement de la plateforme DLO-SFC

Le déploiement de la plateforme DLO-SFC repose sur une architecture micro-services, soigneusement conçue pour garantir une performance optimale. Cette architecture est déployée sur un Cluster Kubernetes, une solution open-source pour l'orchestration de conteneurs. Kubernetes offre une plateforme robuste qui permet le déploiement, le dimensionnement et la gestion efficace des applications conteneurisées [90].

Les contrôleurs SDN peuvent être mis en œuvre grâce à la technologie ONOS (Open Network Operating System), comme décrit dans la Section 1.2.4. ONOS permet la création d'un système d'exploitation de réseau défini par logiciel, offrant ainsi une infrastructure flexible et puissante pour la gestion et le contrôle des réseaux.

Du côté du composant *Network Driver*, le N*MA (Network Monitoring Agent)* est déployé en tant que service Kubernetes reposant sur Prometheus. Ce dernier est un système open-source de surveillance des systèmes informatiques et des infrastructures IT, reconnu pour sa robustesse et sa flexibilité [91].

Pour permettre à Prometheus de surveiller les contrôleurs SDN et leur topologie, deux Jobs doivent être configurés : le Node Exporter et l'ONOS Exporter.

Le Node Exporter agit comme un agent d'exportation pour Prometheus, facilitant la collecte d'informations détaillées sur l'état et les performances des serveurs. Il offre un accès à une multitude de métriques système, telles que l'utilisation du CPU, de la mémoire, du réseau et du stockage [91].

Tandis que l'ONOS Exporter permet à Prometheus de collecter des métriques à partir du contrôleur ONOS, extrayant ainsi des informations spécifiques relatives aux contrôleurs SDN, comme la topologie du réseau et d'autres données pertinentes, offrant ainsi un aperçu détaillé de l'état et des performances de l'infrastructure SDN. [92].

Le module d'apprentissage profond *DLC (Deep Learning Component)* est conçu sous la forme d'un service basé sur le framework d'intelligence artificielle TensorFlow. TensorFlow

est une bibliothèque open-source, initialement développée par Google, spécialisée dans la construction et l'entraînement de modèles d'apprentissage automatique (ML) [93].

Le validateur de modèle *MC (Model Checker)* est un service implémenté en Python [1]. Son rôle est d'implémenter l'algorithme de vérification du résultat fourni par le service *DLC*.

En d'autres termes, le *Model Checker* prend en charge la validation des résultats produits par le module d'apprentissage profond (*DLC*) en utilisant l'algorithme spécifié. En utilisant Python, le *Model Checker* peut bénéficier de la flexibilité et de la richesse des bibliothèques disponibles dans cet écosystème pour effectuer efficacement la vérification des modèles.

Le module d'optimisation *OC (Optimisation Component)* est un autre service du Cluster Kubernetes. Il repose sur le solveur Gurobi Optimizer, une puissante solution d'optimisation mathématique. Gurobi Optimizer est spécialement conçu pour résoudre efficacement des problèmes d'optimisation complexes dans divers domaines d'application [94].

En utilisant ce solveur, le module *OC* peut analyser et résoudre des problèmes d'optimisation avancés, ce qui lui permet d'optimiser les performances et les résultats dans le cadre du déploiement de la plateforme DLO-SFC.

De même, l'*OC*, s'appuie sur le système de gestion de base de données MongoDB pour assurer la conservation et la gestion des informations concernant les configurations produites. Ces données, issues des divers processus de génération, sont essentielles pour enrichir le jeu de données (dataset). MongoDB, en tant que base de données NoSQL orientée document, offre une architecture flexible et évolutive, parfaitement adaptée à la nature variée et en constante évolution des configurations générées par l'*OC* [95].

Pour la génération des solutions heuristiques, le module d'optimisation heuristique *HC (Heuristic Component)* est déployé sous la forme d'un service basé sur la bibliothèque Python NetworkX. NetworkX est une librairie Python open source utilisée pour l'analyse et l'étude des graphes et des réseaux [96]. En utilisant NetworkX, le module *HC* peut résoudre des problèmes d'optimisation de manière efficace et robuste.

La Figure 3.4 offre un aperçu détaillé des technologies essentielles employées pour la mise en œuvre réussie de la plateforme DLO-SFC sur un Cluster Kubernetes, tout en intégrant étroitement des contrôleurs SDN basés sur la technologie ONOS.

Cette représentation visuelle met en lumière l'écosystème complexe et interconnecté de solutions logicielles et d'infrastructures qui sont nécessaires pour garantir le bon fonctionnement de la plateforme.

1. https ://www.python.org/

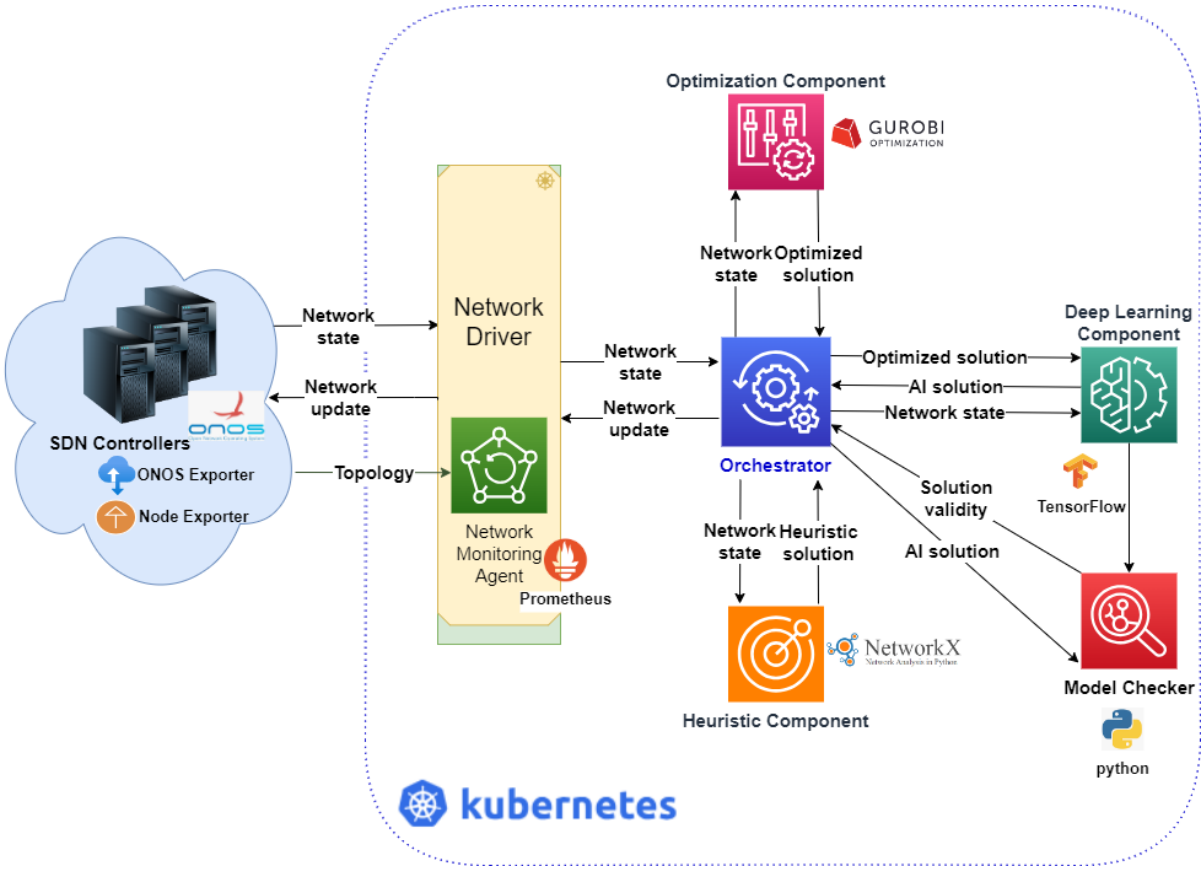


FIGURE 3.4 – Déploiement de la plateforme DLO-SFC dans un Cluster Kubernetes

3.4 Conclusion

En conclusion, ce chapitre a présenté en détail le framework d'optimisation basé sur l'IA, baptisé *DLO-SFC*, conçu pour répondre aux défis complexes posés par les réseaux cellulaires émergents, notamment dans le contexte des technologies 5G et 6G.

À travers une approche originale intégrant les principes de l'apprentissage profond, de l'optimisation et des heuristiques, le *DLO-SFC* offre une solution complète pour l'automatisation en boucle fermée, permettant une configuration efficace du réseau et une gestion autonome des services.

L'adoption du concept ETSI ZSM dans le cadre du *DLO-SFC* souligne l'importance accordée à l'automatisation et à l'efficacité opérationnelle, contribuant ainsi à la réduction des coûts et de l'empreinte carbone associés aux opérations réseau.

Cette approche s'aligne également avec les impératifs de Green IT, en minimisant l'utilisation d'énergies fossiles pour alimenter les centres de données.

Le déploiement pratique du *DLO-SFC* sur une architecture micro-services basée sur Kubernetes, intégrant des technologies telles que les contrôleurs SDN ONOS, le système de surveillance Prometheus, TensorFlow pour l'intelligence artificielle, Gurobi Optimizer pour l'optimisation mathématique, MongoDB pour la base de données et NetworkX pour les heuristiques, démontre la viabilité et l'efficacité de cette solution dans des environnements réels.

Dans le chapitre suivant, nous approfondirons le fonctionnement du module (*MC*) et nous présenterons de manière détaillée les approches *Full Paths Recomputation (FPR)* et *Heuristic Paths Recomputation (HPR)*, telles que définies par Bagaa et al. [89]. Ces approches occupent respectivement les rôles de l'*OC* et du *HC* au sein du framework *DLO-SFC*. Ceci mettra en lumière notre étude en exposant les mécanismes sous-jacents qui régissent le processus d'optimisation dans le contexte du *DLO-SFC*.

En résumé, le framework d'optimisation basé sur l'IA offre une approche holistique et originale pour répondre aux défis complexes des réseaux mobiles modernes, ouvrant la voie à une gestion autonome, efficace et durable des infrastructures réseau et des services associés.

Chapitre 4

Routage multi-chemins intelligent

La réussite, c'est un peu de savoir, un peu de savoir-faire et beaucoup de faire-savoir.

Jean Nohain (1900-1981)

4.1 Introduction

Dans le cadre des réseaux mobiles 5G/6G, la connectivité revêt une importance capitale en interconnectant divers clients aux serveurs fournissant des services 5G/6G, grâce à l'utilisation de switchs virtuels. Cette configuration complexe englobe une variété d'éléments tels que des clients, des switches virtuels et des serveurs, chacun jouant un rôle crucial dans la mise en œuvre efficace de ces réseaux.

Toutefois, afin d'optimiser les coûts opérationnels (OpEx), il est essentiel de désactiver les switchs virtuels inutilisés. Les opérateurs disposant de réseaux définis par logiciel (SDN) peuvent considérablement améliorer leur efficacité opérationnelle tout en réduisant les coûts OpEx. Cela peut être réalisé en minimisant le nombre de switchs virtuels activés, tout en veillant à maintenir les performances requises.

En utilisant des techniques telles que le routage multi-chemins intelligent, les opérateurs peuvent efficacement allouer les ressources du réseau et réduire considérablement le surprovisionnement en n'activant que les switchs virtuels nécessaires. Cette approche offre une solution efficace pour optimiser l'utilisation des ressources réseau tout en garantissant une qualité de service (QoS) optimale.

Dans ce chapitre, nous explorons divers aspects du routage multi-chemins intelligent, en mettant particulièrement l'accent sur les solutions basées sur le Software-Defined Networking (SDN) et l'intelligence artificielle (IA).

La première section examine l'optimisation du routage multi-chemins dans les environnements SDN. Nous présentons les travaux de Bagaa et al. [89], qui ont proposé des stratégies exploitant les fonctionnalités du SDN pour rationaliser l'allocation des ressources réseau tout en maintenant les garanties de qualité de service (QoS).

A cet effet, nous explorons deux approches proposées, à savoir *Full Paths Recomputation (FPR)* et *Heuristic Paths Recomputation (HPR)*, qui offrent un équilibre entre la précision des résultats et les contraintes de temps.

La deuxième section se concentre sur l'application pratique du concept d'intelligence artificielle dans le routage multi-chemins, en introduisant le modèle *iPare (Intelligent Path Re-computation)* dans le cadre du framework DLO-SFC. Nous décrivons en détail la structure et le fonctionnement de ce modèle, ainsi que son interaction avec d'autres composants du framework.

Nous examinons également le processus d'entraînement du modèle *iPare* et son utilisation pour générer des configurations optimales de chemins de transmission dans les réseaux SDN.

Enfin, dans la troisième section, nous explorons la mise en œuvre concrète du routage multi-chemins dans la plateforme DLO-SFC. Nous présentons l'architecture de la plateforme, organisée en différents espaces de noms, et examinons comment elle peut être déployée pour prendre en charge le routage multi-chemins en utilisant la solution *iPare*.

Nous mettons en lumière les interactions entre les différents composants de la plateforme DLO-SFC et discutons de leur rôle dans la réalisation d'un routage multi-chemins efficace et adaptatif.

En résumé, ce chapitre offre un aperçu des différents aspects du routage multi-chemins intelligent, en mettant en avant son importance et ses applications dans les réseaux informatiques contemporains.

4.2 Optimisation du routage multi-chemins dans les SDN

Dans leur étude, Bagaa et al [89] ont proposé une stratégie qui exploite les fonctionnalités du Software-Defined Networking (SDN) afin de rationaliser l'utilisation des ressources réseau. Cette approche vise à réduire le nombre de switchs virtuels activés tout en optimisant les temps d'exécution. En outre, elle intègre une considération essentielle pour les performances du réseau en assurant le maintien des garanties de qualité de service.

Pour ce faire, la solution qu'ils proposent exploite la transmission par multi-chemins dans un environnement de réseau mobile 5G. Les auteurs ont proposé deux solutions : *Full Paths Recomputation (FPR)* et *Heuristic Paths Recomputation (HPR)* pour configurer l'ensemble de l'allocation des ressources du réseau lors de la réception d'une nouvelle demande de service utilisateur ou d'un mouvement de l'utilisateur.

La première solution, appelée FPR, permet de déterminer une configuration optimale en s'appuyant sur une exploration de la programmation linéaire en nombres entiers. Cette approche implique une recherche méthodique de la solution optimale, assurant ainsi une configuration adaptée aux exigences du réseau.

D'autre part, la seconde approche, appelée HPR, adopte une perspective différente en abordant un problème d'optimisation approximative. Cette approche est motivée par la nécessité de limiter le temps de calcul requis pour déterminer une configuration, en exploitant l'algorithme du chemin le plus court de Dijkstra. Elle offre ainsi un équilibre judicieux entre la précision des résultats obtenus et les contraintes de temps inhérentes au processus de calcul.

4.2.1 FPR : Full Paths Re-computation

La méthode FPR (Recalcul complet des chemins, acronyme anglais pour *Full Paths Re-computation*) vise à optimiser la configuration des chemins de transmission dans un réseau virtualisé utilisant des OVSs (Open vSwitch) (voir Section 1.2.4).

Son principal objectif est de fournir une configuration optimale en prenant en compte l'ensemble des utilisateurs. À cette fin, elle se base sur une matrice de variables entières représentant le trafic généré et transmis vers chaque serveur. Cette matrice est essentielle pour déterminer la répartition optimale des flux de données entre les clients et les serveurs.

De manière plus formelle, soit C, O et S qui représentent respectivement l'ensemble des clients, des switchs virtuels (OVSs) et des serveurs du réseau. Pour chaque serveur $s \in S$, la solution FPR définit une matrice F^s de variables entières qui représente le trafic généré et transmis vers ce serveur. Chaque élément $F^s_{i,j}$ de cette matrice représente la quantité de flux devant être transmise de l'entité $i \in C \cup O$ (client ou OVS) vers l'entité $j \in O \cup S$ (OVS ou serveur).

L'approche FPR s'inscrit dans le cadre d'une méthodologie de résolution de problèmes d'optimisation. Au cœur de cette méthodologie réside l'utilisation de la technique appelée *branch and bound*, ou en français *séparation et évaluation*.

En outre, l'approche FPR vise à parvenir à une solution optimale pour un problème d'optimisation spécifique, à savoir la minimisation du coût total de transmission de l'ensemble des flux circulant au sein du réseau, tout en préservant le respect des contraintes de bande passante imposées par chaque client du réseau. Il convient de noter que le coût global de transmission est défini comme la somme agrégée des coûts de transmission associés à chacun des flux au sein du réseau.

La solution FPR utilise également des contraintes visant à assurer que chaque client bénéficie de la bande passante nécessaire pour ses flux. Ces contraintes sont formulées en tenant compte des besoins de bande passante spécifiques de chaque client, ainsi que de la capacité de bande passante disponible sur chaque chemin de transmission.

À chaque exécution, la solution FPR calcule une nouvelle configuration des chemins de transmission en se basant sur la matrice $F^s_{i,j}$ ainsi que sur les contraintes de bande passante. Cette configuration actualisée est ensuite mise en œuvre pour le transfert des flux au sein du réseau.

En résumé, la solution FPR représente une approche visant à optimiser la configuration des chemins de transmission dans un réseau virtualisé basé sur SDN. Elle se sert d'une matrice de variables entières pour déterminer le nombre de flux qui doivent être transmis de chaque client vers chaque serveur, tout en exploitant la méthode de séparation et d'évaluation *"branch and bound"*.

4.2.2 HPR : Heuristic Paths Re-computation

Considérons un graphe pondéré $G(V, E, W)$, où V représente un ensemble de nœuds et E l'ensemble des arêtes du réseau. L'ensemble V est formalisé comme suit :

$$V = C \cup O \cup S$$

où C, O et S représentent respectivement l'ensemble des clients, l'ensemble des switchs virtuels (OVSs) et l'ensemble des serveurs du réseau. En outre, chaque arête est assortie d'un poids W, noté $W_{u,v}$, où (u, v) est une arête, et il symbolise la capacité de la bande passante entre les nœuds u et v. Il convient de souligner que (u, v) est un élément de l'ensemble E, soit $(u, v) \in E$.

La méthode HPR *(Recalcul des Chemins Heuristiques, ou bien Heuristic Paths Recomputation en anglais)* est une approche visant à optimiser la configuration des chemins de transmission dans un réseau virtualisé en utilisant un algorithme heuristique pour déterminer les chemins de transmission optimaux.

Plus formellement, HPR utilise en entrée un graphe pondéré $G(V, E, W)$, C et S, tandis que ses sorties se composent des OVS activés Φ et des chemins générés P.

L'algorithme HPR repose sur l'utilisation de l'algorithme de Dijkstra pour déterminer les chemins de transmission optimaux entre les clients et les serveurs. L'algorithme de Dijkstra est un algorithme de recherche de plus court chemin qui calcule le trajet le plus court entre un nœud source et tous les autres nœuds dans un graphe pondéré. Dans le cas de la solution HPR, le graphe est pondéré en fonction de la bande passante disponible sur chaque chemin de transmission.

L'algorithme HPR commence par initialiser un ensemble vide de switchs virtuels (OVS) sélectionnés, soit $\Phi = \emptyset$. Il entre ensuite dans une boucle pour calculer les chemins de transmission optimaux entre chaque client c ($c \in C$) et chaque serveur S_c. Pour chaque client c, l'algorithme HPR détermine le chemin le plus court vers chaque serveur S_c en utilisant l'algorithme de Dijkstra. Ensuite, il choisit le chemin de transmission optimal en fonction de la bande passante disponible sur chaque chemin.

Une fois que tous les chemins de transmission optimaux sont calculés, l'algorithme HPR sélectionne les OVS nécessaires pour acheminer les flux sur ces chemins. Cette sélection se base sur la bande passante disponible sur chaque chemin et sur la capacité de traitement de chaque OVS.

Enfin, l'algorithme HPR génère une nouvelle configuration des chemins de transmission en utilisant les OVS sélectionnés et les chemins de transmission optimaux. Cette nouvelle configuration est ensuite utilisée pour le transfert des flux dans le réseau.

Il est à mentionner que l'objectif principal de l'algorithme HPR réside dans la maximisation de la réutilisation des OVS activés, de l'ensemble Φ, dans le but de réaliser des économies significatives au niveau des dépenses d'exploitation (OpEx), tout en préservant et assurant la qualité de service (QoS) au sein du réseau.

Dans cette algorithme, un nouveau graphe est créé, noté $G'(V', E', W')$, qui prend forme à partir du graphe initial $G(V, E, W)$ en suivant une procédure de sélection des OVS à partir de l'ensemble Φ, dans le but de les utiliser pour l'interconnexion des clients c avec les serveurs s.

Cette opération importante vise à optimiser l'utilisation des ressources disponibles en intégrant les OVS appropriés dans le réseau, tout en évitant une redondance inutile des éléments déjà activés.

La solution HPR a été élaborée pour être exécutée de manière périodique ou en réponse aux changements significatifs dans la charge du réseau. Lorsque l'approche périodique est privilégiée, son exécution est prévue à une fréquence d'environ 10 MHz, mais elle peut également être déclenchée au minimum toutes les 100 ms. Elle est également conçue pour

s'opérer en arrière-plan, afin de ne pas affecter la qualité de service (QoS) au niveau de la couche de données.

En résumé, la solution HPR est une méthode visant à optimiser la configuration des chemins de transmission dans un réseau virtualisé en utilisant un algorithme heuristique pour déterminer les chemins de transmission optimaux. Elle se base sur l'algorithme de Dijkstra pour calculer ces chemins, sélectionne les OVS nécessaires pour le transfert des flux, et peut s'exécuter périodiquement ou en cas de modification importante de la charge du réseau, tout en préservant la qualité de service (QoS) au niveau de la couche de données.

4.3 iPare : Intelligent Path Re-computation

Le framework *DLO-SFC*, tel que proposé dans la Section 3.2, se caractérise par son caractère générique et modulaire, offrant ainsi la souplesse de mettre en œuvre ou de modifier ses composants sans altérer la fonctionnalité fondamentale du système.

La présente section se consacre à la présentation détaillée du fonctionnement et de l'interaction entre le *MC* (Model Checker) ainsi que du modèle d'intelligence artificielle désigné sous le nom d'***iPare*** *(Re-calcul de Chemin Intelligent)*, tous deux utilisés par le *DLO-SFC*.

Cette présentation vise à offrir une appréciation approfondie des éléments constitutifs du framework *DLO-SFC*, lesquels agissent en synergie pour répondre aux exigences et aux défis des réseaux définis par logiciel (SDN) en conjonction avec les capacités de l'intelligence artificielle.

Dans un premier temps, nous introduisons iPare, modèle d'intelligence artificielle novateur destiné à fournir des configurations de chemins multiples optimales pour les réseaux pilotés par SDN. iPare représente une avancée significative dans la gestion dynamique des flux de données au sein de ces réseaux.

En d'autres termes, iPare est considéré comme un cas particulier de la fonction du module générique DLC (*Deep Learning Component*) présenté dans le Chapitre 3, plus précisément dans le contexte des switchs virtuels OvS.

Cependant, iPare se distingue par sa capacité à fournir des solutions pratiques et performantes, ajustées aux spécificités de chaque réseau. Cette capacité d'adaptation est essentielle pour garantir une gestion efficiente des flux de données dans des contextes variables.

Par la suite, nous abordons le *MC* (*Model Checker*), un outil essentiel pour évaluer la faisabilité des configurations générées par iPare. Le *MC* agit comme un mécanisme de validation, s'assurant que les configurations de chemins multiples préconisées par iPare sont compatibles avec les contraintes et les spécifications du réseau. Ainsi, cette composante complémentaire renforce la robustesse et la crédibilité de la proposition iPare en assurant que les solutions configurées sont non seulement optimales, mais également pratiquement réalisables.

En somme, cette section présente deux éléments fondamentaux du framework *DLO-SFC*. iPare incarne la convergence de l'intelligence artificielle et des réseaux SDN, en proposant une approche originale pour configurer les chemins multiples de manière optimale. Le *MC*, quant à lui, exerce un rôle de contrôle et de validation, assurant la robustesse et la validité pratique des configurations préconisées.

Ces éléments, pris dans leur ensemble, contribuent à une vision cohérente et équilibrée du framework *DLO-SFC*, enrichissant ainsi la compréhension de son fonctionnement et de ses avantages potentiels dans le contexte en constante évolution des réseaux informatiques modernes.

Enfin, les notations pertinentes, utilisées pour décrire de manière cohérente la solution proposée, sont récapitulées dans le Tableau 4.1.

TABLE 4.1 – Synthèse des notations utilisées

Notation	**Description**	**Notation**	**Description**
DLO-SFC	Deep Learning Optimization for Service Function Chaining, le framework proposé.	ND	Network Driver, assure la communication entre le framework DLO-SFC et l'environnement externe, constitué notamment des contrôleurs SDN.
Orchestrator	Composant central du DLO-SFC, assure une interaction intrinsèque avec l'ensemble des éléments du framework.	NMA	Network Monitoring Agent, est une fonctionnalité conçue pour surveiller à la fois le réseau et les changements dans sa topologie.
DLC	Deep Learning Component, utilise son réseau neuronal pour calculer une configuration optimale du réseau, offrant une solution basée sur l'IA.	HC	Heuristic Component, génère une configuration valide dans un délai raisonnable en utilisant un algorithme heuristique.
MC	Model Checker, l'algorithme chargé de vérifier la validité de la solution proposée par le DLC (solution IA).	OC	Optimisation Component, génère une configuration réseau optimisée en utilisant un algorithme d'optimisation.
FPR	Full Paths Re-computation, l'algorithme qui fournit une configuration optimale en explorant la programmation linéaire en nombres entiers.	HPR	Heuristic Paths Re-computation, l'algorithme qui permet une configuration réseau efficace en exploitant l'algorithme du plus court chemin de Dijkstra.
iPare	Intelligent Path Re-computation, la solution utilisée par le framework DLC-SFC pour calculer les chemins optimaux en utilisant l'apprentissage profond et la classification multi-label.	Mc-DLN	Multi-label Classification Deep Learning Network, est un réseau neuronal qui spécifie quels switchs virtuels doivent être activés.
ActiVect	Un vecteur binaire généré par le Mc-DLN et qui spécifie l'état de chaque switch virtuel dans le réseau.	NCG	Network Configuration Generator, est un composant de l'iPare qui utilise le tableau ActiVect généré par le composant Mc-DLN pour changer l'état de chaque switch virtuel dans le réseau.

4.3.1 Présentation d'iPare

Le framework DLO-SFC fait appel à la solution iPare pour calculer les chemins optimaux en exploitant l'apprentissage profond (*Deep Learning*) et la classification multi-étiquettes.

Avant de plonger dans les détails de la solution iPare, il est crucial de comprendre sa structure fondamentale. Divisée en deux parties distinctes, cette solution combine habilement deux éléments essentiels : d'une part, le réseau d'apprentissage profond pour la classification multi-étiquettes, appelé ***Mc-DLN*** pour *Multi-label Classification Deep Learning Network*, et d'autre part, le générateur de configuration réseau, connu sous le nom de ***NCG*** pour *Network Configuration Generator*.

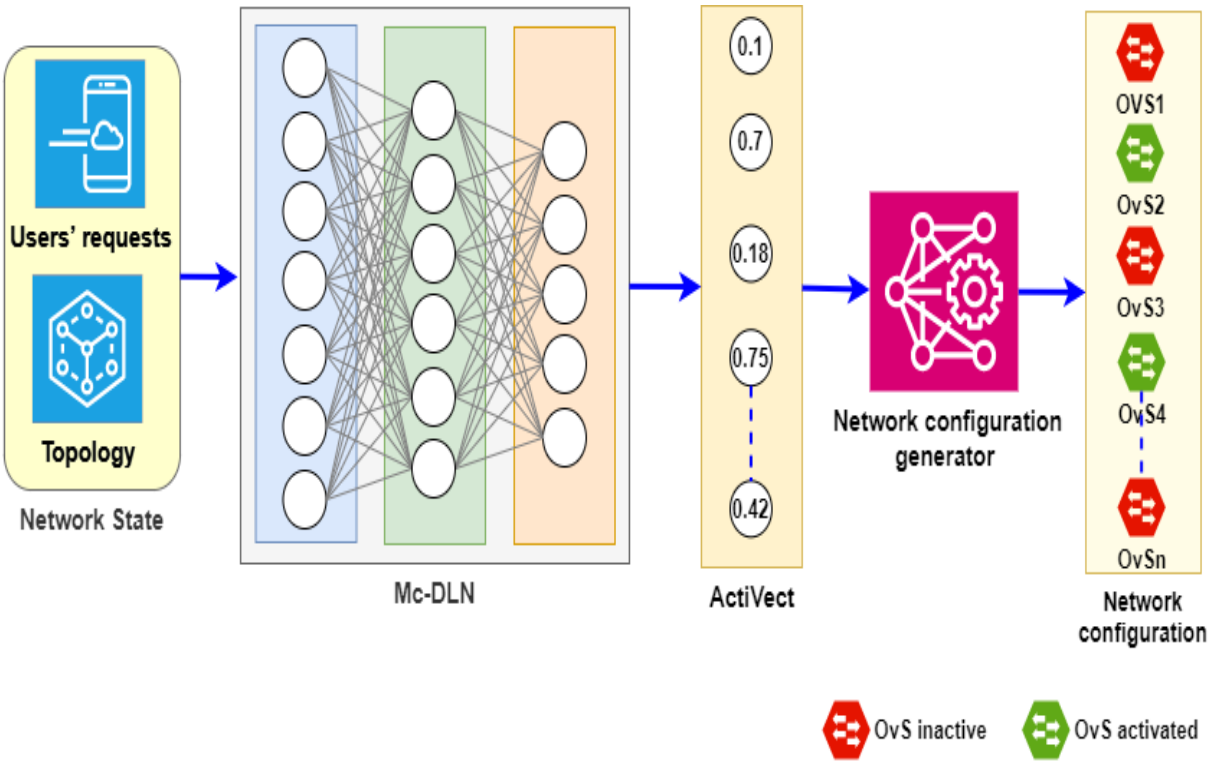


FIGURE 4.1 – Architecture d'*iPare*

La Figure 4.1 offre une représentation visuelle de la composition de la solution proposée. Cette segmentation en deux composants clés illustre la nature complexe et interconnectée de la solution *iPare*, qui exploite ces éléments de manière synergique pour répondre aux défis qui se présentent.

Le résultat de Mc-DLN est la configuration du réseau présentée sous la forme d'un vecteur d'activation (ActiVect). Chaque sortie dans ActiVect représente un switch virtuel dans le réseau. L'ActiVect sera exploité par NCG pour générer une configuration optimale à chemins multiples en spécifiant les switchs virtuels Open vSwitch (OvS) qui doivent être activés.

En effet, les switchs virtuels qui doivent être activés sont ceux dont les valeurs dans ActiVect dépassent un seuil prédéfini ϵ. L'augmentation de la valeur de ϵ entraîne une réduction du nombre de switchs virtuels qui doivent être activés, ce qui a un impact positif sur le coût.

Cependant, l'augmentation de ϵ a un impact négatif sur le nombre de configurations valides garantissant l'objectif de niveau de service *SLA* souhaité (*Service Level Agreement*). Il existe un compromis entre la précision et le coût. La valeur de ϵ représente un hyperparamètre crucial à définir lors du processus d'entraînement.

Il convient de noter que la détermination des valeurs seuils ϵ représente un élément critique dans le déploiement de la solution iPare. Un choix judicieux de cette valeur permettra d'optimiser la performance en fonction des exigences spécifiques du réseau et des objectifs de service. Cette considération souligne la nécessité d'une approche méthodique lors de la configuration de l'iPare et de ses composants associés.

Le modèle *Mc-DLN* est constitué d'une couche d'entrée, d'un ensemble de couches cachées et d'une couche de sortie. Alors que la fonction d'activation ReLu (Rectified Linear Unit) est utilisée dans la couche d'entrée et les couches cachées, la couche de sortie utilise

la fonction d'activation *Sigmoïd*.

Le modèle iPare prend en entrée l'état du réseau, qui comprend les requêtes des utilisateurs ainsi que la topologie du réseau. En sortie, il produit la configuration du réseau sous la forme d'un ActiVect. Ce dernier se présente comme un vecteur de neurones, où chacun correspond à un switch virtuel et contient une valeur comprise entre 0 et 1.

Mc-DLN représente un problème de classification multi-étiquettes qui permet la sélection de plusieurs switchs virtuels. Le rôle du NCG est crucial dans ce processus, car il intervient pour activer spécifiquement les switchs virtuels dont les valeurs associées dans ActiVect surpassent le seuil défini ϵ.

Le modèle est entraîné sur la base de l'ensemble de données généré par l'OC du framework DLO-SFC. L'OC utilise l'état du réseau, comprenant les demandes du réseau et sa topologie, en tant qu'entrée, puis génère les chemins optimaux en activant les switchs virtuels nécessaires.

Dans cette dynamique, l'iPare exploite l'ensemble de données généré par l'OC pour entraîner le modèle Mc-DLN. L'ensemble de données généré comprend : $i)$ les données d'entrée, représentant l'état du réseau, comprenant les requêtes des utilisateurs et la topologie du réseau ; $ii)$ les données de sortie, qui sont les étiquettes, comprenant le vecteur ActiVect dont les valeurs sont égales à 1 pour les switchs virtuels activés.

Cet ensemble de données (dataset) contribue à l'amélioration des performances du modèle Mc-DLN en facilitant la génération de configurations réseau optimales. Chaque entrée dans l'ensemble de données spécifie les chemins optimaux, composés des switchs virtuels OvS à activer en fonction de l'état du réseau.

À cet égard, le framework DLO-SFC s'engage dans une démarche continue d'amélioration du modèle Mc-DLN en exploitant les nouvelles sorties générées par l'OC.

Afin de représenter de manière adéquate les informations historiques du réseau générées par l'OC, nous avons puisé notre inspiration dans la méthode de représentation des images en vision par ordinateur, où les images sont structurées en un ou plusieurs canaux.

Cette approche est familière dans le domaine de la vision par ordinateur, où les images en niveaux de gris sont encodées en un canal unique, tandis que les images en couleur sont composées de trois canaux distincts correspondant aux couleurs primaires : Rouge, Vert et Bleu.

Dans le cadre du framework DLO-SFC, nous utilisons un système de représentation des informations réseau à travers trois canaux distincts :

— le Canal de Capacité, *Capacity Channel* (***CC***) en anglais,
— le Canal de Requête, *Request Channel* (***RC***) en anglais,
— le Canal d'Allocation, *Allocation Channel* (***AC***) en anglais.

Dans chaque canal, nous avons considéré une matrice de dimensions $\mathcal{N} \times \mathcal{N}$, où $\mathcal{N}$ représente le nombre de clients, de switchs virtuels et de serveurs.

Pour clarifier davantage, notons $\mathcal{C}$, $\mathcal{O}$ et $\mathcal{S}$ comme les dénominations respectives du nombre de clients, de switchs virtuels et de serveurs.

En termes formels, nous pouvons exprimer ceci par l'équation 4.1.

$$\mathcal{N} = \mathcal{C} + \mathcal{O} + \mathcal{S} \tag{4.1}$$

Le Canal de Capacité (CC) représente la topologie du réseau, définissant les liaisons de communication et leurs capacités entre les clients, les switchs virtuels et les serveurs. En revanche, le Canal de Requête (RC) spécifie les services demandés par chaque client, identifiant ainsi les serveurs requis.

En parallèle, le RC précise différentes requêtes incluant la bande passante souhaitée par chaque client et ses serveurs respectifs. Cette configuration du réseau, combinant à la fois le CC et le RC, sert de couche d'entrée pour le modèle Mc-DLN.

En effet, tel que mentionné précédemment, le modèle Mc-DLN continue son processus d'apprentissage de façon continue grâce à l'OC, suite à la réception des informations de topologie par le framework DLO-SFC via son ND.

En conséquence, le framework DLO-SFC calcule et envoie la configuration réseau optimale correspondant à l'état du réseau reçu au contrôleur SDN via l'interface ND. À la réception réussie de la configuration réseau via les API northbound du contrôleur SDN, ce dernier met en application les politiques reçues sur le réseau pour répondre aux exigences souhaitées.

La sortie du modèle Mc-DLN consiste en des décisions, c'est-à-dire des mises à jour réseau, qui doivent être envoyées au contrôleur SDN. De manière complémentaire, ces mises à jour du réseau sont présentées sous la forme d'un Canal d'Allocation (AC), un vecteur de dimension $\mathcal{O}$ ayant pour rôle de déterminer si les switchs virtuels doivent être activés ou non.

Cette activation est motivée par la nécessité de respecter les indicateurs de performance *KPIs* souhaités, tout en optimisant la gestion des coûts et de l'énergie.

En effet, le canal AC est principalement constitué de zéros, sauf pour les switchs virtuels qui doivent être activés pour assurer un service de qualité conforme aux attentes. De cette manière, le contrôleur SDN activera les switchs virtuels ayant une valeur de 1, tout en désactivant les autres switchs.

Ce processus garantit une communication fluide entre les clients et leurs serveurs respectifs, assurant ainsi une expérience utilisateur optimale et conforme aux exigences opérationnelles.

4.3.2 Entraînement du modèle *Mc-DLN*

Dans le cadre du processus d'entraînement, le modèle *Mc-DLN* considère un ensemble de données (dataset) composé de $\mathcal{M}$ informations réseau constituées des trois canaux.

Avant de procéder à la formation proprement dite du modèle *Mc-DLN*, une étape préliminaire est réalisée par le modèle *iPare*. Ce dernier se charge de transformer les données provenant des trois canaux en un ensemble étiqueté contenant $\mathcal{M}$ éléments provenant d'un seul canal.

En effet, pour chaque information réseau, nous générons le canal d'état du réseau (*Network State Channel (NSC)).* En premier lieu, le *NSC* est initialisé avec les valeurs du *RC*. Ensuite, nous mettons à jour le *NSC* en additionnant ses valeurs avec celles du *CC* un par un.

Il est important de noter qu'il n'y a pas de chevauchement entre le *RC* et le *CC*. Les seules cellules qui pourraient différer de zéro dans le *RC* sont représentées par $RC_{i,j}$, indiquant la requête du client i envers le serveur j, tel que $i \in \mathcal{C}$ et $j \in \mathcal{S}$. Formellement, cela se traduit par :

$$\forall i \notin \mathcal{C} \vee \forall j \notin \mathcal{S} \implies RC_{i,j} = 0 \tag{4.2}$$

Pendant ce temps, le *CC* affiche les liens de capacité entre les éléments du réseau dans l'ensemble $\mathcal{C} \cup \mathcal{O} \cup \mathcal{S}$.

En se basant sur l'observation qu'il n'existe pas de lien direct entre les clients et les serveurs, il en découle qu'il n'y a pas de chevauchement entre le *RC* et le *CC*. Formellement, cela se traduit par :

$$\forall i \in \mathcal{C} \wedge \forall j \in \mathcal{S} \implies CC_{i,j} = 0 \tag{4.3}$$

Le processus de transformation des canaux a pour but de simplifier la structure des données et de créer un ensemble de données étiquetées qui peut être utilisé pour l'apprentissage supervisé du modèle *Mc-DLN*.

En effet, en transformant les informations contenues dans les trois canaux en un seul canal, nous réduisons la complexité du problème d'apprentissage. Cette approche peut conduire à une meilleure généralisation du modèle et à des performances améliorées.

L'approche de fusion des canaux permet de tirer parti des caractéristiques spécifiques de chaque canal, tout en conservant une structure de données cohérente pour l'entrée du modèle *Mc-DLN*.

L'absence de chevauchement entre les canaux *RC* et *CC* découle des caractéristiques inhérentes à la topologie du réseau. Les choix de conception tels que l'absence de liens directs entre les clients et les serveurs, ainsi que la distribution spécifique des capacités entre les éléments du réseau, sont reflétés dans cette distinction.

Cette distinction revêt une importance cruciale pour l'exactitude des données utilisées dans le processus d'apprentissage du modèle *Mc-DLN*, garantissant ainsi que les informations pertinentes sont correctement intégrées dans le modèle. Cela contribue à améliorer la prise de décision lors de la génération de configurations optimales du réseau.

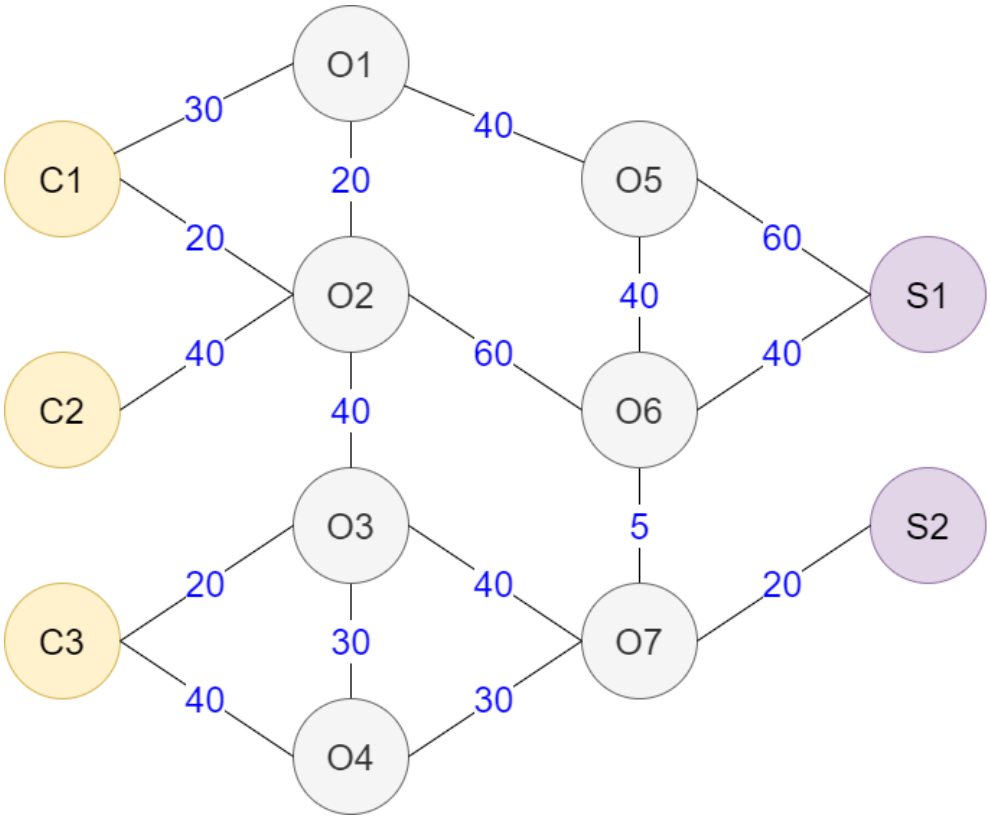
4.3.3 Exemple illustratif

Le modèle *Mc-DLN* fait usage du canal d'allocation (*AC*) pour étiqueter le canal d'état du réseau (*NSC*). Tandis que le *NSC* est généré à partir des entrées provenant des contrôleurs SDN via l'interface *ND*, le *AC* est généré par le composant d'optimisation (*OC*).

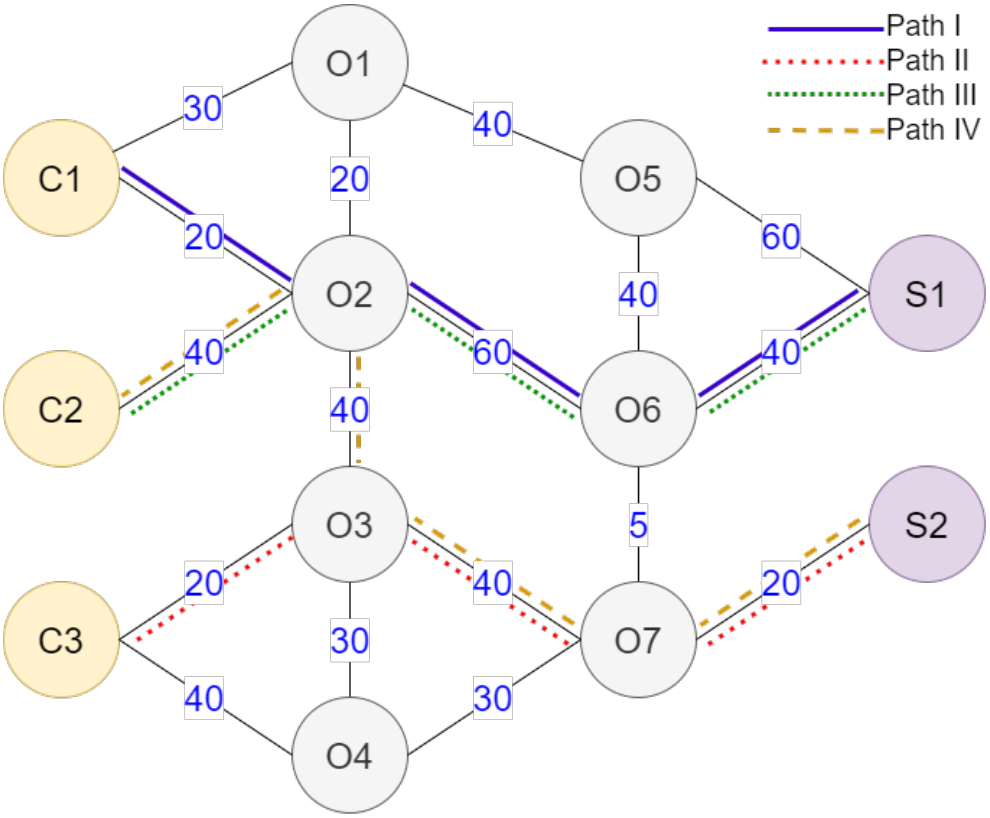
Pour illustrer la fonctionnalité du modèle *Mc-DLN*, prenons le réseau représenté dans la Figure 4.2 comme exemple.

La Figure 4.2(a) présente la topologie du réseau, c'est-à-dire le *CC*, tandis que la Figure 4.2(b) représente les mises à jour du réseau, soit le *AC*, générées par la solution *iPare*. Dans cet exemple, nous considérons un réseau composé de sept switchs virtuels (O1, O2, O3, O4, O5, O6, O7).

Comme mentionné dans le Tableau 4.2, le canal de capacité (*CC*) est généré à partir de la topologie du réseau présentée dans la Figure 4.2(a). En effet, le *CC* est une matrice de dimensions $\mathcal{N} \times \mathcal{N}$, où $\mathcal{N} = \mathcal{C} + \mathcal{O} + \mathcal{S}$.



(a) Network topology



(b) Allocation Channel

FIGURE 4.2 – Exemple illustratif

Dans cet exemple, le nombre de $\mathcal{C}$, $\mathcal{O}$ et $\mathcal{S}$ est respectivement de 3, 7 et 2. Le *CC* est rempli de zéros, à l'exception des cas où un lien entre les composants est présent, auquel cas la quantité de bande passante est spécifiée.

Le canal de requête (*RC*) présente une forme similaire à celle du *CC* et il spécifie les requêtes des clients pour chaque serveur dans le réseau. Les informations de requête sont stockées dans une matrice de requêtes qui définit pour chaque client les services souhaités ainsi que les exigences de qualité de service (QoS) requises. Le Tableau 4.3 décrit le *RC* utilisé dans cet exemple.

Selon le Tableau 4.3, le client C1 a besoin d'un chemin de communication avec le serveur S1, caractérisé par une bande passante de $10Mbps$. Parallèlement, C3 demande $20Mbps$ du même serveur. Enfin, le client C2 demande $5Mbps$ et $10Mbps$ des serveurs S1 et S2, respectivement.

TABLE 4.2 – The Capacity Channel (CC)

	C1	**C2**	**C3**	**O1**	**O2**	**O3**	**O4**	**O5**	**O6**	**O7**	**S1**	**S2**
C1	0	0	0	30	20	0	0	0	0	0	0	0
C2	0	0	0	0	40	0	0	0	0	0	0	0
C3	0	0	0	0	0	20	40	0	0	0	0	0
O1	0	0	0	0	20	0	0	40	0	0	0	0
O2	0	0	0	0	0	40	0	0	60	0	0	0
O3	0	0	0	0	0	0	30	0	0	40	0	0
O4	0	0	0	0	0	0	0	0	0	30	0	0
O5	0	0	0	0	0	0	0	0	40	0	60	0
O6	0	0	0	0	0	0	0	0	0	5	40	0
O7	0	0	0	0	0	0	0	0	0	0	0	20
S1	0	0	0	0	0	0	0	0	0	0	0	0
S2	0	0	0	0	0	0	0	0	0	0	0	0

TABLE 4.3 – The Request Channel (RC)

	C1	**C2**	**C3**	**O1**	**O2**	**O3**	**O4**	**O5**	**O6**	**O7**	**S1**	**S2**
C1	0	0	0	0	0	0	0	0	0	0	10	0
C2	0	0	0	0	0	0	0	0	0	0	5	10
C3	0	0	0	0	0	0	0	0	0	0	20	0
O1	0	0	0	0	0	0	0	0	0	0	0	0
O2	0	0	0	0	0	0	0	0	0	0	0	0
O3	0	0	0	0	0	0	0	0	0	0	0	0
O4	0	0	0	0	0	0	0	0	0	0	0	0
O5	0	0	0	0	0	0	0	0	0	0	0	0
O6	0	0	0	0	0	0	0	0	0	0	0	0
O7	0	0	0	0	0	0	0	0	0	0	0	0
S1	0	0	0	0	0	0	0	0	0	0	0	0
S2	0	0	0	0	0	0	0	0	0	0	0	0

Dans cet exemple, le résultat de sortie (output) du modèle *Mc-DLN*, c'est-à-dire le canal d'allocation (*AC*), est récapitulé dans le Tableau 4.4. Ce tableau représente le chemin optimal sélectionné respectant la *CC* (contrainte de capacité) et satisfaisant la *RC* (recherche de chemin).

Le *AC* contient des valeurs égales à 1 dans les éléments correspondant aux switchs virtuels activés, et des 0 dans les autres éléments, comme illustré dans le Tableau 4.4.

Une fois entraîné, le modèle *Mc-DLN* transforme chaque état du réseau reçu en un canal d'état du réseau *NSC*, puis prédit un vecteur d'activation qui spécifie les switchs virtuels devant être activés afin de satisfaire les demandes du *NSC* en fonction de leurs capacités respectives.

Comme mentionné précédemment, pendant l'étape d'entraînement, le *AC* correspond à la configuration optimale du réseau générée par l'*OC*. Cette approche assure une prise de décision optimale en générant des configurations réseau respectant à la fois les contraintes de capacité et les demandes des utilisateurs dans le contexte des réseaux SDN.

Ainsi, les chemins évoqués ci-dessus se présentent de la manière suivante :

— Chemin I : C1-O2-O6-S1
— Chemin II : C3-O3-O7-S2
— Chemin III : C2-O2-O6-S1
— Chemin IV : C2-O2-O3-O7-S2

Le Tableau 4.4 décrit le *AC* qui correspond aux chemins énumérés précédemment, où les switchs virtuels O2, O3, O6 et O7 devraient être activés. Ainsi, dans le *AC*, toutes les valeurs des neurones représentant ces switchs ont des valeurs supérieures au seuil prédéfini, tandis que les autres valeurs sont nulles.

Par la suite, le framework *DLO-SFC* renvoie au contrôleur SDN le *AC* tel que décrit dans le Tableau 4.4.

TABLE 4.4 – The Allocation Channel (AC)

O1	**O2**	**O3**	**O4**	**O5**	**O6**	**O7**
0	1	1	0	0	1	1

4.3.4 Model Checker (MC)

La configuration réseau optimale, générée par le modèle *Mc-DLN* (solution IA), est soumise à la vérification par le *Model Checker (MC)* afin de garantir sa faisabilité et le respect des indicateurs de performance souhaités. En outre, cette vérification s'assure que les chemins générés ne sont ni déconnectés ni ne forment de boucles.

Pour attester de la validité de la solution IA produite par le *Mc-DLN*, le *MC* exécute le processus de vérification décrit dans l'Algorithme 1. En effet, le *MC* prend en entrée les trois canaux formant l'information réseau : le *CC*, le *RC* et le *AC*.

Le processus de vérification renvoie une valeur booléenne correspondant à la faisabilité de la solution IA. Le *MC* ne valide que les solutions iPare réalisables qui connectent tous les clients du *RC* à leurs serveurs correspondants.

Algorithme 1 : aiSolutionVerification

Input : **cc**, **rc**, **ac** : Channel
clts : Client
srvs : Server
validity : Boolean
Output : validity

```
1  BEGIN
2  /* Extraire la liste des switchs virtuels activés à partir de l'AC */;
3  listActOVS = extractActivatedOVS(ac);
4  foreach c_i in clts do
5      foreach s_j in srvs do
6          if rc[c_i][s_j] ≠ 0 then
7              /* Construire un ensemble de nœuds accessibles depuis c_i */;
8              oldSet = {c_i};
9              newSet = {};
10             while newSet ≠ oldSet do
11                 newSet = newSet ∪ oldSet /* Pour la première itération */;
12                 oldSet = oldSet ∪ newSet;
13                 foreach x in oldSet do
14                     foreach y in listActOVS do
15                         if cc[x][y] ≠ 0 then
16                             newSet = newSet ∪ {y};
17                         end
18                     end
19                 end
20             end
21             /* Vérifier que le client est connecté à son serveur demandé */;
22             existS_j = FALSE;
23             foreach x in newSet do
24                 if cc[x][s_j] ≠ 0 then
25                     existS_j = TRUE;
26                     break;
27                 end
28             end
29             if existS_j = FALSE then
30                 validity = FALSE;
31                 return validity;
32             end
33         end
34     end
35 end
36 validity = TRUE;
37 return validity;
38 END
```

Le processus de vérification de l'Algorithme 1 construit, pour chaque client c_i demandant une connectivité vers le serveur s_j dans le *RC* (rc$[c_i][c_j] \neq 0$), un ensemble *newSet* contenant le client c_i et les nœuds (switchs virtuels actifs et serveurs) qui lui sont connectés. Le *MC* vérifie que chaque client c_i est relié à son serveur demandé s_j.

Dans le cas où un client c_i n'est pas relié au serveur s_j requis dans le *RC*, la solution iPare générée par le *Mc-DLN* est considérée comme invalide (validité = FAUX). La solution IA est considérée comme valide uniquement si tous les clients sont connectés à leurs serveurs demandés (validité = VRAI).

À noter que seules les configurations réseau générées valides sont mises en œuvre ; sinon, ce sont les configurations réseau générées par le *HC* ou le *OC* qui sont utilisées.

4.4 Routage multi-chemins dans la plateforme DLO-SFC

La plateforme DLO-SFC est déployée au sein d'un cluster Kubernetes et est interconnectée avec les divers contrôleurs SDN. Les éléments constitutifs de la plateforme sont organisés en espaces de noms (ou ***namespaces***) afin de renforcer la sécurité du système. Cette structuration permet la mise en place de politiques de sécurité spécifiques à chaque espace de noms, contribuant ainsi à garantir un environnement sécurisé et bien délimité.

Dans le cadre de notre solution DLO-SFC, nous avons mis en place au minimum six (06) espaces de noms, à savoir :

- **monitoring** : dédié à la supervision et à la collecte des informations du réseau SDN,
- **storage** : destiné à regrouper l'ensemble des bases de données utilisées par la plateforme,
- **orchestration** : cet espace de noms abrite les composants du système chargés de la coordination entre les différents éléments,
- **heuristic** : réservé à la mise en œuvre des solutions d'optimisation heuristique,
- **optimal** : dédié à la mise en œuvre des solutions d'optimisation exacte.
- **iPare** : spécifique au cas d'utilisation du routage multi-chemin optimal, cet espace de noms contient les éléments de la solution iPare.

Dans cette section, nous allons explorer comment la plateforme DLO-SFC peut être déployée pour le routage multi-chemins en utilisant la solution iPare. Il convient de noter que si d'autres fonctionnalités que iPare sont prises en charge, un espace de noms est créé pour chacune de ces fonctionnalités. En outre, les espaces de noms *heuristic* et *optimal* peuvent être étendus par l'ajout de composants relatifs à la fonctionnalité supplémentaire.

La Figure 4.3 offre une vue d'ensemble illustrative de la configuration de la plateforme DLO-SFC. En examinant le schéma, nous pouvons discerner comment chaque élément de la plateforme est agencé pour fournir la solution iPare.

Dans ce cadre, le Tableau 4.5 fournit une description détaillée des différentes opérations, comprenant leurs origines, leurs destinations, leurs descriptions ainsi que les types de protocoles utilisés pour garantir la sécurité des échanges de données.

Ce tableau offre une vue d'ensemble des flux d'informations critiques, notamment les transmissions de données entre le contrôleur SDN et les divers composants du réseau, tels que le Network Driver, Prometheus, l'Orchestrator, ainsi que d'autres services pertinents.

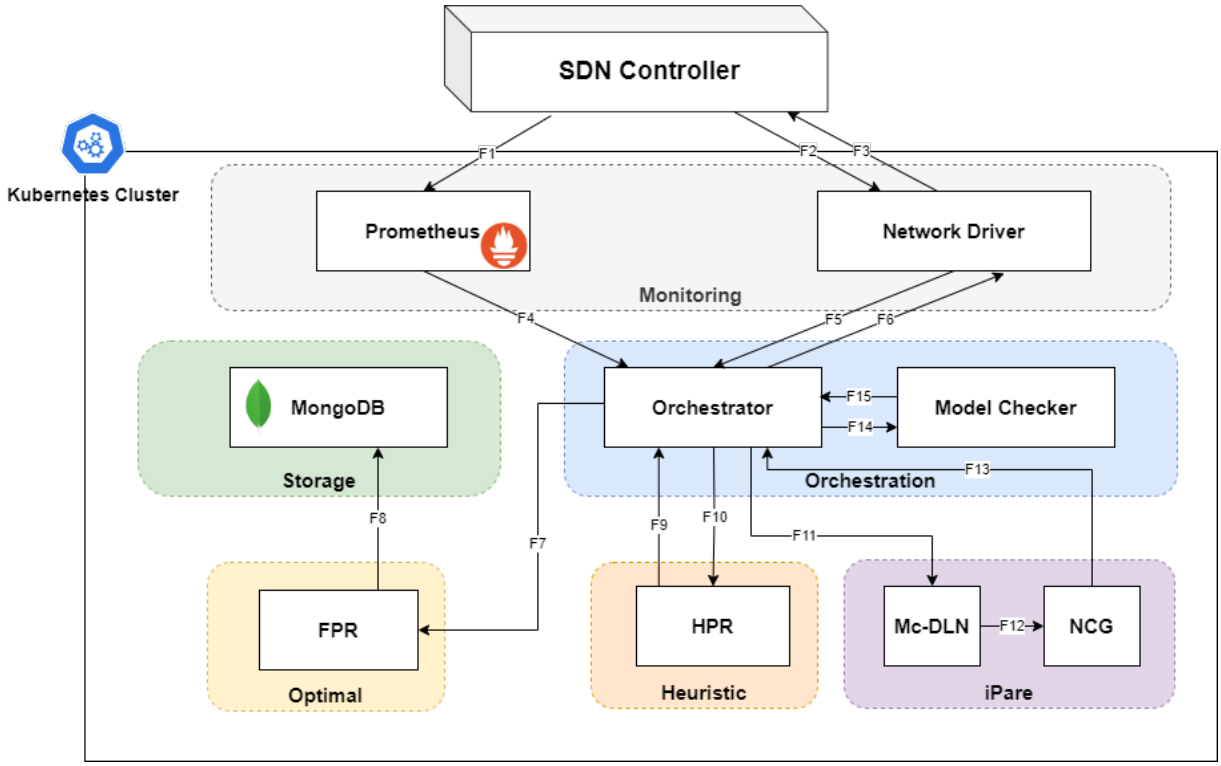


FIGURE 4.3 – Architecture micro-services de la plateforme DLO-SFC pour le routage multi-chemins

Il met en lumière l'usage répandu du protocole HTTP Over TLS (HTTPS) pour assurer la confidentialité et l'intégrité des données échangées au sein du système.

En effet, dans l'espace de noms *monitoring*, nous trouvons le composant **Prometheus**, qui fonctionne comme un agent de surveillance réseau (*Network monitoring agent*), ainsi que le composant **Network Driver**. Pour stocker les données de la plateforme telles que les topologies, les ensembles de données, etc., nous avons déployé la base de données NoSQL **MongoDB** dans l'espace de noms *storage*.

TABLE 4.5: Description des opérations

ID	Source	Destination	Description	Type
F1	SDN Controller	Prometheus	Les informations de la topologie du réseau récupérées depuis les *Exporters* présents au niveau des contrôleurs SDN sont transmises cryptées en utilisant le protocole HTTP Over TLS (HTTPS).	HTTPS
F2	SDN Controller	Network Driver	L'état du réseau est transmis par les contrôleurs SDN au Network Driver de manière sécurisée en utilisant HTTP Over TLS (HTTPS).	HTTPS

ID	Source	Destination	Description	Type
F3	Network Driver	SDN Controller	Une fois que l'orchestrateur récupère la configuration optimale (générée par iPare ou HPR), le Network Driver envoie les mises à jour à appliquer sur le réseau aux contrôleurs SDN. Le protocole HTTP Over TLS est utilisé pour garantir la sécurité des données échangées.	HTTPS
F4	Prometheus	Orchestrator	Prometheus notifie l'orchestrateur des changements de la topologie du réseau. Selon le modèle de menace considéré, il est possible d'opter soit pour le protocole HTTP, soit pour le protocole HTTP Over TLS. Par exemple, si la solution déployée cohabite avec d'autres applications sur le Cluster Kubernetes, il est recommandé (voire exigé) d'utiliser le protocole HTTP Over TLS pour garantir la sécurité des échanges. En revanche, si le cluster est dédié exclusivement à la solution, l'utilisation du protocole HTTP peut être envisagée.	HTTP / HTTPS
F5	Network Driver	Orchestrator	L'orchestrateur récupère l'état du réseau depuis le Network Driver en utilisant soit le protocole HTTP, soit le protocole HTTP Over TLS.	HTTP / HTTPS
F6	Orchestrator	Network Driver	Une fois que la configuration optimale est choisie, l'orchestrateur l'envoie vers le Network Driver en utilisant soit HTTP, soit HTTP over TLS.	HTTP / HTTPS
F7	Orchestrator	FPR	L'orchestrateur transmet l'état du réseau et sa topologie au service FPR pour que ce dernier lance le calcul de la solution exacte FPR, en utilisant soit le protocole HTTP, soit le protocole HTTP Over TLS.	HTTP / HTTPS
F8	FPR	MongoDB	Le service FPR enregistre la configuration optimale calculée dans la base de données MongoDB en utilisant le protocole MongoDB Wire Protocol.	MongoDB Wire Protocol

ID	Source	Destination	Description	Type
F9	HPR	Orchestrator	Le service HPR retourne la solution heuristique à l'orchestrateur en utilisant le protocole HTTP ou le protocole HTTP Over TLS	HTTP / HTTPS
F10	Orchestrator	HPR	L'état du réseau et sa topologie sont transmis au service HPR en utilisant soit le protocole HTTP, soit le protocole HTTP over TLS.	HTTP / HTTPS
F11	Orchestrator	Mc-DLN	L'orchestrateur transmet l'état du réseau et sa topologie au service d'apprentissage profond Mc-DLN du nom d'espace iPare en utilisant le protocole HTTP ou le protocole HTTP Over TLS.	HTTP / HTTPS
F12	Mc-DLN	NCG	Le service d'apprentissage profond Mc-DLN transmet le vecteur d'activation des OVS au service de génération de configuration réseau NCG en utilisant le protocole HTTP ou le protocole HTTP Over TLS.	HTTP / HTTPS
F13	NCG	Orchestrator	Le service de génération de configuration réseau NCG envoie la configuration réseau générée à l'orchestrateur en utilisant le protocole HTTP ou le protocole HTTP Over TLS.	HTTP / HTTPS
F14	Orchestrator	Model Checker	La configuration générée par la solution iPare et retournée par le générateur de configuration réseau NCG est envoyée pour validation au Model Checker en utilisant le protocole HTTP ou le protocole HTTP Over TLS.	HTTP / HTTPS
F15	Model Checker	Orchestrator	Le Model Checker retourne le résultat de validation à l'Orchestrateur en utilisant le protocole HTTP ou le protocole HTTP Over TLS.	HTTP / HTTPS

L'**Orchestrateur** se distingue comme le composant central de la plateforme DLO-SFC. Son rôle prépondérant consiste à coordonner les actions des divers composants afin d'appliquer un routage multi-chemin optimal, tout en garantissant une qualité de service optimale. Ce composant stratégique est déployé dans l'espace de noms *orchestration*, tout comme le composant **Model Checker**. Ce dernier assure la cohérence de la solution fournie par le modèle d'intelligence artificielle de la solution iPare.

Dans notre architecture, l'espace de noms *optimal* abrite un micro-service qui implémente la solution FPR, tandis que dans l'espace de noms *heuristic*, nous trouvons un micro-

service fournissant la solution HPR. Il est à noter que d'autres solutions, en plus de ces deux algorithmes, peuvent être implémentées en tant que solutions exactes et/ou heuristiques. Le système peut être configuré pour utiliser telle ou telle solution en fonction des besoins spécifiques.

Enfin, , dans l'espace de noms *iPare*, sont hébergés les composants essentiels de la solution *iPare*. Ces composants comprennent le réseau de neurones **Mc-DLN**, responsable de l'analyse et de la prise de décisions basées sur des modèles pour optimiser le routage multi-chemin, ainsi que le générateur de configuration réseau **NCG**, chargé de mettre en œuvre ces décisions dans la configuration effective du réseau.

Le micro-service Prometheus est intégré à la plateforme DLO-SFC afin de capturer la topologie du réseau. Cette topologie est ensuite enregistrée dans la base de données MongoDB. À chaque modification de cette topologie, le micro-service Prometheus reçoit une notification afin de mettre à jour la base de données MongoDB en conséquence.

Lorsque le Network Driver reçoit l'état actuel du réseau, il transmet ces données à l'Orchestrateur, qui les partage ensuite avec iPare, FPR et HPR. Le micro-service FPR calcule la solution optimale pour mettre à jour le jeu de données (dataset) dans la base de données MongoDB. Ce jeu de données est ultérieurement utilisé pour mettre à jour le modèle du réseau Mc-DLN, employé par iPare pour générer des solutions basées sur l'IA.

iPare et HPR calculent chacun une solution de routage multi-chemin correspondant à l'état du réseau reçu. La solution basée sur l'IA générée par iPare est évaluée par le Model Checker. Si elle est validée par ce dernier, elle est retenue. Dans le cas contraire, l'orchestrateur retient la solution heuristique générée par le composant HPR. Par la suite, le Network Driver communique les mises à jour qui seront transmises et appliquées par les contrôleurs SDN.

4.5 Conclusion

En conclusion, ce chapitre a exploré les divers aspects du routage multi-chemins intelligent, mettant en lumière des solutions et des architectures pour optimiser l'utilisation des ressources réseau dans les environnements SDN.

Dans la première section, nous avons examiné les stratégies d'optimisation du routage multi-chemins dans les SDN, en se focalisant sur les approches *FPR (Full Paths Recomputation)* et *HPR (Heuristic Paths Recomputation)* proposées par Bagaa et al. Ces solutions offrent des perspectives intéressantes pour configurer efficacement les chemins de transmission, tout en tenant compte des contraintes de temps et des exigences de qualité de service (QoS).

La deuxième section a introduit *iPare (Intelligent Path Re-computation)*, un modèle d'intelligence artificielle innovant conçu pour fournir des configurations de chemins multiples optimales dans les réseaux basés sur SDN. *iPare*, intégré au framework DLO-SFC, tire parti de l'apprentissage profond et de la classification multi-étiquettes pour générer des configurations réseau adaptées à chaque situation.

Enfin, la troisième section a présenté la mise en œuvre pratique du routage multi-chemins dans la plateforme DLO-SFC, illustrant la manière dont les solutions abordées peuvent être intégrées dans un contexte opérationnel. Cela met en évidence son déploiement au sein d'un environnement Kubernetes ainsi que son intégration étroite avec les contrôleurs SDN.

Nous avons également décrit l'architecture modulaire de la plateforme DLO-SFC, qui permet d'ajouter facilement de nouvelles fonctionnalités et de garantir la sécurité et la fiabilité du système.

Grâce à une architecture bien définie et une orchestration précise des composants, la plateforme DLO-SFC proposée offre une solution complète pour le routage multi-chemins, allant de la collecte des données réseau à la prise de décision intelligente, tout en garantissant la cohérence et la fiabilité du réseau.

Dans l'ensemble, ce chapitre a démontré l'importance croissante du routage multi-chemins intelligent dans les réseaux modernes, ainsi que les défis et les opportunités associés à son déploiement.

En combinant des techniques d'intelligence artificielle, les solutions présentées permettent d'améliorer les performances et l'adaptabilité des réseaux SDN pour répondre aux besoins évolutifs des applications et des utilisateurs.

Chapitre 5

Étude expérimentale

Il faut toujours viser la lune, car même en cas d'échec, on atterrit dans les étoiles.

Oscar Wilde (1854-1900)

5.1 Introduction

Dans ce chapitre, nous présenterons la configuration de la simulation mise en place pour notre étude. Nous procéderons également à une analyse approfondie des résultats obtenus à partir de notre réseau d'apprentissage profond de classification multi-étiquettes *Mc-DLN*, tel que proposé au sein de la solution iPare.

Cette dernière constitue une approche basée sur l'apprentissage profond visant à fournir des configurations optimales de réseau pour orienter le trafic dans un réseau compatible avec le paradigme SDN (Software-Defined Networking).

L'objectif principal de cette démarche est de canaliser efficacement le flux de données au sein d'un environnement réseau tout en garantissant le respect des impératifs des accords de niveau de service (SLA) convenus.

En outre, cette méthodologie s'inscrit dans une perspective économique et opérationnelle en cherchant à réduire de manière significative les coûts d'exploitation (OpEx) associés à la gestion courante de l'infrastructure.

Cette réduction des coûts est obtenue en minimisant le nombre de switchs actifs.

À cet égard, *iPare* a été élaboré pour pallier les limitations relatives à FPR. Le temps d'exécution constitue un problème majeur entravant la capacité de FPR à s'adapter aux changements rapides survenant au sein du réseau.

En effet, bien que FPR offre des configurations réseau optimales en recourant à des méthodes d'optimisation, plus précisément la programmation linéaire en nombres entiers, son principal inconvénient réside dans le temps d'exécution considérable requis pour générer ces configurations.

En revanche, pour la mise en place d'*iPare*, FPR a été employé pour générer l'ensemble de données (*dataset*) sous-jacent à son processus d'entraînement. Puis, lors de la phase d'inférence, nous avons évalué l'exactitude de la performance d'*iPare* en comparant ses résultats à ceux générés par FPR, correspondant à la configuration optimale.

Il convient de souligner que la finalité d'*iPare* ne réside pas dans la volonté de dépasser FPR pour générer de meilleures configurations, mais plutôt chercher à fournir des configurations similaires, qui demeurent optimales, dans un intervalle de temps raisonnable.

Cette approche est particulièrement pertinente dans les situations où l'efficacité temporelle s'avère être une contrainte cruciale, tout en maintenant une qualité de service équivalente à celle obtenue par des méthodes plus chronophages.

Dans la continuité de ce chapitre, nous entreprendrons tout d'abord de détailler la configuration de la simulation mise en place. Cette configuration exhaustive revêt une importance cruciale, car elle assure la mise en œuvre méthodique et précise des paramètres intrinsèques à notre étude. En effet, il est primordial d'instaurer un environnement de simulation fidèle à la réalité, ce qui requiert une attention particulière lors de l'établissement des configurations.

Par la suite, notre attention se portera sur l'évaluation méthodique et rigoureuse de l'apprentissage orchestré par le modèle *iPare*. Cette évaluation comprendra une analyse approfondie des résultats obtenus par le modèle, sondant particulièrement ses performances et ses niveaux de précision.

En procédant à cette évaluation méthodique comparative, nous serons en mesure d'identifier les avantages et les éventuelles limites du mode d'inférence du modèle *iPare*, en comparaison avec une approche établie de référence telle que FPR.

Ce processus d'analyse comparative permettra d'approfondir notre compréhension, permettant ainsi de formuler des conclusions avisées quant à la pertinence et à l'efficacité des choix méthodologiques mis en œuvre dans le cadre de notre étude.

5.2 Configuration expérimentale

Pour la validation des performances de notre travail, nous avons eu recours à l'implémentation du modèle au sein du framework *Tensorflow* [93], une plateforme Open Source dédiée au Machine Learning.

Le composant *Mc-DLN*, de notre solution *iPare*, fait appel à un Réseau de Neurones Convolutif (*Convolutional Neural Network (CNN)*, en anglais), au sein duquel la couche d'entrée est constituée d'éléments clés de notre modèle, à savoir les clients, les switchs virtuels et les serveurs, ainsi que la couche de sortie qui se compose de différentes classes correspondant au nombre de switchs virtuels dans le réseau étudié.

Dans le contexte de l'optimisation des routages multi-chemins dans les réseaux SDN (Software Defined Networking), l'utilisation des Réseau de Neurones Convolutif (CNN) se présente comme une solution prometteuse pour la prédiction et l'optimisation. Cette approche pourrait être appliquée pour sélectionner les chemins les plus efficaces en fonction des conditions du réseau en temps réel, offrant ainsi une fiabilité et une adaptabilité optimales pour répondre aux exigences croissantes des environnements réseau contemporains. Elle permet une gestion dynamique et intelligente du trafic, contribuant à améliorer les performances réseau.

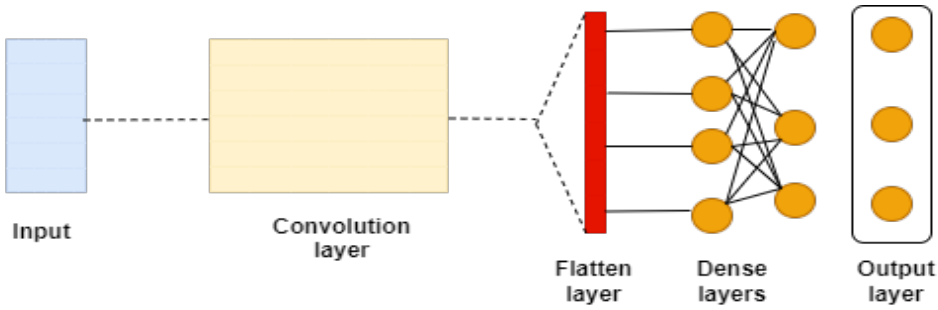


FIGURE 5.1 – Architecture du CNN (Mc-DLN)

La Figure 5.1 illustre l'architecture du modèle de réseau de neurones convolutifs utilisé dans le cadre du Mc-DLN. Ce modèle est conçu pour optimiser la reconnaissance et la classification multi-label en analysant les données des requêtes des utilisateurs ainsi que la topologie des réseaux SDN. Chaque couche du modèle joue un rôle crucial dans l'extraction et l'apprentissage des caractéristiques des données d'entrée.

Les couches de convolution, suivies de fonctions d'activation *ReLU*, permettent de capturer les motifs complexes présents dans les données des requêtes et les configurations topologiques. La couche de flattening transforme les données en une forme compatible avec les couches denses, où des connexions pleinement connectées sont établies.

La couche de sortie dense utilise la fonction d'activation *sigmoïde*, particulièrement adaptée pour les tâches de classification multi-label dans le contexte des réseaux SDN. La fonction *sigmoïde* convertit les sorties du modèle en probabilités, chaque sortie représentant la probabilité qu'un certain OvS soit activé (1) ou non (0). Cela permet au modèle de faire des prédictions indépendantes pour chaque OvS.

Les différents hyperparamètres du modèle, tels que le nombre de filtres, la taille des kernels, et autres, sont soigneusement ajustés pour maximiser les performances du modèle. Ces hyperparamètres sont présentés dans le Tableau 5.1, fournissant une vue d'ensemble complète des configurations utilisées. Ce tableau permet de comprendre les choix techniques et les réglages qui ont conduit à l'optimisation du modèle CNN pour les tâches spécifiques du Mc-DLN.

TABLE 5.1 – Hyperparamètres du modèle CNN (Mc-DLN)

Setting	Value
Optimizer	Adam
Learning_rate	0.001
Loss	binary_crossentropy
Metrics	binary_accuracy
Activation_threshold	0.5
Filters	32
Kernel size	3 x 3

Cependant, l'évaluation de nos expérimentations repose sur des mesures temporelles d'exécution, essentielles pour établir une compréhension approfondie de l'efficacité de notre modèle. Ces mesures nous permettent d'évaluer notre approche et de comparer ses résultats avec d'autres méthodes.

Les mesures ont été consignées sur une plateforme matérielle, équipée d'un processeur CPU Intel® Genuine Intel® cadencé à $2.00\ GHz$ pour une puissance de calcul considérable, et une mémoire vive de $32\ GB$ pour garantir la fluidité des opérations tout au long du processus d'évaluation de nos expérimentations.

En outre, la distribution Ubuntu Server [97] version $16.04\ LTS$ a été adoptée comme environnement d'exécution pour assurer une cohérence et une stabilité optimales. Les spécifications matérielles et logicielles utilisées sont résumées dans le Tableau 5.2.

TABLE 5.2 – Spécifications matérielles et logicielles utilisées

Composant	Spécification
Processeur	Intel® Genuine Intel® CPU cadencé à 2.00 GHz
Mémoire vive	32 GB
Système d'exploitation	Ubuntu Server 16.04 LTS

TABLE 5.3 – Données expérimentales

Setting	Value
Number of clients	5
Number of OvS	20
Number of servers	3
Batch size	32
Training data size	4800
Test data size	1200
Total data size	6000

Pour évaluer la précision du *Mc-DLN*, nous avons pris en considération l'ensemble de données expérimentales décrit dans le Tableau 5.3, lequel se compose de 6000 états de réseau ainsi que de leurs configurations optimales correspondantes. Ces configurations ont été obtenues à l'aide de l'algorithme d'optimisation *Full Path Re-computation (FPR)*, tel que présenté dans l'étude de Bagaa et al. [89] (voir Section 4.2).

Afin de constituer cet ensemble de données, nous avons implémenté l'algorithme FPR en utilisant le langage de programmation Python, qui a permis d'effectuer avec précision les calculs des configurations optimales pour chaque état du réseau au sein de l'ensemble de données.

Cette implémentation a été simplifiée grâce à l'exploitation de la bibliothèque NetworkX [96], qui offre des fonctionnalités puissantes pour la modélisation et l'analyse des graphes, et du logiciel Gurobi Optimizer [94], reconnu pour son efficacité dans la résolution de problèmes d'optimisation complexes.

Pour la phase de constitution de l'ensemble de données (*Dataset*), nous avons attribué 80% (4800 configurations) de la taille des données à l'entraînement du modèle Mc-DLN (*Training dataset*), tandis que les 20% restants (1200 configurations) ont été réservés à l'évaluation de la précision et les performances du modèle après l'entraînement (*Testing dataset*).

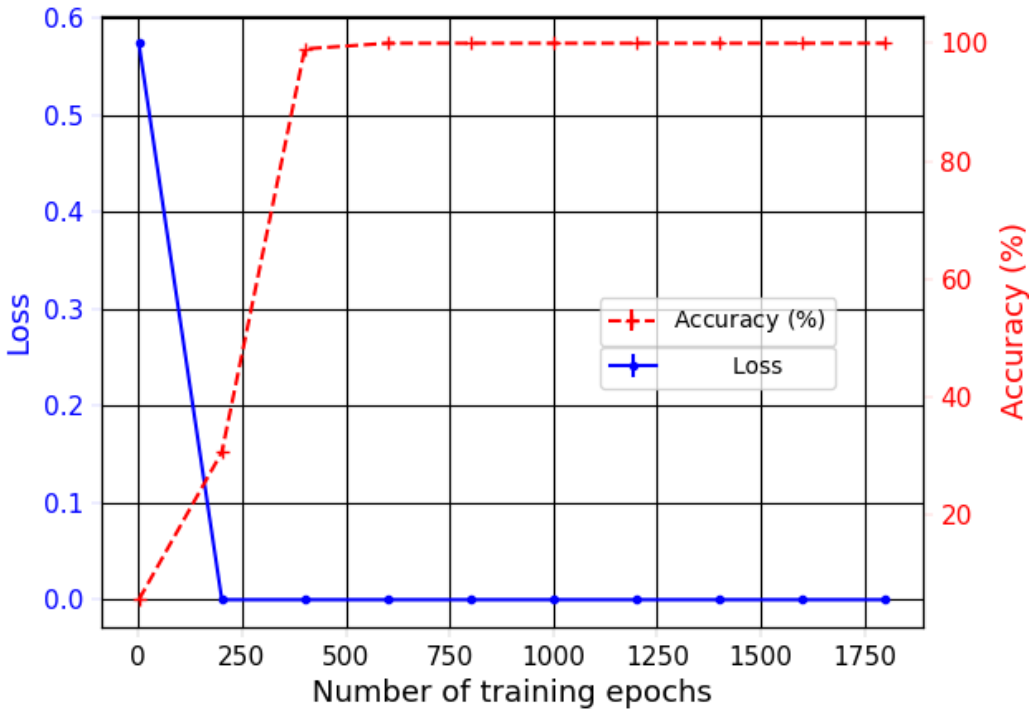
Pendant le processus d'entraînement, nous avons choisi une taille de lot (*batch size*) égale à 32 pour chaque itération. Cela signifie que le modèle a été mis à jour après avoir traité 32 exemples d'entraînement à la fois. La taille de lot de 32 a été choisie pour établir un judicieux équilibre entre l'efficacité du traitement et les ressources de calcul nécessaires.

Nous avons répété ce processus sur l'ensemble du jeu de données d'entraînement pendant 700 époques d'entraînement, permettant ainsi au modèle d'apprendre progressivement les caractéristiques et les relations dans les données.

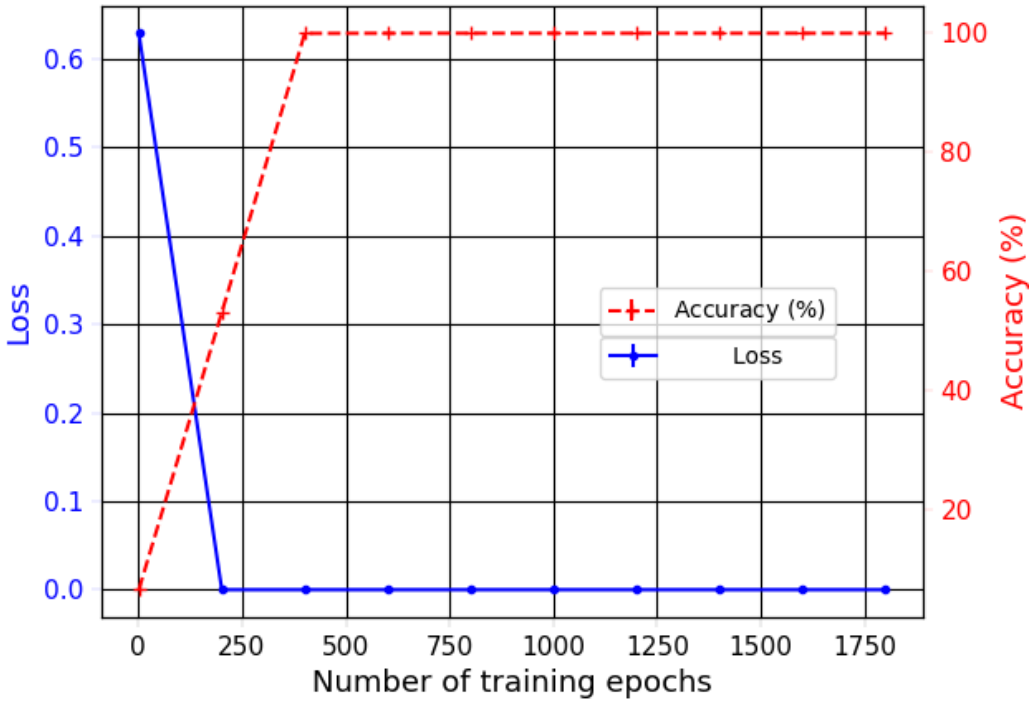
Cette approche nous a permis d'obtenir des poids de modèle optimaux pour prédire et optimiser les chemins de manière efficace dans un environnement SDN.

L'ensemble de données expérimentales mentionné précédemment a été rigoureusement élaboré pour garantir une représentation adéquate des divers états du réseau, ainsi que de leurs configurations optimales respectives. Cette élaboration rigoureuse garantit que les données utilisées pour l'entraînement et les tests reflètent fidèlement la complexité et la diversité des scénarios réels rencontrés dans les réseaux SDN.

Enfin, l'évaluation de la précision du modèle *Mc-DLN* repose sur des bases expérimentales robustes établies autour d'un ensemble de données représentatif et d'une méthodologie d'entraînement réfléchie. Les choix de mise en œuvre, telles que l'adoption de l'algorithme FPR, l'utilisation des ressources Python, NetworkX et Gurobi Optimizer, ainsi que la configuration des paramètres d'entraînement, ont été prises avec soin pour garantir des résultats fiables et significatifs dans la validation du modèle proposé.



(a) Nombre de clients 5



(b) Nombre de clients 10

FIGURE 5.2 – La perte (*Loss*) et la précision (*Accuracy*) d'iPare pendant le mode d'entraînement.

5.3 Évaluation du mode d'entraînement iPare

Au sein de cette section, nous procédons à une démarche d'évaluation de l'entraînement de la solution iPare, se focalisant plus particulièrement sur le modèle *Mc-DLN*. Nous nous concentrons spécifiquement sur la manipulation des variations du nombre de clients pour évaluer les performances du modèle.

Comme mentionné précédemment, nous avons généré l'ensemble de données (*Dataset*) à l'aide de la solution FPR. Ce *Dataset* comprend des informations sur l'état du réseau, telles que les requêtes des utilisateurs et la topologie du réseau, ainsi que les graphes de transfert multi-chemins valides (*multi-path forwarding graphs*, en anglais), issus des mécanismes de génération de FPR.

Dans cette perspective, alors que l'état du réseau constitue la variable d'entrée au sein de cet ensemble de données, le graphe de transfert se pose en tant que variable cible. Chaque graphe de transfert se compose d'un ensemble de switchs qui doivent être activés.

À cet effet, chaque état du réseau, accompagné de son graphe de transfert respectif, s'est vu assujetti à une démarche de traitement permettant l'extraction précise de canaux essentiels, à savoir le canal de capacité (CC), le canal de requête (RC), ainsi que leur canal d'allocation correspondant (AC).

En raison de l'utilisation de la programmation linéaire en nombres entiers par FPR, l'exécution de cet algorithme requiert un temps considérable pour générer l'ensemble de données. En plus, il convient de noter que FPR peut rencontrer des difficultés à déterminer la solution optimale dans le cas de scénarios complexes, en raison des limites inhérentes aux ressources disponibles.

Dans cette situation, la création de notre vaste ensemble de données s'est avérée être une tâche particulièrement exigeante. Pour relever ces défis, nous avons choisi d'explorer des configurations de réseau de taille modeste, ceci dans le but de mettre en lumière la viabilité de la solution iPare que nous proposons.

Conformément aux données expérimentales présentées dans le Tableau 5.3, notre approche expérimentale s'est principalement concentrée sur une topologie réseau englobant 20 switchs virtuels (OvS), 5 clients et 3 serveurs. Cette configuration a été sélectionnée pour représenter une variété de scénarios tout en permettant une exécution fluide des calculs.

La flexibilité de notre approche est également illustrée par la variabilité que nous avons intégrée dans l'évaluation. Nous avons opéré des ajustements dans le nombre de clients, de serveurs et de switchs virtuels, afin d'évaluer l'impact de ces paramètres sur l'efficacité et la performance de la solution iPare.

Cette approche systématique nous a permis d'explorer en détail la réaction de notre modèle aux variations des éléments clés du réseau, contribuant ainsi à notre compréhension des paramètres optimaux pour l'utilisation d'iPare.

Nous avons effectué plusieurs itérations afin de déterminer la valeur optimale du seuil ϵ (*threshold*, en anglais), dans le but de parvenir à un compromis équitable entre la précision (*accuracy*, en anglais) et la charge (*overhead*, en anglais). Ce processus a nécessité une série d'exécutions itératives, visant à affiner les paramètres pour assurer une performance optimale.

La Figure 5.2 offre une représentation visuelle du processus d'entraînement d'iPare, démontrant son fonctionnement dans deux scénarios distincts. Pour ces deux scénarios, nous avons opté pour l'utilisation de 20 switchs OvS et 3 serveurs.

Pour apporter une variabilité contrôlée à notre évaluation, nous avons choisi de faire varier le nombre de clients entre les scénarios. Plus précisément, nous avons utilisé 5 clients dans le premier scénario, puis avons augmenté ce nombre à 10 clients dans le second. Cette approche nous permet d'explorer les répercussions des variations de la charge de trafic sur les performances du modèle.

L'analyse de la figure révèle que le modèle iPare atteint un état de convergence satisfaisant après seulement 300 époques de formation. Cette convergence est illustrée par l'atteinte de niveaux de précision de 100%, signifiant que les prédictions du modèle correspondent en totalité aux configurations optimales fournies par l'algorithme FPR. En outre, le modèle affiche une fonction de perte pratiquement insignifiante, soulignant sa capacité à minimiser les écarts entre les prédictions et les solutions optimales.

Enfin, il convient de souligner que l'atteinte d'une précision de 100% et d'une perte quasi nulle est un résultat remarquable, témoignant de l'efficacité du modèle iPare dans la résolution de ce problème complexe.

Ces performances sont d'autant plus significatives lorsqu'elles sont considérées dans le contexte des configurations optimales fournies par FPR, qui servent de référence pour l'évaluation. En mesurant simultanément la précision et la perte et en les comparant à ces configurations de référence, nous établissons un critère de validation qui confirme la capacité du modèle à produire des résultats fiables.

5.4 Évaluation du mode d'inférence d'iPare

Dans cette section, nous abordons la phase de mode d'inférence de notre modèle iPare, au cours de laquelle il est mis à l'épreuve en effectuant des prédictions ou des classifications sur de nouvelles données, nous permettant ainsi d'évaluer ses performances dans des situations réelles.

Cette étape nous conduit à entreprendre une comparaison systématique entre les méthodes iPare, FPR et HPR, en nous basant sur un ensemble de paramètres d'évaluation préalablement définis, à savoir :

— temps d'exécution ;
— précision ;
— surcharge ;
— coût.

Le dernier indicateur (coût) renvoie au nombre de switchs virtuels qu'il convient d'activer. Au sein de cette phase d'inférence, nous élaborons un ensemble complet d'états de réseau, c'est-à-dire les requêtes des utilisateurs et la configuration topologique du réseau, ainsi que leurs graphes de transfert multi-chemins valides, générés selon la méthode FPR [89].

Conformément à l'étape d'entraînement, nous extrayons les canaux de capacité (CC), de requête (RC) ainsi que leur canal d'allocation associé (AC). Ces deux éléments essentiels sont ensuite utilisés conjointement au niveau de la couche d'entrée du modèle *Mc-DLN*.

Ensuite, nous procédons à l'activation des switchs virtuels en fonction des informations contenues dans le canal d'allocation (AC), ainsi que de la valeur préalablement définie du seuil ϵ lors de l'étape d'entraînement.

Par la suite, nous entreprenons une comparaison entre les switchs virtuels activés, également appelés Open vSwitches (OvSs), ainsi que les graphes de transfert générées, avec ceux produits au moyen de l'algorithme FPR.

Cette démarche analytique vise à quantifier avec précision la concordance entre les résultats engendrés par le modèle *Mc-DLN* et ceux procurés par la méthode de référence FPR, en termes de précision et de surcharge.

Il est important de souligner que la notion de surcharge se réfère au nombre supplémentaire de switchs virtuels (OvSs) activés par *Mc-DLN*, en comparaison avec FPR. Cependant, la mesure de précision repose sur l'évaluation en pourcentage des clients ayant atteint un niveau de service (SLA) répondant à leurs attentes.

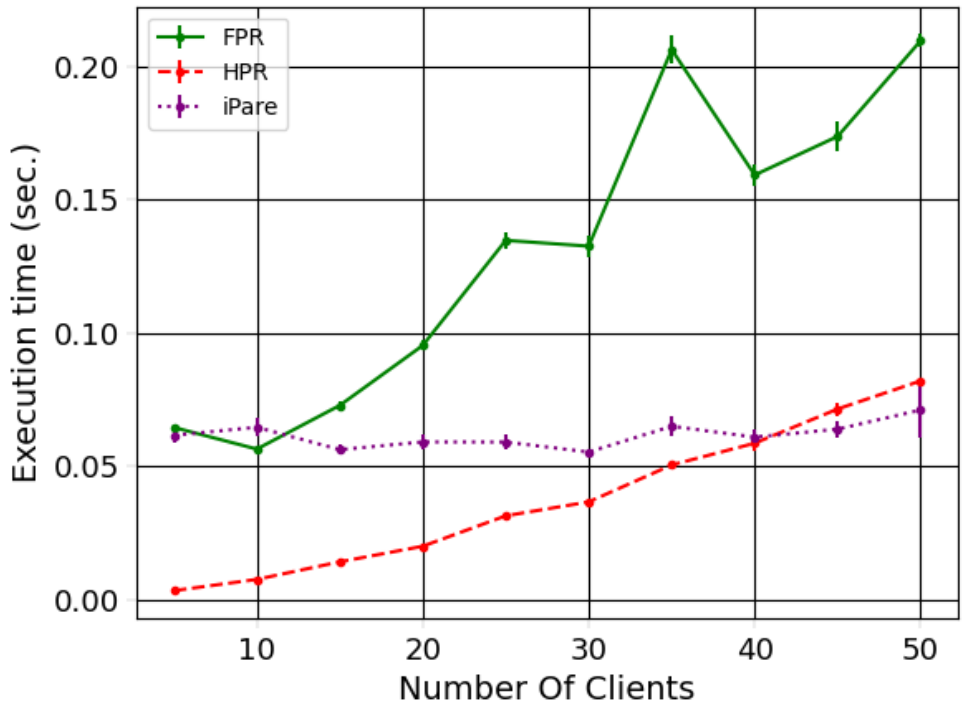


FIGURE 5.3 – La complexité d'exécution de la solution iPare comparée à FPR et HPR, en fonction du nombre de clients.

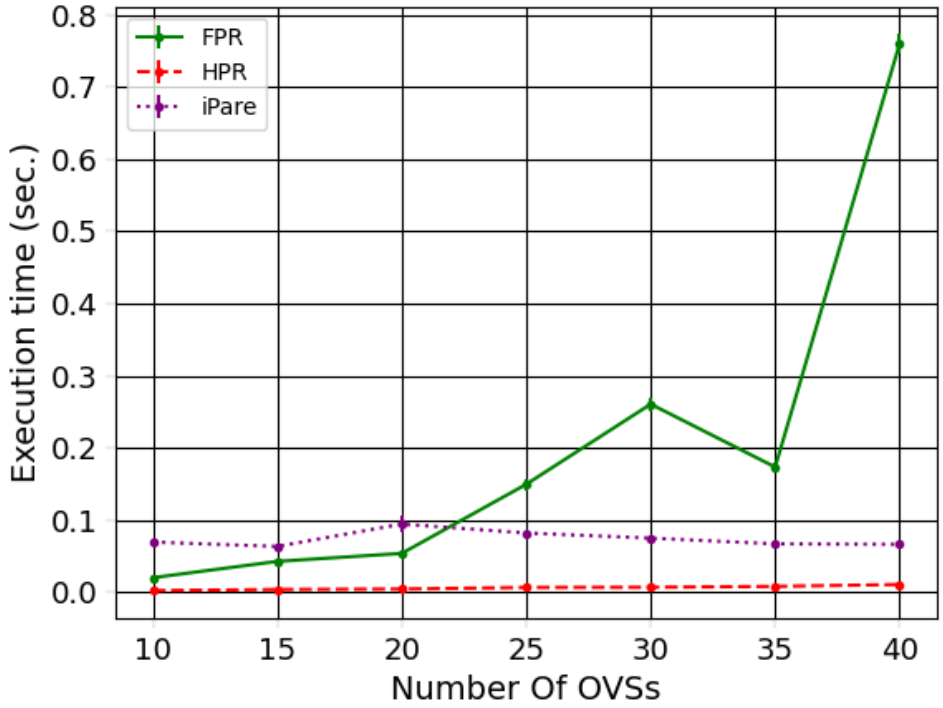


FIGURE 5.4 – La complexité d'exécution de la solution iPare comparée à FPR et HPR, en fonction du nombre de OVSs.

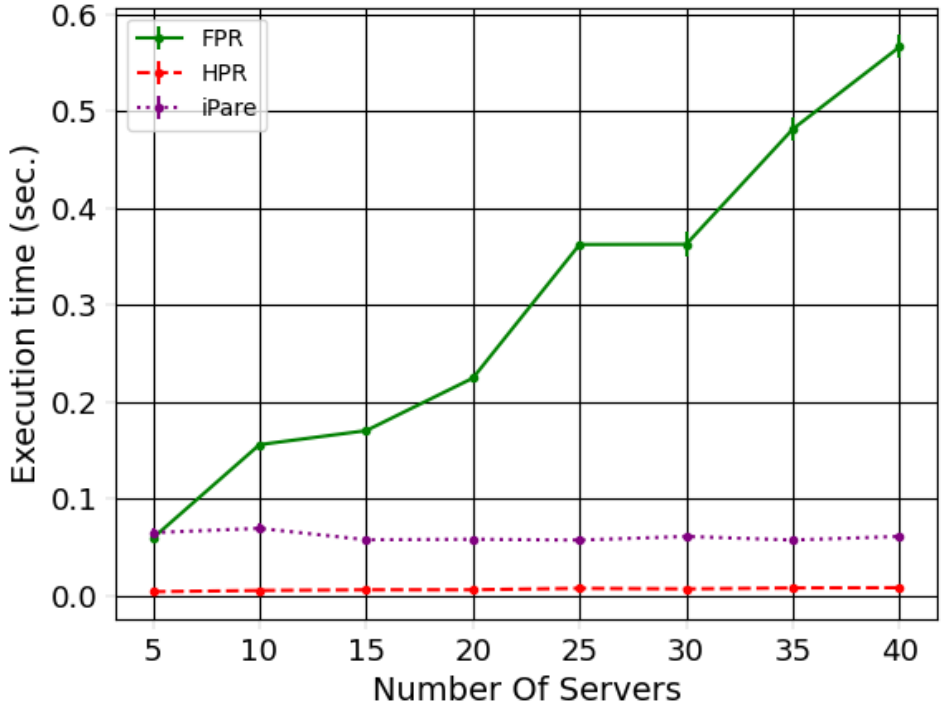


FIGURE 5.5 – La complexité d'exécution de la solution iPare comparée à FPR et HPR, en fonction du nombre de serveurs.

Dans le cadre de notre étude, nous avons entrepris trois séries d'expérimentations.

Dans un premier temps, nous avons procédé à une comparaison des performances d'*iPare* par rapport à celles de FPR, en mettant l'accent sur la complexité du temps d'exécution. L'objectif sous-jacent à cette expérience était de démontrer la capacité inhérente d'*iPare* à capturer les variations rapides qui se produisent dans le réseau.

Afin d'atteindre cet objectif, nous avons entrepris trois séries distinctes d'évaluations méthodiques :

$i)$ nous avons fait varier le nombre de clients tout en maintenant le nombre de switchs virtuels (OVSs) et de serveurs à 20 et 3 respectivement,

$ii)$ nous avons fait varier le nombre de switchs virtuels (OVSs), tout en maintenant le nombre de clients et de serveurs à 5 et 3 respectivement,

$iii)$ enfin, nous avons fait varier le nombre de serveurs tout en maintenant les nombres de switchs virtuels (OVSs) et de clients à 20 et 5 respectivement.

Chacun de ces ensembles d'évaluations a été minutieusement conçu pour examiner les performances d'*iPare* dans des contextes différents, tout en maintenant certains paramètres constants, ce qui nous a permis d'éclairer la façon dont le modèle réagit aux fluctuations au sein du réseau.

Cette approche méthodique renforce la validité et la pertinence des résultats obtenus, et offre des aperçus précieux quant à la capacité d'*iPare* à saisir et à gérer efficacement les dynamiques complexes et évolutives du réseau.

En second lieu, une évaluation approfondie a été menée, impliquant l'analyse de la précision et de la surcharge, tout en manipulant diverses valeurs du seuil d'activation ϵ.

Il convient de souligner que le nombre de switchs virtuels (OVSs) et de serveurs a été fixé à 20 et 3 respectivement, dans le but de maintenir un contexte d'expérimentation cohérent.

Parallèlement, deux scénarios distincts ont été pris en considération, où le nombre de clients a été maintenu à 5 et 10 respectivement, engendrant ainsi des conditions variées et représentatives.

Au sein de cette expérimentation méthodique, la précision a été rigoureusement évaluée, s'exprimant sous la forme d'un pourcentage de configurations valides calculées à l'aide du modèle *Mc-DLN*, et ensuite comparées aux configurations optimales générées par l'approche FPR.

Par cette démarche, il s'agit de mesurer le degré d'adéquation et de conformité du modèle proposé envers les solutions optimales établies par FPR.

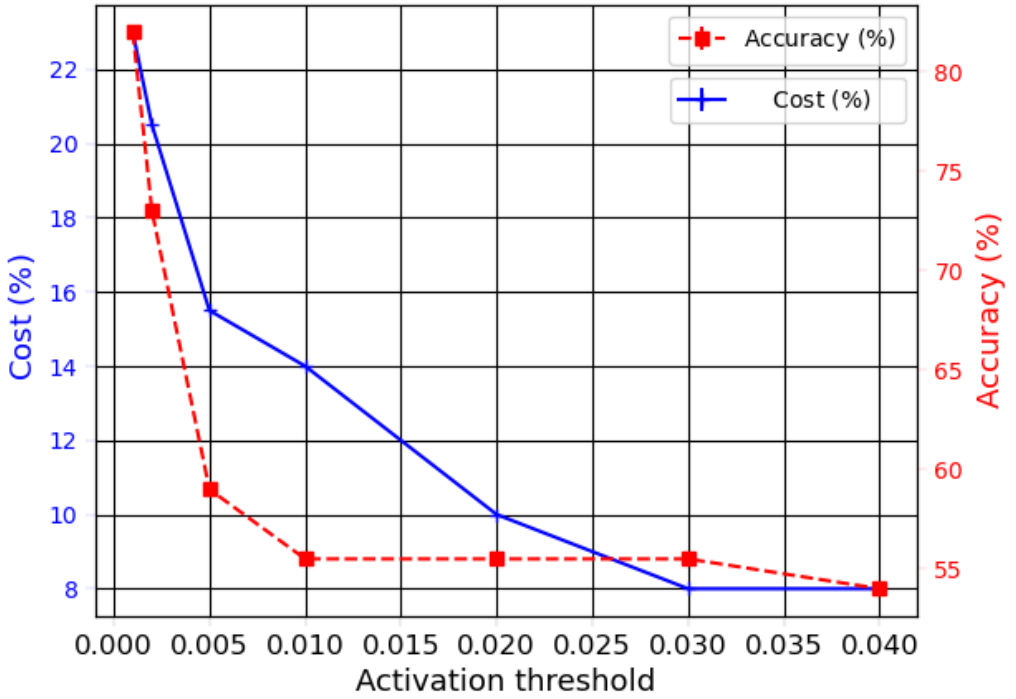
En outre, une attention particulière a été accordée à l'évaluation du coût, synonyme de surcharge, inhérent au modèle *Mc-DLN*. Il s'agit là du pourcentage de switchs virtuels (OVSs) supplémentaires activés dans la construction des graphes de routage multi-chemins, en comparaison avec ceux générés par FPR.

Cette analyse permet d'appréhender de manière exhaustive l'impact du modèle *Mc-DLN* en termes de consommation de ressources, soulignant ainsi les différences majeures en matière de surcharge entre les deux approches.

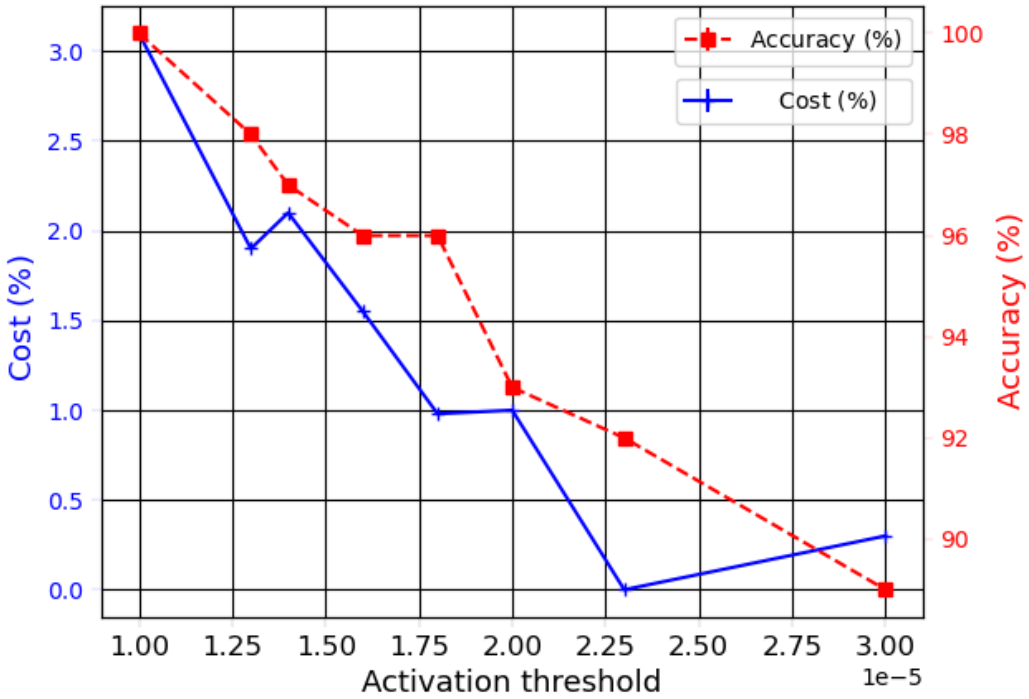
Enfin, et non des moindres, nous avons procédé à une évaluation comparative du coût induit par *iPare*, mesuré en termes du nombre de switchs virtuels (OVSs) nécessitant une activation, en comparaison avec les solutions FPR et HPR.

Dans notre démarche, nous avons mené une série de 100 répétitions en faisant varier les emplacements des clients ainsi que la topologie sous-jacente du réseau.

Par la suite, l'ensemble des résultats de simulation présentent la moyenne observée ainsi que l'intervalle de confiance à 95% (*confidence interval*, en anglais), offrant ainsi une vision solide relatives aux diverses méthodes en comparaison.



(a) Nombre de clients 5



(b) Nombre de clients 10

FIGURE 5.6 – L'effet du seuil d'activation (*Activation threshold*) sur la précision (*Accuracy*) et la surcharge (*Overhead*) d'iPare pendant le mode d'inférence.

Les Figures 5.3, 5.4 et 5.5 présentent les performances fondamentales d'*iPare*, évaluées en termes du temps d'exécution, en faisant varier respectivement le nombre de clients, d'OvSs et de serveurs.

Cette visualisation conduit à une première observation qui peut être déduite de cette illustration : à savoir, que tant *iPare* que HPR se distinguent par un temps d'exécution quasi-nul en comparaison de FPR [89], quelle que soit la configuration particulière en cours, c'est-à-dire le nombre de clients, d'OVS et de serveurs

En revanche, nous constatons également que : contrairement à *iPare* et HPR, les nombres de clients, d'OVS et de serveurs ont un impact négatif sur le temps d'exécution de FPR.

Cette disparité peut être expliquée en discernant que la stratégie sous-jacente à FPR, désireuse d'atteindre une configuration optimale par le biais de la méthode de *branch and bound*, entrave de manière significative les performances en termes de temps d'exécution (autrement dit, la dimension de scalabilité) au fur et à mesure que l'ampleur et la complexité du réseau s'accroissent.

De ce fait, les possibilités d'application de FPR se trouvent restreintes au sein d'un réseau sujet à des modifications fréquentes, ce qui peut compromettre son efficacité et sa flexibilité dans de telles conditions.

Contrairement à l'algorithme FPR, dont le temps d'exécution croît de manière exponentielle en fonction du nombre de clients, les temps d'exécution associés aux approches *iPare* et HPR se maintiennent à une constance remarquable, atteignant des valeurs extrêmement proches de zéro, comme en témoigne la Figure 5.3.

Cette observation met en lumière l'efficacité de *iPare* en matière de prise de décision optimale dans un délai court.

De façon similaire, les conclusions à tirer des résultats obtenus dans les Figures 5.4 et 5.5 apparaissent clairement : la solution iPare et les solutions HPR opèrent de manière indépendante du nombre de switchs virtuels (OvSs) et de serveurs.

De cette observation, nous déduisons que le temps nécessaire à l'exécution de FPR est lié au nombre de nœuds présents dans les réseaux (clients, switchs virtuels et serveurs) ainsi qu'à leurs degrés respectifs. En revanche, le temps d'exécution de la solution iPare demeure indépendant de ces facteurs et reste proche de zéro.

Nous observons également que la solution HPR est légèrement influencée par le nombre de clients. Cela démontre l'efficacité de iPare dans la mise à disposition rapide d'une configuration optimale, conforme à son objectif principal de conception.

En outre, les résultats obtenus dans les expérimentations présentées dans les Figure 5.3, 5.4 et 5.5 montrent que les performances en termes de temps d'exécution se rapprochent de celles de la solution heuristique HPR, laquelle produit des solutions de moindre qualité que celles générées par iPare.

Incontestablement, tel qu'illustré dans les Figures 5.7, 5.8 et 5.9, il s'avère évident que la solution iPare présente non seulement une précision accrue, mais également qu'elle offre des configurations optimales en activant un nombre réduit de switchs virtuels (OvS), en comparaison avec l'approche HPR.

Les résultats obtenus mettent en évidence l'efficacité d'*iPare* en termes d'obtention de configurations optimales dans un intervalle de temps raisonnable.

De surcroît, selon le framework DLO-SFC proposé, le HPR pourrait être utilisé par le composant heuristique (HC) dans les situations où la solution d'intelligence artificielle générée par iPare ne serait pas validée par le vérificateur de modèle (MC), témoignant ainsi de la polyvalence et de l'interaction de ces mécanismes algorithmiques soigneusement conçus.

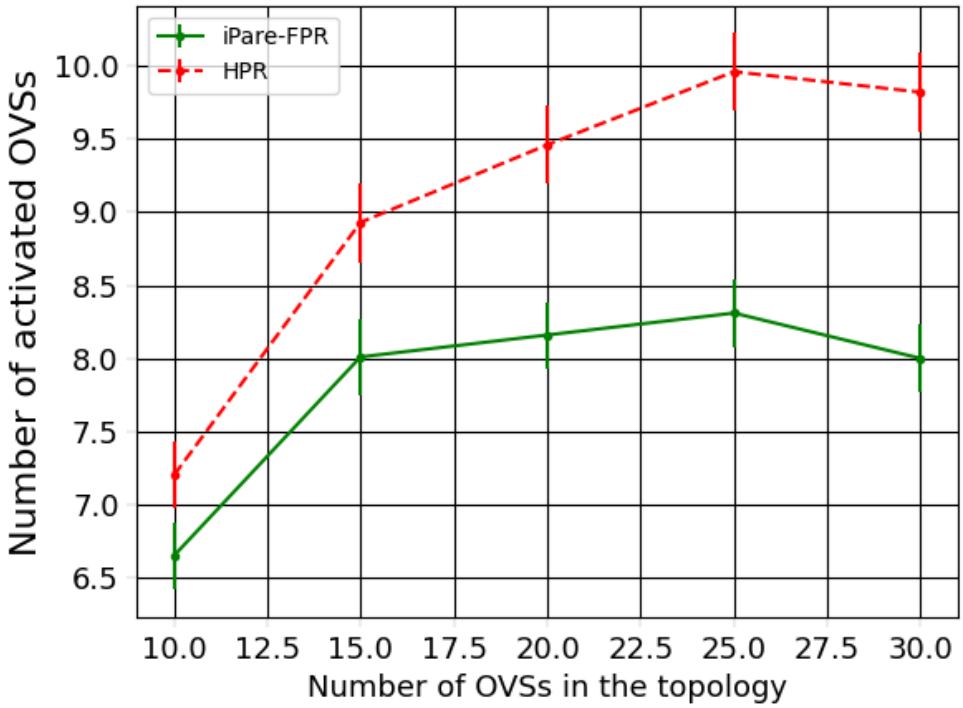


FIGURE 5.7 – Comparaison avec la solution heuristique *HPR* en fonction de la variation du nombre de OvSs.

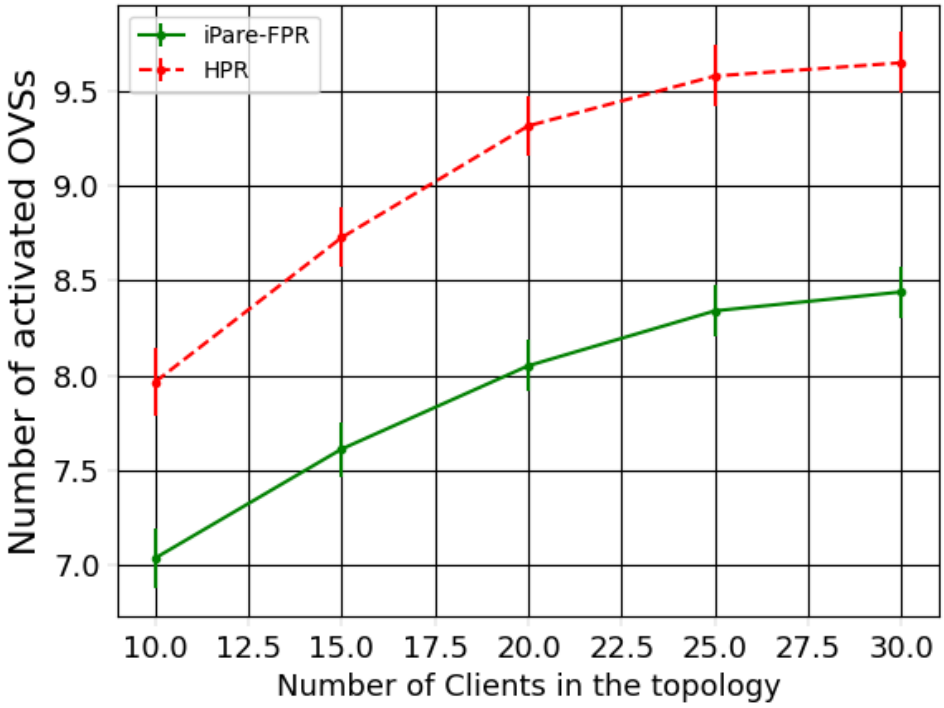


FIGURE 5.8 – Comparaison avec la solution heuristique *HPR* en fonction de la variation du nombre de clients.

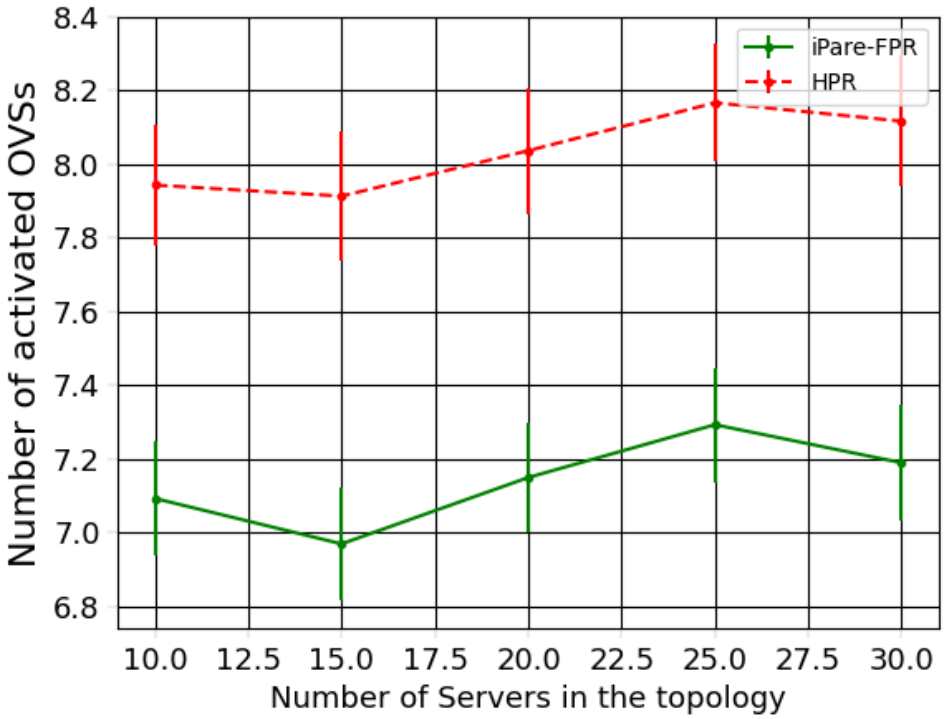


FIGURE 5.9 – Comparaison avec la solution heuristique *HPR* en fonction de la variation du nombre de serveurs.

L'illustration présentée dans la Figure 5.6 montre l'impact du seuil d'activation ϵ sur la surcharge, c'est-à-dire le coût, ainsi que sur la précision. Au sein de cette relation, émerge un compromis entre la précision et la surcharge.

Il est observable que l'augmentation de la valeur de ϵ manifeste un effet positif pour la réduction du coût, mais s'accompagne d'un effet négatif sur la précision. Si nous choisissons une valeur de ϵ proche de zéro, davantage de switchs virtuels OvS seront activés par rapport à FPR. Cette configuration induit un impact négatif sur le coût.

Néanmoins, cette augmentation engendre des effets positifs sur la précision, car elle nous permet de créer un graphe de routage multi-chemin plus efficace, garantissant l'atteinte de l'objectif de service désiré, également appelé *SLA*.

Cette tendance se révèle clairement dans la figure, indépendamment du nombre de clients considéré, soit au nombre de 5 ou de 10. En effet, l'augmentation de la valeur de ϵ améliore le coût, mais a parallèlement un impact négatif sur la précision.

À partir de l'analyse de la Figure 5.6(a), il est évident qu'avec un échantillon de 5 clients, 3 serveurs, 20 OvS, et en fixant le seuil d'erreur ϵ à une valeur de 5×10^{-3}, il est possible d'atteindre un niveau de précision équivalent à 80%.

Cette performance est obtenue tout en ayant activé un pourcentage supplémentaire de 22.5% de switchs par rapport à la configuration optimale préconisée par l'algorithme FPR.

Il est également important de relever que lorsque la valeur de ϵ est égale à 0.04, la précision connaît une chute significative, s'établissant à 55%, tandis que le surcoût supplémentaire atteint 7.5%. Les implications déduites de ces résultats viennent corroborer les hypothèses sous-jacentes à notre démarche.

En parallèle, l'analyse de la Figure 5.6(b) démontre que, pour un scénario impliquant 10 clients, 3 serveurs et un ensemble de 20 switchs OvS, avec une valeur de ϵ fixée à 10^{-5}, la solution *iPare* parvient à atteindre un niveau de précision exemplaire de 100%, et ce en supportant un surcoût additionnel de l'ordre de 3% par rapport à l'approche FPR.

Toutefois, il convient de noter que la précision et les coûts chutent respectivement à 89% et 0% lorsque ϵ est augmenté à 3×10^{-5}.

Les Figures 5.7, 5.8 et 5.9 offrent une illustration de la manière dont le nombre d'OvS, de clients et de serveurs exerce un impact significatif sur les coûts associés aux solutions FPR, HPR, et iPare.

Une première observation que nous pouvons déduire de ces figures est que tant la méthode FPR que la solution iPare parviennent à fournir une configuration optimale en activant un nombre minimal de switchs OvS.

En conséquence, ces deux méthodes surpassent de manière incontestable l'approche HPR en matière de coûts, avec une amélioration de plus de 20%.

À travers ces résultats, il est possible de constater que la solution iPare parvient à réaliser un niveau de précision élevée, tout en supportant un surcoût supplémentaire minime, et ce, tout en affichant un temps d'exécution quasi nul.

L'examen des conclusions tirées de ces résultats, comme en témoignent les Figures 5.3, 5.4, 5.5, 5.6, 5.7, 5.8 et 5.9, confirme clairement que la conception fondamentale de la solution iPare a atteint son objectif principal.

A cet effet, la solution iPare parvient à établir des configurations optimales dans un délai raisonnable, surpassant significativement l'algorithme FPR en matière de temps d'exécution et surclassant également l'approche HPR en termes de coût.

Enfin, il convient de souligner que le temps d'exécution de iPare s'aligne de manière similaire avec celui de HPR, tout en fournissant des configurations optimales, à l'instar de FPR. Cela atteste de son efficacité dans la conciliation entre précision, coût et temps d'exécution.

5.5 Conclusion

Le présent chapitre a été consacré à l'étude expérimentale de la solution iPare visant à fournir des configurations optimales de réseau dans un environnement de Software-Defined Networking (SDN).

La solution iPare repose sur un réseau de deep learning en classification multi-étiquettes (*Mc-DLN*) et vise à garantir un niveau de service (SLA) désiré tout en minimisant les coûts d'exploitation (OpEx).

Pour atteindre cet objectif, iPare réduit le nombre de switchs actifs nécessaires pour acheminer le trafic, en utilisant une approche d'apprentissage profond.

L'évaluation de la performance a été menée à travers une série d'expérimentations rigoureuses, abordant tant la phase d'entraînement que la phase d'inférence du modèle *Mc-DLN*.

Pour ce faire, nous avons utilisé un environnement de simulation par le biais du framework *Tensorflow*, prenant en compte une diversité de configurations de réseau et de charges de trafic. L'ensemble des données a été généré en utilisant l'algorithme de *Full Path Recomputation (FPR)*, qui permet d'obtenir des configurations optimales en matière d'optimisation réseau en se basant sur la programmation linéaire en nombres entiers.

Dans la phase d'entraînement, les performances du modèle *Mc-DLN* ont été minutieusement évaluées en fonction de divers paramètres, notamment le nombre de clients.

Les résultats ont démontré la capacité de iPare à converger rapidement vers des configurations optimales, réalisant un taux de précision de 100% et une perte quasi nulle.

Ces résultats, validés par comparaison avec les configurations optimales générées par FPR, ont témoigné de l'efficacité de iPare en tant que solution d'apprentissage profond pour la configuration rapide et précise des réseaux SDN.

La phase d'inférence a été consacrée à la comparaison des performances d'iPare avec les approches FPR et HPR *(Heuristic Path Re-computation)*, en termes de temps d'exécution, de précision, de surcharge et de coût.

Les résultats ont illustré la supériorité d'iPare en termes de temps d'exécution, avec des performances proches de zéro, à l'opposé du caractère chronophage de l'exécution de FPR. En outre, iPare s'est révélé plus précis et économiquement efficace que HPR, surpassant ces deux approches de manière significative.

L'analyse de l'effet du seuil ϵ a révélé un compromis inhérent entre précision et surcoût, avec des résultats indiquant que iPare peut atteindre des niveaux élevés de précision tout en maintenant un surcoût supplémentaire limité.

Cependant, l'évaluation de l'impact du nombre de switchs, de clients et de serveurs a démontré que iPare et FPR peuvent parvenir à des configurations optimales en activant un nombre minimal de switchs.

En conclusion, cette étude expérimentale a consolidé la pertinence et l'efficacité de la solution iPare en matière de configuration optimale de réseau SDN.

Les résultats obtenus soutiennent l'efficacité de l'approche d'apprentissage profond de iPare pour atteindre des configurations optimales rapides, précises et économiquement avantageuses, tout en surpassant les contraintes temporelles et de coût inhérentes aux solutions FPR et HPR.

Ces conclusions renforcent la crédibilité d'iPare en tant qu'outil puissant pour la conception et la gestion de réseaux SDN performants et économiques.

Conclusion et perspectives

L'esprit qui invente est toujours mécontent de ses progrès, parce qu'il voit au-delà.
Jean Le Rond d'Alembert (1746-1757)

Dans ce dernier chapitre, nous dressons le bilan du travail effectué dans cette thèse, en mettant en lumière les principaux résultats et conclusions obtenus. Nous introduisons également les perspectives futures qui peuvent émerger de cette recherche, ouvrant ainsi de nouvelles voies pour des études ultérieures dans ce domaine passionnant.

Conclusion

La présente thèse propose une étude sur les avancées majeures dans le domaine des réseaux, mettant particulièrement l'accent sur l'évolution des réseaux définis par logiciel (SDN) et l'intégration de l'intelligence artificielle (IA) dans ces environnements, dans le but d'optimiser et de gérer efficacement les réseaux modernes.

À travers cinq chapitres distincts, nous avons examiné divers aspects de cette convergence entre SDN et l'IA, allant de l'optimisation du routage multi-chemins à l'utilisation de l'apprentissage automatique, en passant par le développement d'un framework d'optimisation basé sur l'IA, le routage multi-chemins intelligent et une étude expérimentale approfondie.

Dans le premier chapitre, nous avons retracé l'évolution du SDN depuis ses origines dans les années 1990 jusqu'à son émergence en tant que solution de gestion de réseau révolutionnaire, offrant une flexibilité et une programmabilité inégalées, répondant ainsi aux exigences croissantes des applications et des utilisateurs. Le chapitre a également souligné l'importance de l'optimisation du routage multi-chemins au sein des réseaux SDN pour améliorer la qualité de service, la fiabilité et l'efficacité opérationnelle.

Le deuxième chapitre a examiné comment l'IA, en particulier l'apprentissage automatique et le deep learning, transforme les réseaux SDN en entités intelligentes et adaptables, capables de prendre des décisions autonomes en temps réel pour optimiser la gestion du trafic, améliorer la sécurité et réduire les coûts opérationnels (OpEx).

Le troisième chapitre a introduit un framework d'optimisation basé sur l'IA, offrant une solution originale pour la configuration efficace et autonome des réseaux SDN. Ce chapitre

a également mis en avant la praticité et l'efficacité du déploiement de ce framework.

Dans le quatrième chapitre, nous avons exploré les stratégies de routage multi-chemins intelligent, démontrant comment ces approches améliorent les performances et l'adaptabilité des réseaux dans les environnements SDN.

Enfin, le cinquième et dernier chapitre offre une étude expérimentale approfondie de la solution iPare, qui vise à fournir des configurations optimales de réseau dans un environnement SDN. Cette étude met en évidence l'efficacité de l'approche d'apprentissage profond de iPare en termes de rapidité, de précision et d'efficacité économique.

Au terme de ce travail, nous avons enrichi notre compréhension du domaine des réseaux SDN, un domaine de recherche toujours aussi captivant. Ce secteur est en constante évolution pour répondre aux demandes croissantes liées à l'exploitation massive des données dans les réseaux, notamment les réseaux mobiles. L'intégration de l'intelligence artificielle dans les SDN ouvre la voie à des réseaux plus intelligents, adaptatifs et efficaces.

A travers nos travaux, nous espérons avoir contribué de manière significative à l'avancement du domaine de la recherche scientifique.

Perspectives

À la lumière de notre travail, de nouvelles perspectives de recherche émergent, offrant des pistes prometteuses pour explorer de nouveaux horizons dans ce domaine dynamique.

Ci-après, quelques perspectives potentielles qui pourraient découler de notre thèse sur les réseaux automatisés basés sur la technologie SDN :

Appliquer d'autres approches sur le réseau basé sur SDN

Une perspective prometteuse qui consiste à explorer et à appliquer d'autres approches et techniques aux réseaux SDN. Cela pourrait inclure l'utilisation de méthodes d'optimisation différentes, telles que les algorithmes évolutifs, ou même des approches bio-inspirées comme les essaims particulaires ou les algorithmes génétiques, ou encore les techniques de réseaux de neurones artificiels plus récentes comme le Generative Adversarial Network (GAN).

Effectuer des expérimentations réelles à plus grande échelle de l'étude

Une perspective importante qui consiste à étendre les expérimentations à des échelles plus grandes dans des environnements réels. Cela pourrait impliquer le déploiement de solutions SDN dans des réseaux de taille réelle, tels que des infrastructures de centres de données des opérateurs de téléphonie mobile. Ces expérimentations à grande échelle permettraient de mieux comprendre les défis et les opportunités rencontrés dans des environnements réels.

Intégrer de l'automatisation dans les opérations réseau

Une perspective essentielle qui consiste à explorer davantage l'intégration de l'automatisation dans tous les aspects des opérations réseau, allant de la configuration initiale à la

gestion quotidienne et à la résolution des problèmes. L'automatisation est un domaine en plein essor dans les réseaux. Dans ce contexte, les réseaux définis par logiciel (SDN) offrent un cadre idéal pour la mise en œuvre d'opérations réseau automatisées.

Bibliographie

[1] Thomas Rincy N and Roopam Gupta. A survey on machine learning approaches and its techniques. In *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, pages 1–6, 2020.

[2] Inderjeet Kaur, Anupama Sharma, Amita Agnihotri, and Charu Agarwal. *Software-Defined Networks : Perspectives and Applications*, chapter 2, pages 29–61. John Wiley & Sons, Ltd, 2022.

[3] Cisco. Cisco Annual Internet Report, 2018–2023, March 2020.

[4] Doug Austin. 2023 Internet Minute Infographic, April 2023. Accessed on May 20, 2023.

[5] Abdelhakim Nacef, Abdellah Kaci, Youcef Aklouf, and Diego Leonel Cadette Dutra. Machine learning based fast self optimized and life cycle management network. *Computer Networks*, 209 :108895, May 2022.

[6] Quadri Waseem, Wan Isni Sofiah Wan Din, Afrig Aminuddin, Muzammil Hussain Mohammed, and Rifda Faticha Alfa Aziza. Software-defined networking (sdn) : A review. In *2022 5th International Conference on Information and Communications Technology (ICOIACT)*, pages 30–35, 2022.

[7] Huawei Technologies Co. Ltd. *SDN and NFV*, pages 521–535. Springer Nature Singapore, Singapore, 2023.

[8] Xenofon Foukas, Mahesh Marina, and Kimon Kontovasilis. *Software Defined Networking Concepts*, pages 21–44. John Wiley & Sons, 06 2015.

[9] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow : Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2) :69–74, March 2008.

[10] Huawei Technologies Co. Ltd. Sdn and nfv. Data Communications and Network Technologies pp 521–535, Oct 2022. Retrieved September 23, 2023. from `https://link.springer.com/chapter/10.1007/978-981-19-3029-4_17`.

[11] Konstantinos Poularakis, Leandros Tassiulas, and T.V. Lakshman. Modeling and optimization in software-defined networks. *Synthesis Lectures on Learning, Networks, and Algorithms*, 2 :1–174, 10 2021.

[12] Konstantinos Psounis. Active networks : Applications, security, safety, and architectures. *IEEE Communications Surveys*, 2(1) :2–16, 1999.

[13] Preet Sanghavi, Sheel Sanghvi, and Ramchandra S. Mangrulkar. *Software-Defined Networks A Brief Overview and Survey of Services*, pages 1–31. Springer International Publishing, 2022.

[14] Subhra Priyadarshini Biswal and Sanjeev Patel. *Introduction to Software Defined Networking*, chapter 1, pages 1–27. John Wiley & Sons, Ltd, 2022.

[15] Open Networking Foundation. SDN Architecture Issue 1.1. Technical report, TR-521, 2016.

[16] Dharmik Lunagariya and Bhargavi Goswami. A comparative performance analysis of stellar sdn controllers using emulators. In *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, pages 1–9, 2021.

[17] Liehuang Zhu, Md M Karim, Kashif Sharif, Chang Xu, Fan Li, Xiaojiang Du, and Mohsen Guizani. Sdn controllers : A comprehensive analysis and performance evaluation study. *ACM Computing Surveys (CSUR)*, 53(6) :1–40, 2020.

[18] Chander Prabha, Anjuli Goel, and Jaspreet Singh. A survey on sdn controller evolution : A brief review. In *2022 7th International Conference on Communication and Electronics Systems (ICCES)*, pages 569–575, 2022.

[19] David Erickson. The beacon openflow controller. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 13–18, 2013.

[20] Atlassian. Floodlight controller. Retrieved June 01, 2023 from `https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview`.

[21] Byungjoon Lee, Sae Hyong Park, Jisoo Shin, and Sunhee Yang. Iris : The openflow-based recursive sdn controller. In *16th International Conference on Advanced Communication Technology*, pages 1227–1231, 2014.

[22] Byungjoon Lee, Jisoo Shin, and Sae Hyong Park. Iris, Augut 2015. Retrieved May 23, 2023 from `https://github.com/openiris/IRIS`.

[23] Zheng Cai, Alan L. Cox, and T. S. Eugene Ng. Maestro : A system for scalable openflow control. *Rice University, Houston, TX, USA, TSEN Maestro-Techn. Rep, TR10-08*, 2010.

[24] Amin Tootoonchian. The nox controller, February 2014. Retrieved May 22, 2023 from `https://github.com/noxrepo/nox`.

[25] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, and Guru Parulkar. Onos : Towards an open, distributed sdn os. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, page 1–6, New York, NY, USA, 2014. Association for Computing Machinery.

[26] Linux Foundation. Opendaylight. Retrieved May 16, 2023 from `https://www.opendaylight.org/`.

[27] POX. Pox documentation. Retrieved May 13, 2023 from `https://noxrepo.github.io/pox-doc/html/`.

[28] Rui Kubo, Tomonori Fujita, Yuji Agawa, and Hikaru Suzuki. Ryu sdn framework-open-source sdn platform software. *NTT Tech. Rev*, 12(8) :1–5, 2014.

[29] Zhihao Wu, Xulu Fan, and Khaled Harfoush. Measurement-driven flow selection for open vswitch offload. In *ICC 2022 - IEEE International Conference on Communications*, pages 3635–3640, 2022.

[30] Open Networking Foundation. Open network operating system (onos). Retrieved May 22, 2023 from `https://opennetworking.org/onos/`.

[31] Mininet Project Contributors. Mininet overview. Retrieved June 16, 2023 from `https://mininet.org/overview/`.

[32] Open Networking Foundation. OpenFlow Switch Specification Version 1.5.1. Technical report, ONF TS-025, March 2015.

[33] Linux Foundation. Open vswitch. Retrieved May 28, 2023 from `https://www.openvswitch.org/features/`.

[34] Open Networking Foundation. Openflow-enabled sdn and network functions virtualization. Technical report, ONF Solution Brief, February 2014.

[35] Besmir Tola and Yuming Jiang. Modeling and provisioning highly available nfv services. In *2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 117–123, 2022.

[36] ETSI GS NFV 002. Network functions virtualization (NFV) ; architectural framework v1.2.1. Technical report, ETSI, December 2014.

[37] Joel M. Halpern and Carlos Pignataro. Service Function Chaining (SFC) Architecture. RFC 7665, October 2015.

[38] Karamjeet Kaur, Veenu Mangat, and Krishan Kumar. A comprehensive survey of service function chain provisioning approaches in sdn and nfv architecture. *Computer Science Review*, 38 :100298, 2020.

[39] Ilhem Boussaid. *Perfectionnement de métaheuristiques pour l'optimisation continue*. Theses, Université Paris-Est ; Université des Sciences et de la Technologie Houari-Boumediène (Alger), June 2013.

[40] Max Cerf. *Techniques d'optimisation*. EDP Sciences, 2022.

[41] Adrien Gausseran. *Optimization algorithms for network slicing for 5G*. Theses, Université Côte d'Azur, November 2021.

[42] Hilmi E. Egilmez and A. Murat Tekalp. Distributed qos architectures for multimedia streaming over software defined networks. *IEEE Transactions on Multimedia*, 16(6) :1597–1609, 2014.

[43] Sachin Sharma, Dimitri Staessens, Didier Colle, David Palma, João Gonçalves, Ricardo Figueiredo, Donal Morris, Mario Pickavet, and Piet Demeester. Implementing quality of service for the software defined networking enabled future internet. In *2014 Third European Workshop on Software Defined Networks*, pages 49–54, 2014.

[44] Faïz Gallouj, Camal Gallouj, Marie-Christine Monnoyer, Luis Rubalcaba, and Markus Scheuer. *Elgar Encyclopedia of Services*. Edward Elgar Publishing Limited, Cheltenham, UK, 2023.

[45] Tao Wang, Fangming Liu, and Hong Xu. An efficient online algorithm for dynamic sdn controller assignment in data center networks. *IEEE/ACM Transactions on Networking*, 25(5) :2788–2801, 2017.

[46] Slavica Tomovic and Igor Radusinovic. Toward a scalable, robust, and qos-aware virtual-link provisioning in sdn-based isp networks. *IEEE Transactions on Network and Service Management*, 16(3) :1032–1045, 2019.

[47] Xu Xiaolong, Chen Yun, Hu Liuyun, and Kumar Anup. Mtss : Multi-path traffic scheduling mechanism based on sdn. *Journal of Systems Engineering and Electronics*, 30(5) :974–984, 2019.

[48] M. Rasih Celenlioglu and H. Ali Mantar. An sdn based intra-domain routing and resource management model. In *2015 IEEE International Conference on Cloud Engineering*, pages 347–352, 2015.

[49] Jinyao Yan, Hailong Zhang, Qianjun Shuai, Bo Liu, and Xiao Guo. Hiqos : An sdn-based multipath qos solution. *China Communications*, 12(5) :123–133, May 2015.

[50] Sahel Sahhaf, Wouter Tavernier, Didier Colle, and Mario Pickavet. Adaptive and reliable multipath provisioning for media transfer in sdn-based overlay networks. *Computer Communications*, 106 :107–116, July 2017.

[51] Syed Asad Hussain, Shuja Akbar, and Imran Raza. A dynamic multipath scheduling protocol (dmsp) for full performance isolation of links in software defined networking (sdn). In *2017 2nd Workshop on Recent Trends in Telecommunications Research (RTTR)*, pages 1–5, Feb 2017.

[52] Abdul Basit, Saad Qaisar, Syed Hamid Rasool, and Mudassar Ali. Sdn orchestration for next generation inter-networking : A multipath forwarding approach. *IEEE Access*, 5 :13077–13089, March 2017.

[53] Che Huang, Chawanat Nakasan, Kohei Ichikawa, and Hajimu Iida. A multipath controller for accelerating gridftp transfer over sdn. In *2015 IEEE 11th International Conference on e-Science*, pages 439–447, Munich, Germany, Aug 2015.

[54] Che Huang, Chawanat Nakasan, Kohei Ichikawa, and Hajimu Iida. An sdn-based multipath gridftp for high-speed data transfer. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 763–764, Nara, Japan, June 2016.

[55] Mingjian Fu and Fan Wu. Investigation of multipath routing algorithms in software defined networking. In *2017 International Conference on Green Informatics (ICGI)*, pages 269–273, August 15-17 2017.

[56] Luis Guillen, Satoru Izumi, Toru Abe, Takuo Suganuma, and Hiroaki Muraoka. Sdn-based hybrid server and link load balancing in multipath distributed storage systems. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, April 23-27 2018.

[57] Yunchong Guan, Weimin Lei, Wei Zhang, Shaowei Liu, and Hao Li. Scalable orchestration of software defined service overlay network for multipath transmission. In *Computer Networks*, volume 137, pages 132–146, June 2018.

[58] Lely Hiryanto, Sieteng Soh, Kwan-Wu Chin, Duc-Son Pham, and Mihai Lazarescu. Green multi-stage upgrade for bundled-links sdn/ospf-ecmp networks. In *IEEE International Conference on Communications (ICC), 2021*, pages 1–7. IEEE, 2021.

[59] Marjan Mahmoudi, Avid Avokh, and Behrang Barekatain. Sdn-dvfs : an enhanced qos-aware load-balancing method in software defined networks. *Cluster Computing*, pages 1–26, 2022.

[60] Srinivasan Dwarakanathan, Len Bass, and Liming Zhu. Cloud application ha using sdn to ensure qos. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 1003–1007, June 2015.

[61] Min Sang Yoon and Ahmed E. Kamal. Power minimization in fat-tree sdn datacenter operation. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, December 2015.

[62] Sana Tariq and Mostafa Bassiouni. Qamo-sdn : Qos aware multipath tcp for software defined optical networks. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 485–491, 2015.

[63] Qingtian Wang, Guochu Shou, Yaqiong Liu, Yihong Hu, Zhigang Guo, and Wei Chang. Implementation of multipath network virtualization with sdn and nfv. *IEEE Access*, 6 :32460–32470, May 2018.

[64] Stuart Russell and Peter Norvig. *Artificial Intelligence : A Modern Approach*. Pearson Education, United Kingdom, 4th edition, 2022.

[65] Le Monde avec AFP. L'intelligence artificielle alphago bat une nouvelle fois le champion du monde de go. Le Monde, May 2017. Retrieved July 05, 2023. from `https://www.lemonde.fr/pixels/article/2017/05/25/l-intelligence-artificielle-alphago-bat-une-nouvelle-fois-le-champ 5133716_4408996.html`.

[66] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[67] Andriy Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.

[68] Abdellah Kaci and Abderrezak Rachedi. Toward a machine learning and software defined network approaches to manage miners' reputation in blockchain. *Journal of Network and Systems Management*, 28, 07 2020.

[69] Guillaume Saint-Cirgue. *Apprendre le Machine Learning en une semaine*. Guillaume Saint-Cirgue, eBook, 2019. `https://machinelearnia.com`.

[70] Mohamed Amine Ferrag, Leandros Maglaras, Sotiris Moschoyiannis, and Helge Janicke. Deep learning for cyber security intrusion detection : Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50 :102419, 2020.

[71] Eugene Charniak. *Introduction au Deep Learning*. Dunod, 2021.

[72] Chloé-Agathe Azencott. *Introduction au Machine Learning*. Dunod, 2019.

[73] Teik Toe Teoh and Zheng Rong. *Convolutional Neural Networks*, pages 261–275. Springer Singapore, Singapore, 2022.

[74] Sandro Skansi. *Introduction to Deep Learning : from logical calculus to artificial intelligence*. Springer, 01 2018.

[75] Whai-En Chen, Xiang-Yuan Fan, and Li-Xian Chen. A cnn-based packet classification of embb, mmtc and urllc applications for 5g. In *2019 International Conference on Intelligent Computing and its Emerging Applications (ICEA)*, pages 140–145, 2019.

[76] Xiangle Cheng, Yulei Wu, Geyong Min, Albert Zomaya, and Xuming Fang. Safeguard network slicing in 5g : A learning augmented optimization approach. *IEEE Journal on Selected Areas in Communications*, 2020.

[77] Jianing Pei, Peilin Hong, Kaiping Xue, Defang Li, David S. L. Wei, and Feng Wu. Two-phase virtual network function selection and chaining algorithm based on deep learning in sdn/nfv-enabled networks. *IEEE Journal on Selected Areas in Communications*, 38(6) :1102–1117, April 2020.

[78] Hee-Gon Kim, Se-Yeon Jeong, Do-Young Lee, Heeyoul Choi, Jae-Hyung Yoo, and James Won-Ki Hong. A deep learning approach to vnf resource prediction using correlation between vnfs. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 444–449, June 2019.

[79] El Hocine Bouzidi, Abdelkader Outtagarts, and Rami Langar. Deep reinforcement learning application for network latency management in software defined networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, December 2019.

[80] Beakal Gizachew Assefa and Öznur Özkasap. Hymer : A hybrid machine learning framework for energy efficient routing in SDN. *CoRR*, abs/1909.08074, 2019.

[81] Mutaz Hamed Hussien Khairi, Sharifah Hafizah Syed Ariffin, Nurul Mu'Azzah Abdul Latiff, Kamaludin Mohamad Yusof, Mohamed Khalafalla Hassan, Fahad Taha Al-Dhief, Mosab Hamdan, Suleman Khan, and Muzaffar Hamzah. Detection and classification of conflict flows in sdn using machine learning algorithms. *IEEE Access*, 9 :76024–76037, 2021.

[82] Afsaneh Banitalebi Dehkordi, MohammadReza Soltanaghaei, and Farsad Zamani Boroujeni. The ddos attacks detection through machine learning and statistical methods in sdn. *The Journal of Supercomputing*, 77(3) :2383–2415, March 2021.

[83] K. Muthamil Sudar, M. Beulah, P. Deepalakshmi, P. Nagaraj, and P. Chinnasamy. Detection of distributed denial of service attacks in sdn using machine learning techniques. In *2021 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5, January 2021.

[84] M. Revathi, V. V. Ramalingam, and B. Amutha. A machine learning based detection and mitigation of the ddos attack by using sdn controller framework. *Wireless Personal Communications*, pages 1–25, September 2021.

[85] Lei Wang and Declan T. Delaney. Qoe oriented cognitive network based on machine learning and sdn. In *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, pages 678–681, November 2019.

[86] Marlon Rodríguez, Ricardo Flores Moyano, Noel Pérez, Daniel Riofrío, and Diego Benítez. Path planning optimization in sdn using machine learning techniques. In *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, pages 1–6, 2021.

[87] Chafika Benzaid and Tarik Taleb. Ai-driven zero touch network and service management in 5g and beyond : Challenges and research directions. *IEEE Network*, 34(2) :186–194, 2020.

[88] ETSI. Zero-touch network and service management. `https://www.etsi.org/technologies/zero-touch-network-service-management`.

[89] Miloud Bagaa, Diego Leonel Cadette Dutra, Tarik Taleb, and Konstantinos Samdanis. On sdn-driven network optimization and qos aware routing using multiple paths. *IEEE Transactions on Wireless Communications*, 19(7) :4700–4714, 2020.

[90] The Linux Foundation. Kubernetes. `https://kubernetes.io`.

[91] The Linux Foundation. Prometheus. `https://prometheus.io/`.

[92] Open Networking Foundation (ONF). Sd-ran documentation. `https://docs.sd-ran.org/master/`.

[93] TensorFlow Community. Tensorflow. `https://tensorflow.org/`.

[94] Gurobi Optimization. Gurobi optimizer. `https://www.gurobi.com/`.

[95] MongoDB Inc. Mongodb. `https://www.mongodb.com`.

[96] NetworkX developers. Networkx. `https://networkx.org/`.

[97] Canonical Ltd. Ubuntu. `https://ubuntu.com/`.

[98] Abdelhakim Nacef, Miloud Bagaa, Youcef Aklouf, Abdellah Kaci, Diego Leonel Cadette Dutra, and Adlen Ksentini. Self-optimized network : When machine learning meets optimization. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, December 2021.

[99] Massinissa Ahman, Aziz Rechache, Abdellah Kaci, Oussama Annad, Abdelhakim Nacef, and Soraya Ait Chellouche. An approach for efficient vehicular tracking in internet of vehicles. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pages 953–954, 2023.

[100] Massinissa Ahman, Aziz Rechache, Abdellah Kaci, Oussama Annad, Abdelhakim Nacef, and Soraya Ait-Chellouche. Toward an efficient geographical routing protocol for internet of vehicles. In *Proceedings of the 2023 6th International Conference on Geoinformatics and Data Analysis*, ICGDA '23, page 53–59, New York, NY, USA, 2023. Association for Computing Machinery.

Annexe A

Publications et contributions scientifiques

A.1 Liste des publications

Journal international

[5] **Abdelhakim Nacef**, Abdellah Kaci, Youcef Aklouf and Diego Leonel Cadette Dutra, “Machine learning based fast self optimized and life cycle management network”, Computer Networks, Volume 209 :108895, 22 May 2022, doi : 10.1016/j.comnet.2022.108895.

Conférence internationale

[98] **Abdelhakim Nacef**, Miloud Bagaa, Youcef Aklouf, Abdellah Kaci, Diego Leonel Cadette Dutra, Adlen Ksentini, "Self-optimized network : When Machine Learning Meets Optimization," 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 07-11 December 2021, pp. 1-6, doi: 10.1109/GLOBECOM46510.2021.9685681.

A.2 Contributions scientifiques

Dans ce qui suit, les diverses participations en tant que co-auteur aux communications internationales :

[99] Massinissa Ahman, Aziz Rechache, Abdellah Kaci, Oussama Annad, **Abdelhakim Nacef**, and Soraya Ait Chellouche. An approach for efficient vehicular tracking in internet of vehicles. In 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), pages 953–954, 08-11 January 2023, doi: 10.1109/CCNC51644.2023.10060508.

[100] Massinissa Ahman, Aziz Rechache, Abdellah Kaci, Oussama Annad, **Abdelhakim Nacef**, and Soraya Ait-Chellouche. Toward an efficient geographical routing protocol for internet of vehicles. In Proceedings of the 2023 6th International Conference on Geoinformatics and Data Analysis, ICGDA ’23, page 53–59, New York, NY, USA, April 2023. ACM, doi: 10.1145/3606180.3606189.