From servies placement to services monitoring in 5G and post-5G networks

Anouar Rkhami

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Anouar RKHAMI

« From services placement to services monitoring in 5G and post-5G networks »

Thèse présentée et soutenue à « Rennes », le « 14 Decembre 2021 »
Unité de recherche : INRIA/IRISA
Thèse N° : « 2021/ »

Rapporteurs avant soutenance :

Thierry TURLETTI	Directeur de Recherche, INRIA Sophia Antipolis – Méditerranée
Mohamed FATEN ZHANI	Professeur, ETS Montreal, CANADA

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Présidente :	Géraldine TEXIER	Professeure, IMT Atlantique, Rennes, France
Examinateurs :	Yassine HADJADJ-AOUL	Maitre de conférences (HDR), Université de Rennes I
	Adlen KSENTINI	Professeur, Eurecom, Sophia Antipolis, France
	Abdelkader OUTTAGARTS	Chercheur Sénior, NOKIA BELL-LABS, France
	Nancy PERROT	Ingénieur de Recherche, Orange-Labs, Paris, France
	Kamal SINGH	Maitre de conférences, Telecom Saint Etienne / Université Jean Monnet
	Géraldine TEXIER	Professeure, IMT Atlantique, Rennes, France
Rapporteur :	Mohamed Faten ZHANI	Professeur, ETS Montreal, Canada
Dir. de thèse :	Gérardo RUBINO	Directeur de recherche, INRIA, Rennes Bretagne Atlantique

Abstract

5G and beyond 5G (B5G) networks are expected to accommodate a plethora of network services with diverse requirements using a single physical infrastructure. Hence, the "one-size fits all" paradigm that characterized the 4^{th} generation of wireless networks is no longer suitable. By leveraging the last advent of Network Function Virtualization (NFV) and Software Defined Networking (SDN), Network Slicing (NS) is considered as one of the key enablers of this paradigm shift. NS will enable the coexistence of heterogeneous services by partitioning the physical infrastructure into a set of virtual networks (i.e., slices), each running a particular service. Besides, NS offers more flexibility and agility into business operations.

Despite the advantages it brings, NS raises some technical challenges : The placement of network slices is one of them, and it is known in the literature as the Virtual Network Embedding Problem (VNEP), which is NP-Hard. Therefore, the first part of this thesis focuses on unveiling the potential of deep reinforcement learning (DRL) and graph neural networks to solve the network slices' placement problem and overcome the limitations of existing methods. Two approaches are considered : The first one aims to learn automatically how to solve the VNEP. Instead of putting any constraint on the topology of the physical infrastructure or to extract features manually, we formulate the task as a reinforcement learning problem, and we use a Graph convolutional based neural architecture to learn how to find an optimal solution. Next, instead of training a DRL agent from scratch to find the optimal solution, a process that may result in unsafe training, we train it to reduce the optimality gap of the existing heuristics. The motivation behind this contribution is to ensure safety during the training of the DRL agent.

The placement of the slices is not the only challenge raised by NS. Once the slices are placed, monitoring the status of network slices becomes a priority for both, network slices' tenants and providers to ensure that Service Level Agreements (SLAs) are not violated. In the second part of this thesis, we propose to leverage machine learning techniques and network tomography to monitor the network slices. Network Tomography (NT) is defined as a set of methods that aim to infer unmeasured network metrics using an end-to-end measurement between monitors.

We focus on two main challenges. First, on the inference of slices metrics based on some end-to-end measurements between monitors, as well as on the efficient placement of monitors. For the inference, we model the task as a multi-output regression problem, which we solve using neural networks. We propose to train on synthetic data to augment the diversity of the training data and avoid the overfitting issue. Moreover, to deal with the changes that may occur either on the slices we monitor or the topology on top of which they are placed, we use transfer learning techniques.

Regarding the monitors' placement problem, we consider a special case where only cycles' probes are allowed. The probing cycle schemes have a significant advantage compared to regular paths, since the source probe is actually the destination, which reduces the synchronization problems. We formulate the problem as a variant of the minimum set cover problem. Owing to its complexity, we introduce a standalone solution based on Graph Neural Networks (GNNs) and genetic algorithms to find a trade-off between the quality of monitors placement and the cost to achieve it.

Acknowledgement

It seems always impossible until it is done. This thesis could not be done without the support of several people. No word can express my gratitude towards all these people.

Firstly I would like to thank and express my sincere gratitude to my supervisors : Yassine Hadjadj-Aoul, Gerardo Rubino and Abdelkader Outtagarts who helped me during the past three years and guided me through the study and the writing of this thesis.

In addition to my supervisors, I would like to thank my thesis commitee : Nancy Perrot, Geraldine Texier, Kamal Singh, Thierry Turletti and Mohamed Faten Zhani for accepting to evaluate this thesis and to be part of the jury.

Besides, I am also grateful for being part of the DIONYSOS team. I would like to thank all the members of DIONYSOS team at IRISA/INRIA Rennes Research center. A special thanks goes to the team assistants : Fabienne Cuyoulla and Lannec Gwenaelle for their precious support in all administrative tasks.

I am grateful also to work with Ghina DANDACHI, Mohamed Rahali, and Pham Tran Anh Qang and to discuss with them different research topics.

Special thanks to my friends who shared with me this research experience and supports me along its ups and downs. In particular : Othamne Belmoukadam, Oussama Wadih ,Brahim Ahammou, Hajar Imami, Idriss Gotni ,Abderrahim Abellal, Ximun Castoreo, and Imane Taibi.

Last but not least, I would like to thank my wife Fatimazahra. She stood by my side through all my circumstances. She gave me support and help and tolerated my mood swings.

Finally, I would also like to thank my parents for their support during my Ph.D. journey and their support since I was born. It is your care and your prayers that make me sustain that far. Thanks also to my twin brother Walid who always encouraged and motivated me to never quit until I get everything I dreamed of.

Thank you.

Table of Contents

TABLE DES FIGURES

Liste des tableaux

Resumé en Français

Le début des années 70 marque le début de l'ère des réseaux mobiles sans fil. Depuis lors et sur quatre décennies, quatre générations de réseaux sans fil mobiles ont vu le jour, comme le montre la figure 1.1. La première génération (1G) a été introduite dans les années 1980 et, basée sur des technologies analogiques, elle n'offrait que la transmission vocale. Une décennie plus tard, et pour surmonter les limitations de la 1G, telles que les problèmes d'évolutivité et les capacités limitées, la deuxième génération (2G) a été introduite. Les réseaux 2G offraient de faibles débits binaires et permettaient la prise en charge de plusieurs utilisateurs en utilisant les technologies numériques et la méthode TDMA (Time-Division Multiple Access). Plus tard en 2001, et afin d'offrir des débits de données plus élevés, la troisième génération a été introduite. Les réseaux 3G étaient connus par la technique d'accès multiple par répartition en code (CDMA) et qu'ils permettent de prendre en charge les services Internet mobiles avec des débits de données plus élevés que la 2G. Une décennie plus tard encore, la quatrième génération (4G) de réseaux cellulaires est arrivée. Connu également sous le nom de réseaux d'évolution à long terme (LTE). La 4G est déployée partout dans le monde. En éliminant la commutation de circuits, la 4G l'a remplacée par un réseau IP basé sur la commutation par paquets pour prendre en charge des débits de données étendus et divers services mobiles à large bande. Depuis la mise en place des réseaux 4G et son adoption par les entreprises industrielles, le nombre d'appareils mobiles et Internet des objets (IoT) a considérablement augmenté. Cela a également conduit à une augmentation du volume de données de trafic mobile. Selon une récente prévision, on s'attendait à ce qu'il y ait une augmentation de 33 fois du trafic mobile dans le monde en 2020 par rapport à 2010 . Par conséquent, une nouvelle façon de gérer le réseau est nécessaire pour faire face à cette augmentation significative. La solution proposée était la cinquième génération (5G). Contrairement à toutes les autres générations, la 5G n'est pas une évolution de son antécédent. Au lieu de cela, il s'agit d'un changement de paradigme dans les réseaux sans fil mobiles. Les travaux décrits dans cette thèse portent sur les réseaux 5G et Beyond 5G (B5G).

Ce chapitre est organisé comme suit : Dans un premier temps, nous présentons le contexte de notre travail ainsi que ses motivations et les défis que nous visons à résoudre.

Ensuite, nous listons nos contributions, et enfin, nous présentons la structure de la thèse.

Contrairement aux autres générations sans fil mobiles où l'objectif principal était de satisfaire uniquement les exigences des communications interhumaines, les réseaux 5G devraient prendre en charge tous les types de communications (par exemple, machine à machine, humain à machine). De plus, les systèmes 5G s'attendent à répondre aux exigences de tous ces services en utilisant une seule infrastructure physique. Par conséquent, les réseaux 5G doivent avoir les fonctionnalités requises pour réaliser cette vision. Software Defined Networking (SDN) et NFV (Network Function Virtualization) sont l'une de ces fonctionnalités obligatoires. En effet, ces technologies sont à l'origine de l'émergence du concept de *Network Slicing*(NS) qui a été présenté comme l'un des catalyseurs critiques vers la réalisation de ce changement de vision. Pour permettre la coexistence de ces services hétérogènes, NS vise à créer plusieurs réseaux virtuels (slices), chacun dédié à répondre aux indicateurs de performance clés (KPI) d'un service particulier. Ainsi, grâce à NS, les opérateurs peuvent réduire leurs dépenses en capital (CAPEX) puisqu'ils partageront une seule infrastructure physique pour fournir plusieurs services. De plus, ils peuvent également réduire leurs Dépenses Opérationnelles (OPEX) grâce à la softwarization et la virtualisation du réseau.

Une telle révolution apporte beaucoup d'avantages mais aussi une pléthore de défis liés à la gestion et à la planification des réseaux 5G. En raison de la diversité des services qu'elles fournissent, la 5G nécessite également un changement dans les méthodes pour faire face à ces défis. L'objectif est de parvenir à une gestion entièrement automatisée des réseaux. Dès lors, l'Intelligence Artificielle apparaît comme une solution naturelle (IA). Les solutions d'IA sont conçues en général pour résoudre des défis complexes, ce qui est le cas pour le découpage de réseau. En effet, avec le déploiement d'un grand nombre de tranches différentes indépendantes les unes des autres, la complexité de leur gestion va augmenter. Parmi ces défis qui seront abordés dans cette thèse, on peut citer le placement des tranches de réseau et leur surveillance.

Le placement des tranches n'est pas un nouveau défi, il peut être considéré comme une variante du problème d'intégration de réseau virtuel qui est considéré dans la littérature comme un problème NP-Hard. Par conséquent, pour faire face à la complexité de la tâche et assurer l'automatisation en même temps, un algorithme d'approximation basé sur l'IA est nécessaire pour la résoudre. Une autre raison derrière le choix de l'IA comme solution est leur capacité à apprendre à fournir des solutions plus précises que les heuristiques conçues par l'homme.

Lorsqu'il s'agit de surveiller les tranches, il est essentiel d'assurer le bon fonctionnement des services gérés par les tranches et de respecter les accords de niveau de service négociés entre les fournisseurs de tranches et les locataires de tranches. Cependant, dans le contexte de la 5G, la surveillance des tranches devient plus difficile. En effet, une tranche peut être déployée au dessus de différents domaines qui sont administrés indépendamment. Par conséquent, la surveillance globale des tranches devient beaucoup plus complexe, en particulier lorsque des restrictions mutuelles entre domaines existent. Cela appelle alors de nouvelles solutions de surveillance qui évitent ces problèmes et s'appuient sur des mesures externes. Ces solutions sont appelées Network Tomography (NT). Ils sont bien étudiés dans la littérature et grâce à leur adéquation avec les paradigmes SDN et NFV [94], ils peuvent facilement s'adapter au contexte 5G. De plus, les solutions basées sur NT atténuent le besoin de coopération de tous les nœuds dans le processus de surveillance et réduisent par conséquent la surcharge de surveillance. Une autre solution très prometteuse pour fournir un système de surveillance automatisé est la combinaison de l'IA et de la NT. Dans [69] met en évidence que les réseaux de neurones peuvent améliorer les solutions NT en assouplissant leurs hypothèses fortes telles que : les mesures de chemin corrélées, le nombre limité de nœuds/liens défectueux dans un réseau, ou un support de protocole réseau spécial.

Suite aux motivations de fournir une génération de réseau entièrement automatisée. L'objectif principal de cette thèse est d'aborder les défis algorithmiques associés au placement des tranches de réseau ainsi qu'à leur surveillance de manière autonome. En d'autres termes, en utilisant des méthodes basées sur l'IA. Ainsi, pour chaque défi, nos contributions peuvent se résumer comme suit :

— Concernant le placement des tranches de réseau, nous proposons de tirer parti des méthodes d'apprentissage par renforcement profond pour surmonter les limitations des méthodes existantes et nous proposons les deux solutions suivantes :
  — Dans le premier, l'objectif est d'apprendre automatiquement à résoudre le problème de découpage de réseau. Au lieu de mettre une contrainte sur la topologie de l'infrastructure physique ou d'extraire des fonctionnalités manuellement, nous formulons la tâche comme un problème de renforcement, et nous utilisons une architecture neuronale basée sur la convolution Graph pour apprendre à trouver une solution optimale.
  — Ensuite, au lieu de former un agent DRL à partir de zéro pour trouver la solution optimale, un processus qui peut entraîner une formation dangereuse,

nous l'entraînons pour réduire l'écart d'optimalité des heuristiques existantes. La motivation derrière cette contribution est d'assurer la sécurité lors de la formation de l'agent DRL

— En matière de surveillance des tranches, nous proposons de tirer parti des techniques de machine learning et de tomographie en réseau afin de proposer des solutions de surveillance flexibles, évolutives et autonomes.

  — Tout d'abord, nous aborderons le problème d'inférer les métriques QoS d'une tranche : Nous modélisons la tâche comme un problème de régression multi-sorties, que nous résolvons en utilisant des réseaux de neurones. Nous proposons de s'entraîner sur des données synthétiques pour augmenter la diversité des données d'entraînement et éviter le problème de surapprentissage. De plus, pour faire face aux changements pouvant survenir soit sur les tranches que nous surveillons, soit sur la topologie sur laquelle elles sont placées, nous utilisons des techniques d'apprentissage par transfert.

  — Notre deuxième contribution dans le cadre du monitoring concerne le placement optimal des moniteurs : Nous considérons un cas particulier où seules les sondes de cycles sont autorisées. Les schémas de cycle de sondage ont un avantage significatif par rapport aux chemins réguliers, puisque la sonde source est en fait la destination, ce qui réduit les problèmes de synchronisation. Nous formulons le problème comme une variante du problème de couverture d'ensemble minimum. En raison de sa complexité, nous introduisons une solution autonome basée sur des réseaux de neurones graphiques (GNN) et des algorithmes génétiques pour trouver un compromis entre la qualité du placement des moniteurs et le coût pour y parvenir

Nous résumons ici les grandes lignes de cette thèse :

— Dans le chapitre 2, nous présentons les principales technologies habilitantes des réseaux 5G et post 5G. Ensuite, nous introduisons les concepts de machine et de deep learning liés à notre travail, et nous mettons en évidence le rôle attendu des algorithmes d'IA dans ces réseaux. Plus tard, nous passons en revue les travaux existants sur le problème de placement des tranches de réseau et ses limites. Enfin, nous expliquons notre intérêt pour l'utilisation de la tomographie de réseau dans la surveillance de tranches de réseau, et nous passons en revue les travaux existants dans le domaine de la surveillance basée sur NT.

— Dans le chapitre 3, nous présentons les deux premières contributions de cette thèse. Nous commençons par une définition formelle du problème du placement des tranches. Ensuite, nous présentons la première contribution. Tout d'abord, une formulation du problème en tant que problème de décision de Markov est présentée. Ensuite, nous présentons les détails de notre architecture neuronale, que nous avons entraînée par apprentissage par renforcement. Ensuite, nous présentons le montage de la simulation et l'analyse des résultats obtenus. Dans la deuxième contribution, nous introduisons les motivations derrière l'apprentissage pour améliorer les performances des heuristiques, puis nous définissons la tâche comme un problème RL. Ensuite, nous effectuons des simulations pour évaluer les performances de notre proposition à l'aide de deux heuristiques, et nous analysons les résultats obtenus.

— Dans le chapitre 4, nous nous concentrons sur la surveillance des tranches. Nous présenterons nos deux contributions citées ci-dessus. Pour la troisième contribution, nous commencerons par définir le problème d'inférence puis les motivations qui nous poussent à le formuler comme un problème de régression. Ensuite, nous soulignons les défis auxquels nous sommes confrontés en le formulant de cette façon, et nous détaillons comment nous les traitons. La configuration de la simulation et les résultats seront présentés plus tard. Enfin, concernant la quatrième et dernière contribution, nous commencerons par définir le problème du placement des moniteurs et montrer sa complexité. Ensuite, nous présentons notre solution : MonGNN et ses principaux composants, à savoir les parties prédiction et identification. Nous terminons ce chapitre par l'évaluation de MonGNN et l'analyse des résultats obtenus.

— Dans le chapitre 5, nous concluons notre travail et présentons les directions à prendre pour étendre et améliorer nos contributions dans les recherches futures.

Chapitre 1

Introduction

The early 70s was the start of the era of mobile wireless networks. Since then and over four decades, four generations of mobile wireless networks have emerged, as presented in Figure 1.1. The first generation(1G) was introduced in the 1980s, and based on analog technologies, it offered only voice transmission. One decade later, and to overcome the limitations of the 1G, such as the scalability issues and limited capacities, the second generation (2G) was introduced. 2G networks offered low bitrates and allowed the support of multiple users by using digital technologies and the Time-Division Multiple Access (TDMA) method. Later in 2001, and in order to offer higher data rates, the third generation was introduced. 3G networks were known by the Code Division Multiple Access (CDMA) technique, and that they allow supporting mobile internet services with higher data rates than 2G. One decade later again, the fourth generation(4G) of cellular networks arrived. Known also as Long Term Evolution (LTE) networks. 4G are deployed all over the world. By eliminating circuit switching, 4G replaced it with an IP network based on the packet switching to support wide data rates and various mobile broadband services. Since the implementation of 4G networks and its adoption by industrial firms, the number of mobile and Internet of Things (IoT) devices has drastically increased. This lead also to an increase in the volume of mobile data traffic. According to a recent forecast [47], mobile data traffic is estimated to grow at an annual rate of around 55% in 2020-2030 to reach 607 exabytes (EB) in 2025 and 5, 016 EB in 2030. Hence, a new way of managing the network is needed to deal with this significant increase. The proposed solution was the fifth generation (5G). Unlike all the other generations, 5G is not an evolution of its antecedent. Instead, it is a paradigm shift in mobile wireless networks. The work described in this thesis is about 5G and Beyond 5G networks (B5G).

This chapter is organized as follows : First, we introduce the context of our work as well as its motivations and the challenges we aim to solve. Then, we list our contributions, and finally, we present the structure of the thesis.

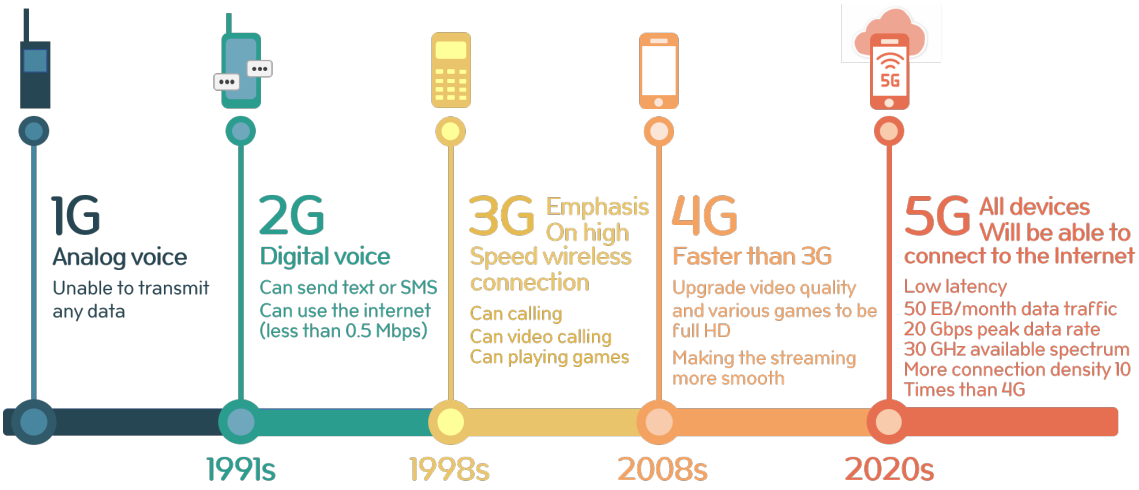


FIGURE 1.1 – timeline of the evolution of the mobile wireless generations [30]

1.1 Context, motivations and challenges

Unlike the other mobile wireless generations, where the main objective was to satisfy only the requirements of human-to-human communications, 5G networks are expected to support all types of communications (e.g., machine to machine, human to machine). Furthermore, 5G systems expect to fulfill the requirements of all these services using a single physical infrastructure. Hence, 5G networks must have the required features to achieve this vision. Software-Defined Networking (SDN) and NFV (Network function virtualization) are one of these mandatory features. Indeed, these technologies are behind the rise of the concept of *Network Slicing*(NS), which was introduced as one of the critical enablers towards the realization of this shift of vision. To accommodate the co-existence of these heterogeneous services, NS aims to create several virtual networks (slices), each dedicated to meet the Key Performance Indicators (KPIs) of a particular service. Hence, thanks to NS, operators can reduce their Capitale EXpenditure (CAPEX) since they will be sharing a single physical infrastructure to provide multiple services. Moreover, they can also reduce their Operational Expenditure (OPEX) thanks to the softwarization and the virtualization of the network.

Such a revolution brings a lot of advantages but also a plethora of challenges related to the management and the planning of 5G networks. Due to the diversity of the services they provide, 5G also requires a change in the methods to deal with these challenges. The objective is to achieve fully automated management of networks. Therefore, Artificial

Intelligence appears as a natural solution (AI). AI solutions are tailored in general to solve complex challenges, which is the case for network slicing. Indeed, with the deployment of a large number of different slices independent from each other, the complexity of their management will increase. Among these challenges and which will be addressed in this thesis, we can cite the placement of network slices and their monitoring.

The placement of slices is not a novel challenge ; it can be considered as a variant of the virtual network embedding problem, which is considered in the literature as an NP-Hard problem. Hence to cope with the complexity of the task and ensure automation at the same time, an AI-based approximation algorithm is needed to solve it. Another reason behind the choice of AI as a solution is its ability to learn to provide solutions more accurately than human-designed heuristics.

When it comes to monitoring the slices, it is a vital aspect to ensure the correct functioning of the services run by the slices and to meet the service level agreements negotiated between slices' providers and slices' tenants. However, in the 5G context, monitoring the slices become more challenging. Indeed, a slice can be deployed on top of different domains independently administrated. Hence, the global monitoring of the slices becomes much more complex, especially when mutual restrictions between domains exist. This calls then for new monitoring solutions that avoid these issues and rely upon external measurements. These solutions are called Network Tomography (NT). They are well studied in the literature, and thanks to their fit with the SDN and NFV paradigms [94], they can easily be adapted to the 5G context. In addition, NT-based solutions mitigate the need for cooperation of all nodes in the monitoring process and consequently reduce the monitoring overhead. Another solution that holds much promise to provide an automated monitoring system is the combination of AI and NT. In [69], the authors highlight that neural networks can enhance NT solutions by relaxing their strong assumptions such as the correlated path measurements, the bounded number of faulty nodes/links in a network, or a special network protocol support.

1.2 Contributions

Following the motivations to provide fully automated generation of network. The main objective of this thesis is to address the algorithmic challenges associated with the placement of the network slices as well as their monitoring using AI-based methods. Hence, for each challenge, our contributions can be summarized as follows :

— Regarding the network slice placement, we propose to leverage deep reinforcement learning methods to overcome the limitations of existing methods, and we provide the two following solutions :
  — In the first one, the aim is to learn automatically how to solve the network slicing problem. Instead of putting any constraint on the topology of the physical infrastructure or extracting features manually, we formulate the task as a reinforcement learning problem, and we use a Graph convolutional-based neural network architecture to learn how to find an optimal solution.
  — Next, instead of training a DRL agent from scratch to find the optimal solution, a process that may result in unsafe training, we train it to reduce the optimality gap of the existing heuristics. The motivation behind this contribution is to ensure safety during the training of the DRL agent. By safety, we mean avoid choosing actions that may lead to an unfeasible placement and then disturb slices functioning.
— When it comes to the monitoring of the slices, we propose to leverage machine learning and network tomography techniques in order to propose flexible, scalable, and autonomous monitoring solutions.
  — First, we address the problem of inferring the QoS metrics of a slice : We model the task as a multi-output regression problem, which we solve using neural networks. We propose to train on synthetic data to augment the diversity of the training data and avoid the overfitting issue. Moreover, to deal with the changes that may occur either on the slices we monitor or the topology on top of which they are placed, we use transfer learning techniques.
  — Our second contribution in the context of the monitoring concerns the optimal placement of monitors : We consider a special case where only cycles' probes are allowed. The probing cycle schemes have a significant advantage compared to regular paths since the source probe is actually the destination, which reduces the synchronization problems. We formulate the problem as a variant of the minimum set cover problem. Owing to its complexity, we introduce a standalone solution based on Graph Neural Networks (GNNs) and genetic algorithms to find a trade-off between the quality of monitors placement and the cost to achieve it.

The use of GNNs rather than the other types of neural networks is justified because they generalize the qualities of the Convolutional Neural Networks (CNNs) to graph-structured

data. CNNs can learn spatial features only when each node has a limited number of neighbors (which is the case of image data, each pixel has at most four neighbors). However, GNNs can learn representations for graphs with an arbitrary number of neighbors.

1.3 Thesis outline

Here we summarize the outline of this thesis :

— In Chapter 2, We introduce the key enabling technologies of 5G and post 5G networks. Then, we introduce the machine and deep learning concepts related to our work, and we highlight the expected role of AI algorithms in these networks. Later, we overview the existing works addressing the network slices placement problem and their limitations. Finally, we explain our interest in using network tomography in monitoring network slices. Then, we overview the existing works in NT-based monitoring field.

— In Chapter 3, we present the first two contributions of this thesis. We start with a formal definition of the problem of the placement of slices. Then, we present the first contribution. First, a formulation of the problem as a Markov decision problem is presented. Next, we present the details of our neural network architecture, which we trained in a reinforcement learning manner. Later, we present the simulation setup and the analysis of obtained results. In the second contribution, we introduce the motivations behind learning to improve the performance of heuristics, and then we define the task as an RL problem. Following this, we conduct simulations to assess the performance of our proposal using two heuristics, and we analyze the obtained results.

— In Chapter 4, we shift our focus to the monitoring of the slices. We will present our two contributions cited above. For the first contribution of this chapter, we will start by defining the inference problem and then the motivations that push us to formulate it as a regression problem. Then, we highlight the challenges we face when formulating it this way, and we detail how we deal with them. The simulation setup and results will be presented later. Finally, regarding the second contribution of this chapter, we will start by defining the problem of monitor placement and showing its complexity. Then we introduce our solution : MonGNN and its main components, namely, the prediction and the identification parts. We finish this chapter with the evaluation of MonGNN and the analysis of the obtained results.

— In Chapter 5, we conclude our work and present the directions to take to extend and improve our contributions in future research.

Chapitre 2

Background and Related Work

As mentioned in the previous chapter, our work in this thesis focuses on two main challenges : The placement of 5G network slices and monitoring their state. Before we dive into the specificities of our contributions, we first need to provide a detailed background of 5G and beyond 5G (B5G) networks and their key enablers. Next, we continue with defining the key concepts of ML and deep learning and discussing their role in 5G and B5G networks. Last, the works that are most relevant to the contributions of this thesis are introduced, explaining the challenges that they try to address as well as their shortcomings, which drive the need for the novel solutions presented in the following chapters.

2.1 5G and post 5G key technologies

2.1.1 5G and post 5G : expectations and opportunities

5G and beyond networks are intended to offer a versatile, scalable, agile, and programmable network platform for deploying and managing various services with different requirements under stringent performance limits. Indeed, 5G networks will support a variety of services that can be classified into three main categories : enhanced mobile broadband (eMBB), ultra-reliable low latency communication (URLLC), and massive machine-type communication (mMTC), as shown in Figure 2.1.

For the eMBB, 5G is expected to offer high data rates to the human-centric scenarios to access multimedia contents such as Augmented Reality (AR), 4K, and 8K screens. Regarding the URLLC cases, 5G promises to reduce latency and increase throughput for the services with strict requirements in terms of QoS, such as self-driving cars and remote medical surgery. Finally, for mMTC, 5G will ensure high connection density for cases with a very large number of connected Internet of Things (IoT) devices which send low volume data. As an example for this category, we can cite smart cities and intelligent

transportation.

Overall, for all these services, 5G networks will ensure :

— A download speed of up to 20 gigabits (Gb) per second, 200 times faster than with current 4G technology.

— Latency less than 1 millisecond

— A high connection density with a target of one million devices per km^2

Alongside enhancing the performance of these various services, and despite their competing requirements, 5G attempts to provide them utilizing a single physical infrastructure. Accordingly, 5G will open to network operators new business opportunities. Indeed, network operators will be increasingly involved in the value chain of the services rather than just maintaining their connectivity and being only a carrier grade network.

Network operators should redesign their architecture to achieve these expectations and to enable eMBB, URLCC, and mMTC services on top of the same infrastructure. The "one-size-fits-all" network paradigm that characterized previous mobile network generations does not fit the 5G vision anymore. Hence, to satisfy the service level requirement (SLR) of each type of these services, 5G architecture should afford more flexibility and agility. Network slicing has emerged as a critical enabler to answer the requirements mentioned above of 5G networks. Besides the multi-tenancy support on top of the physical infrastructure, network slicing seeks to ensure service isolation and customization. Network slicing relies, indeed, on two main technologies : software-defined networking (SDN) and network function virtualization (NFV).

The rest of this section will detail the concept of network slicing followed by its key enablers : SDN and NFV.

2.1.2 Network slicing

Network slicing [1] is one of the essential building blocks of 5G and B5G networks by offering dedicated services to verticals and opening new business opportunities to infrastructure providers since they will be more involved by leasing their physical to verticals and third parties to customize their services. Moreover, It will ease the management of the networks despite their heterogeneity [6].

Network slicing is not a novel concept. Its roots back to the late 80s where overlay networks were established. Overlay networks are made up of nodes connected by virtual links that build a logical network on top of a network composed of physical infrastructure.

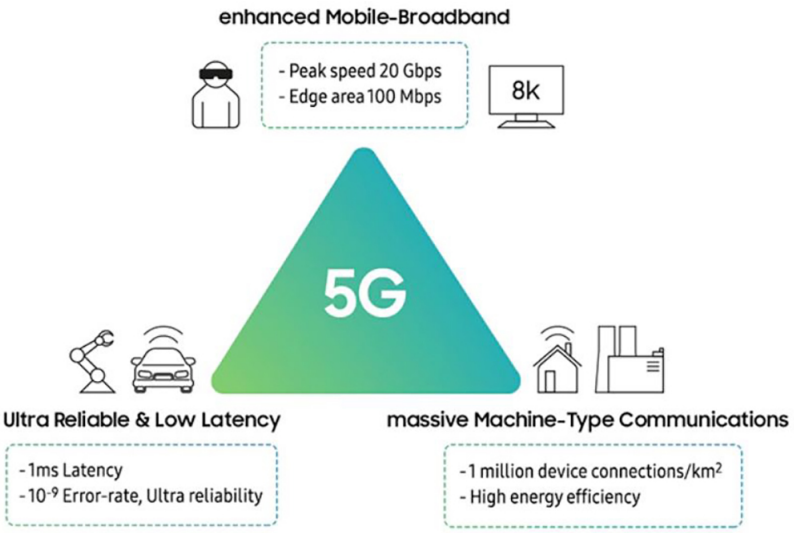


FIGURE 2.1 – 5G Usage Scenarios

Even if they enable the combination of heterogeneous resources from different domains, overlay networks lack programmability and automation, which are considered as one of the requirements of 5G networks.

In the context of 5G networks, *Network slicing* was introduced by NGMN [80] as a paradigm that suggests partitioning the physical infrastructure into several virtual networks called slices, as presented in Figure 2.2. Later, 3GPP defined *network slicing* as a technology that enables the creation of customized networks to provide optimal solutions for diverse market scenarios with diverse demand requirements. Thereby, network slicing can improve network resource utilization, reduce the Capital Expense (CAPEX) and Operational Expense (OPEX) costs of the network infrastructure.

To meet all these requirements and achieve all the expectations we mentioned earlier, network slicing was built on top of the following three main principles :

— **Isolation :** It is one of the fundamental properties that should be satisfied in network slicing. It ensures a secure performance for each tenant as well as inter-slice independence.

— **Elasticity :** Elasticity can be achieved by shifting virtual network functions, scaling up and down allocated resources, and reprogramming the control and data element functionalities. The key problem in implementing elasticity will be the policy for inter-slice negotiation so that the performance of distinct slices is not affected.

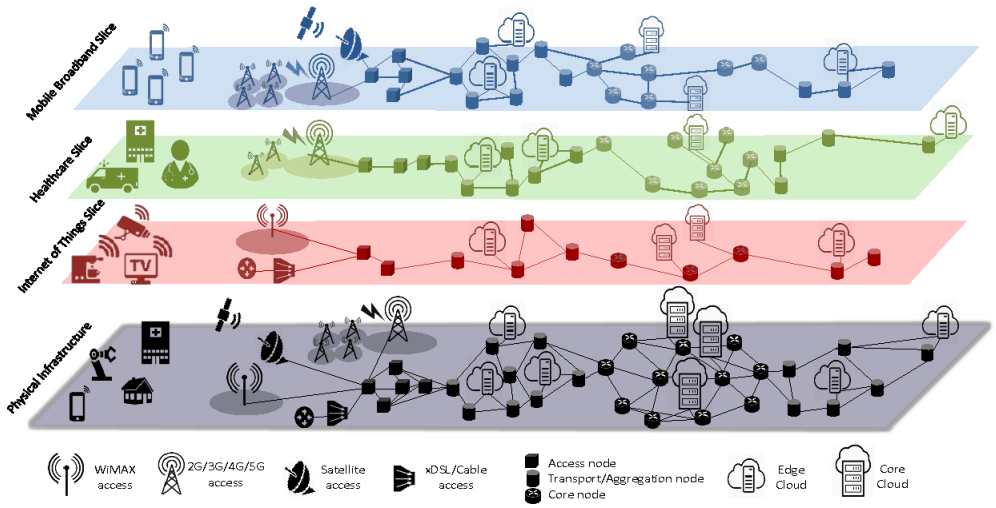


FIGURE 2.2 – 5G Network Slicing concept [81]

— **End-to-End Customization :** Network slicing must guarantee that the resources allocated to a tenant are efficiently utilized in order to assure all service requirements. Furthermore, network slicing is expected to enable this customization all the way from the service provider to the end-user and along with different domains (RAN, transport, and Core)

2.1.3 Software-defined networking

Software-defined networking (SDN) is an emerging networking paradigm that aims to simplify networking and make networks programmable [57]. One of the fundamental principles of SDN is the separation of the control and data planes. As a result, the role of the forwarding devices (routers and switches) will be limited to apply rules dictated by an external centralized entity called the SDN controller. With the SDN controller, these forwarding rules become flow-based instead of being destination-based. In addition, by running software applications at the top of the SDN controller, deploying new protocols and management of the network is made easier.

As illustrated in Figure 2.3, in SDN, the network is divided into three main layers, namely : application, control, and data planes :

— Application plane : This layer includes a set of programs that exchange their needs with the SDN controller through a northbound interface. They describe all the behavior to be applied in the network.

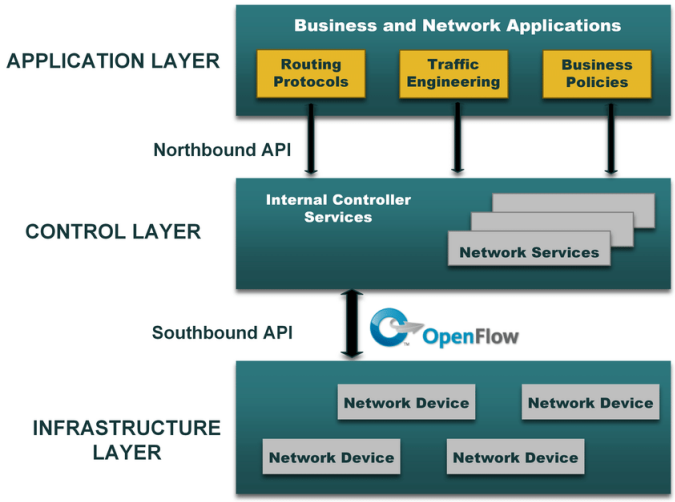


FIGURE 2.3 – SDN architecture

— Control plane : This layer is the place of the SDN controller, a software entity that utilizes its ability to get a global view of the network to push the forwarding rules to the data plane devices. These rules are a translation of the resource requests that the controller gets from the applications deployed in the application layer.

— Data plane : This layer is composed of interconnected forwarding devices. In the SDN era, these devices are programmable and receive forwarding rules from the controller via a southbound interface.

The control layer plays a key role in this architecture. It receives the logic to deploy in the network from the application layer via the northbound interface. It then translates to a set of rules that it transfers to the forwarding devices via the southbound interface. Open Flow (OF) [73] is one of the most used protocols to ensure this latter communication. It was standardized by the Open Networking Foundation (ONF). Other protocols were also adopted for this interface (e.g., NETCONF, SNMP). On the other hand, for the northbound communication, there are no standardized protocols yet, but the communication usually relies on Representational State Transfer (REST) APIs.

SDN is considered as one of the promoters of the 5G slicing vision because it eases the programmability and the flexibility needed for the slices. Moreover, SDN simplifies the management and configuration of the slices with the separation of control and data planes.

2.1.4 Network function virtualization

Network Function Virtualization [63] is a concept that leverages the IT virtualization tools to virtualize network functions (Routers, Firewalls, CDN,..) that used to be run on a dedicated hardware. In other words, it removes the dependency between the network function and the hardware on which it is deployed. These virtual functions can be chained to deliver a specific service, and they can be scaled down/up and even relocated according to the evolution of the service needs.

Both core network and radio access network functions can be virtualized using NFV. As a result, operators can lower their OPEX and CAPEX. Operators will be able to reduce their need for dedicated built-in hardware and eliminate wasteful over-provisioning. They will also lower the cost of power and equipment cooling [8].

Thanks to the flexibility and agility it offers, NFV is regarded as a key enabler of the 5G network slicing paradigm. The creation and the management of a slice are much simpler. A network slice can be created by chaining together a collection of virtual network functions (VNFs) that can be deployed on any standard server and in any location. Furthermore, the NFV framework can manage each VNF in the slice based on the requirements of the service it runs.

In order to manage the NFV framework, the European Telecommunications Standards Institute (ETSI) group has proposed the Management and Orchestration architecture (MANO). As depicted in Figure 2.4 : this architecture is composed of three main blocks :

— Virtual network functions (VNF) : As its name tells, they represent the software (virtual) instantiation of network functions.

— NFV Infrastructure : It includes the elements required to run the VNFs. For instance, the physical resources, the containers, and hypervisors.

— NFV MANO : It is in charge of the life-cycle management of networks services and the performance of the NFVI and VNFs. It is composed of three elements : NFV orchestrator, VNF Manager, and the virtualized infrastructure manager.

2.2 Intelligence within 5G and post 5G networks

5G networks promise more than just increasing capacity and service availability. Autonomous management and dynamic orchestration are also important goals for 5G and B5G networks. In fact, ensuring 5G network expectations is accompanied by an increase

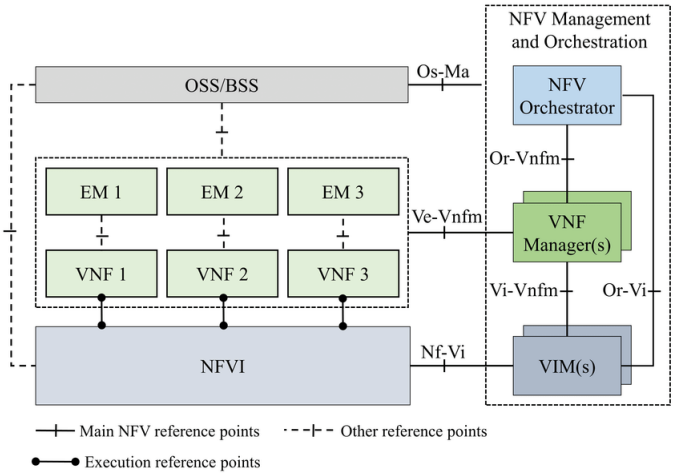


FIGURE 2.4 – NFV MANO architecture

in the complexity of their management. As a result, traditional management methods may be thought to be obsolete. Owing to its importance, autonomous management of 5G networks has piqued the interest of both academia and industry. It leads to the "zero-touch networks" concept. The primary objective is to reduce human intervention and ideally fully automating the network slices life-cycle management. Artificial intelligence (AI) and Machine Learning (ML) are broadly recognized as fundamental features in this vision [34].

In this section, we will provide an overview of ML, with a focus on its sub-field Deep Learning (DL). We will start by defining what ML and DL are. Then, we describe different learning paradigms. Next, we give an overview of different neural network architectures. Last, we end by giving examples of deep learning applications in 5G and B5G networks. In this section, we focus only on the deep learning concepts that will be important to assimilate the rest of this thesis.

2.2.1 Machine learning : definition and paradigms

Machine learning is a sub-field of artificial intelligence that impacted several domains. ML is defined as "*The field of study that allows computers to learn without being explicitly programmed*" [78]. To define properly what machine learning is, we need to answer the following question :" what is meant by learning." The answer was given by Tom Mitchell, who defined learning as follows *"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P improves with experience E."*

Based on the degree of supervision required during training, we distinguish between four main paradigms of learning : supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

— **Supervised learning :** It is one of the most popular types of ML. In supervised learning, the algorithms are trained with labeled data. Each instance in the training data is accompanied by its label or its target. The algorithm is trained then to find the relation between the features of the instances and their corresponding labels. We have two main tasks in this setting : classification and regression. The difference between these tasks relies upon the type of labels to predict. Regression tasks are the ones where the label to predict is a continuous variable, while classification tasks concern the case where the labels represent categories. When the number of categories equals two, we talk about binary classification. Otherwise, it is known as multi-class classification.

— **Unsupervised learning :** In this category, there is no supervision. The training is done on an unlabeled dataset, and the aim is to learn hidden patterns and representations of the data. This type of learning includes several tasks : clustering, dimensionality reduction, association rule learning, and data generation. Clustering consists of gathering a set of inputs into distinct groups. On the other hand, dimension reduction is usually used for visualization purposes. The algorithms attempt to learn a representation of the input that is lower in dimension than the original data while retaining as much structure as possible. For data generation, the algorithms are trained to produce new data instances that follow the same statistical distribution as the data of the training. Finally, association rules learning refers to the task which aims to dig in large amounts of data and learn relations between features of instances.

— **Semi-supervised learning :** This type of learning includes algorithms with the ability to learn from partially labeled data. They can be conceived as hybrid learning algorithms that combine supervised and unsupervised learning methods supervised and unsupervised learning methods.

— **Reinforcement learning :** It is a different concept. The learning mimics how humans learn in their lives. The learning is done by an agent (an algorithm) which interacts with its environment. The agent can observe the state of the environment, select actions, and based on the quality of its choice, it gets a reward (or punishment). The objective is to learn an optimal policy to maximize the rewards it gets

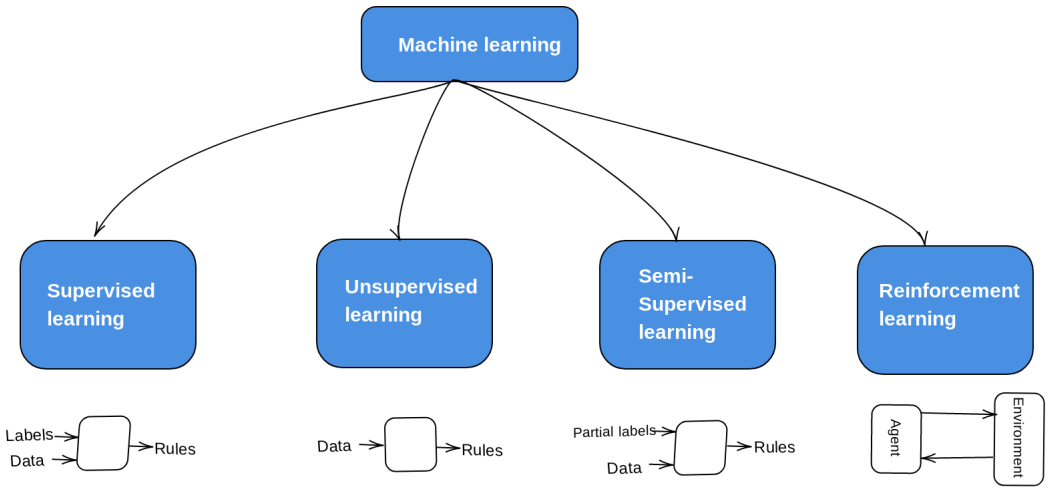


FIGURE 2.5 – Types of machine learning paradigms based on their degree of supervision

over time.

Machine learning approaches can also be categorized based on the algorithms they employ. In the literature, we distinguish between *shallow learning* and *deep learning* algorithms. Shallow learning models, also known as classic machine learning models, are known for their low-level representations. They can not learn complex levels representations of the original data. Besides, shallow ML models execution is usually preceded by a feature engineering step. In other words, to get full advantage from shallow ML, a process of finding the key features that characterize the input must be done first. For example, in an image classification task, we have to manually provide high-level features such as the number of edges, the number of circles, etc.

On the other hand, Deep Learning is defined as a sub-field of ML that employs Artificial neural networks (ANNs) to learn in a fully automated and progressive manner a useful representation from data that will make the final task (e.g., classification or regression) easier. The "deep" part in deep learning refers to the number of layers in ANNs. A layer is defined as a set of *neurons* or *units*. Each neuron is in charge of a simple computation process that transforms an input into an output. As depicted in Figure 2.6, when the input is an n-dimensional vector $X = (x_1, x_2, ..., x_n)$, a neuron will transform it to a new

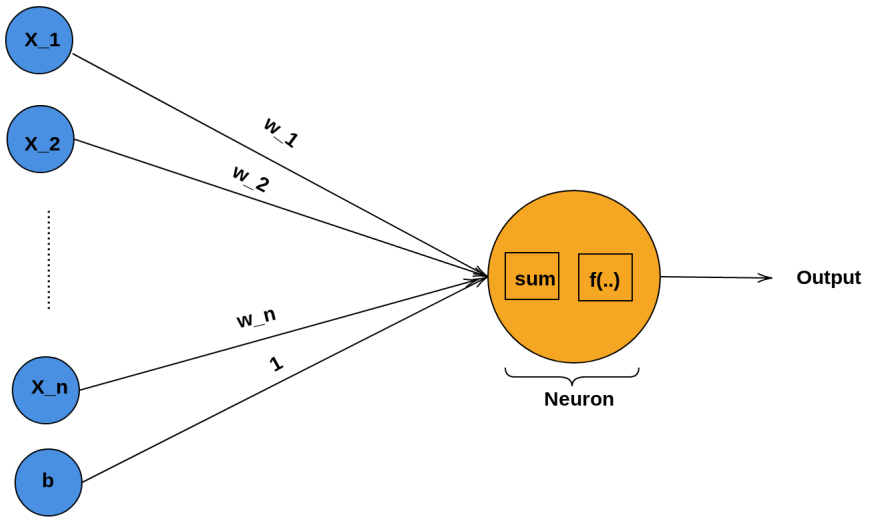


FIGURE 2.6 – Illustration of how a neuron transforms an input vector

representation according to the following equation :

$$output = f(b + \sum_i w_i x_i) \tag{2.1}$$

where w_i represents the weight of the connection between the neuron and the $i-th$ element of the input vector, b is the bias term, and f refers to the activation function. The role of the activation function is essential. It introduces non-linearity and lets ANNs learn complex non-linear representations. As a result, ANNs can be seen as a set of connected layers, where each layer is a collection of stacked neurons.

2.2.2 Types of deep neural networks

— **Feed Forward Neural Networks (FFNNs) :** They are the most basic deep neural networks, where information passes only in the *Forward* direction from the input to the outputs layers via a set of hidden layers. Called also Multi-Layers Perceptron (MLPs) [86], FFNNs are known for their ability to approximate any functions. They can be used to process tabular data, images, and sequential data (e.g., text or audio). Meanwhile, using FFNNs to process image data has some drawbacks. For example, to handle an image classification task, the first step is to

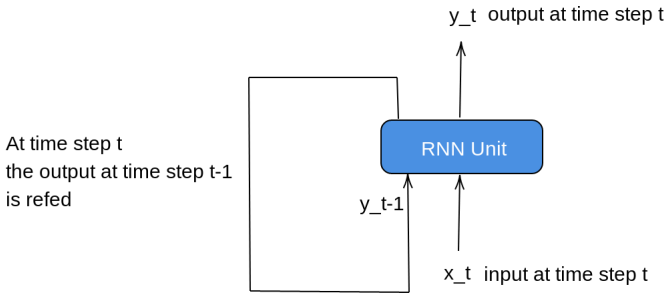


FIGURE 2.7 – Recurrent neural network unit

convert the 2d dimensional image into 1-d ; hence the number of the input layer and the number of trainable parameters will increase. Moreover, when processing images using FFNNs, we lose the spatial information of the image. In other words, we lose the features related to the arrangement of the pixels in an image. Similarly, for sequential data, FFNNs fail in capturing the sequential features of the inputs. Recurrent neural networks and convolutional neural networks were proposed to deal with these limitations.

— **Recurrent Neural Networks (RNNs)** : A category of neural networks specialized in processing sequential data. As illustrated in Figure 2.7, the output at each time step depends on the current step and also the output of previous steps. To take the sequential information into account efficiently, RNNs share the trainable parameters across timesteps, and as a result, the number of trainable parameters will be lesser compared to the FFNNs. Long Short Term Memory[45] (LSTMs) and Gated Recurrent Units (GRU) networks [20] are common variants of RNNs that provide better memory for the long term.

— **Convolutional Neural Networks (CNNs)** : They are the key neural networks behind the success of deep learning in computer vision tasks. CNN's are specialized in dealing with grid-shaped data such as images [14]. The extraction of spatial features is done through a convolution between the input and a filter (kernel). Like RNNs, CNNs use parameter sharing techniques to reduce the complexity of the layers. Moreover, what makes CNNs a good choice for image data is their transition shift-invariance characteristics. For example, an image and its rotation will be recognized as the same and got the same representation.

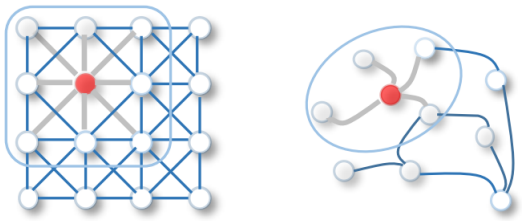


FIGURE 2.8 – From 2D convolutions to graph convolution [120]

— **Graph Neural Networks (GNNs)** : Text, image, and video data are known in the AI literature as euclidean data, which we can model in the euclidean space. An example for non-euclidean data is Graphs, by which we can model several aspects such as social networks, molecules, and communications networks. Due to the non-euclidean nature of graphs, the ANNs we described above can not learn transition invariant representations. Consequently, GNNs were proposed to extend neural networks application in the non-euclidean domains. GNNs can be defined as the category of neural networks dedicated to graph-structured data. Graph convolutional neural networks [53] and Graph ATtention networks (GAT) [116] are the well-known implementations of GNNs. They extend the convolution operation to the graph domain. As illustrated in Figure 2.8 and presented in [120], an image can be considered as a special case of graphs where pixels are connected by adjacent pixels. Similar to 2D convolution, one may perform graph convolutions by taking the weighted average of a node's neighborhood information.

2.2.3 Deep learning in communication networks

As mentioned before, ML is considered as a pillar to achieve the zero-touch network vision. Accordingly, several 5G PPP projects applied deep learning techniques to tackle several problems in 5G and B5G networks, such as Network planning, network diagnostic and network optimization and control.

Network planning

Network planning refers to the set of network designing and dimensioning processes that precede the network deployment. To deal with the complexity of 5G and B5G networks, several AI-based solutions were proposed. For instance, the authors [21] focused on the network element placement problem. The objective was to enhance classical methods

to improve the identification of the optimal placement of base stations while considering their different locations and their various parameters such as coverage, user equipments' density, and overall hardware costs. Following the same spirit of enhancing the pre-network deployment steps using AI, the 5G-COMPLETE project [88] aims to use ML-based algorithms for dimensioning C-RAN clusters. The project focused in particular on finding intelligent solutions to the problem of optimal baseband unit (BBU) functions allocation. They used Neural networks models (LSTM, FFNNS) to estimate the BBU processing requirements in order to achieve an automatic dimensioning of the network.

Network diagnostic

Troubleshooting is of paramount importance for network operators and also for network slices tenants in 5G and beyond 5G networks. Several 5GPPP projects have leveraged DL techniques in order to inspect the state of the network autonomously. In this management aspect, some works had put the accent on using DL in forecasting mobile traffic networks while others focused on traffic classification and others focused on SLA predictions in a multi-tenant environment. For instance, the 5GZORRO project [5] proposed solutions to synthesize high-resolution mobile traffic. The aim was to reduce the complexity of collecting mobile data ; hence a solution that transforms coarse-grained data to a fine grained-data was proposed. The ML technique that was employed is a Generative Adversarial neural network. Within the same project, a solution to forecast mobile traffic efficiently was introduced [129]. They introduced a spatio-temporal neural (STN) architecture that combines LSTMs and CNNs. In this work, the authors used an encoder-decoder architecture that was trained using an available 60-day long traffic measurement collected in the city of Milan and the Trentino region provided by Telecom Italia [12]. The use of deep neural networks yielded a lower error estimation compared to specialized methods in time series forecasting such as ARIMA [31]. On the level of SLA prediction, the 5G-CLARITY[87] project proposed a solution that employs echo state networks to automatically forecast the trend of the network traffic and predict the possible SLA violation rate.

Network optimization and control

Regarding the network optimization and control, the focus was on different network domains, namely : RAN, transport, and end-to-end slices. For instance, the projects 5G-CLARITY, 5G-HEART[2] addressed in the RAN domain several problems such as the

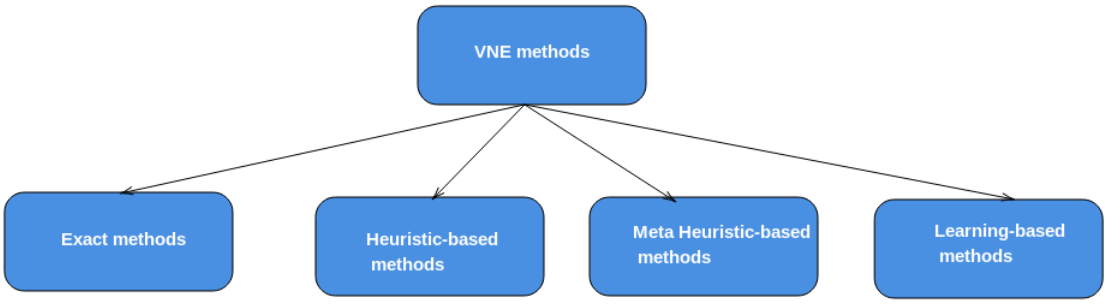


FIGURE 2.9 – Classification of existing methods for the VNEP

slicing in a multi-tenant network, radio resource provisioning, and traffic steering. On the transport domains, several proposals from 5GROWTH [90]and 5G-COMPLETE [88] projects leverage AI algorithms to deal with traffic management, dynamic load balancing, and flow scheduling in transport networks. Finally, in regard to the E2E slicing, 5GVINNI [4], MONB5G[91],5G-TOURS[3], and 5GENESIS [89] used AI to tackle several use cases related to automated service assurance slice admission control and slice optimization. Readers interested in the details may refer to [7]

2.3 Network slice placement

The Network Slice Placement problem can be framed as a variant of Virtual Network Embedding Problem (VNEP) or VNF Forwarding Graph Embedding (VNF-FGE). Both modelizations were the subject of several works. In this section, we present the works that focus on solving the network slicing problem. Existing works can be classified into four main categories as depicted in Figure 2.9 : Exact methods, Heuristics, metaheuristics, and finally, machine learning-based solutions.

2.3.1 Exact methods

In this category of solutions, the VNEP is typically expressed using mathematical optimization models. Indeed, different types of mathematical programming were used based on the characteristics studied. Some works used the Integer Linear Programming (ILP) framework to model it, while others modeled it as a Mixed Integer Linear Program (MILP), and finally, others leveraged the Integer Non-Linear Program framework (INLP)

to model it.

M. C. Luizelli et al. [66] formulated the optimization problem as an Integer Linear Program (ILP). In this work, the objective was to minimize the number of virtual network functions mapped to the physical network. The authors assumed that each physical link has a particular delay, which is not completely realistic because of the non-linearity of the relationship between latency and the traffic traversing a link. Similarly, A. Tomassilli et al. [113] formulated the problem as an ILP. The objective was to minimize the total deployment cost ; the authors introduced the problem as a variant of the set cover problem, which results in a logarithmic approximation factor of the optimal. N. Shahriar et al. [105] also modeled the problems as an ILP, but they focus jointly on the full resource utility and request survivability. MILP setup was also considered in several works. I. Jang et al. [49] formulated the placement of service function chain as a multi-objective MILP, and then they transformed it into a single-objective MILP which remains NP-Hard. Finally, the authors propose a polynomial-time solution based on a linear approximation and rounding. Authors in [22] formulated the problem as a MILP, which was relaxed using deterministic and randomized rounding techniques. In this work, the authors added some location constraints to limit the maximum distance between any two embedded virtual nodes. K. Hansel et al.[55] formulate the SFC as an INLP. The objective is to minimize the sum of dissatisfaction ratios between the actual latency and latency requirements of each chain.

Even if this category of solutions is efficient, it is restricted to the modeling phase because of the high complexity of the problem. Being NP-Hard, the optimal resolution only works for very small network instances. Hence in order to solve the large-scale instance of the network slices' placement, a lot of heuristics were proposed in the literature.

2.3.2 Heuristic-based solution

The majority of works found in the literature fall in this category. S. Khebbache et al. [51] proposed an approach based on the construction of an extended multi-stage graph representing servers available to host the required VNFs. The chains corresponding to the VNF forwarding graphs are placed using a maximum flow between the vertices of the different stages of the multi-stage graph representation. Similarly, in [111], authors proposed a graph-based heuristic that leverages graph centrality and a multi-stage graph to select the minimum set of PoPs for the allocation of the SFCs' requests. To prevent exploring all solutions, the authors [110] leverage the nearest search approach and proposed an effective

solution consisting in placing the VNFs iteratively. Likewise, M. C. Luizelli et al. [66]] propose to guide, in an iterative process and using a binary search to explore only feasible placements. Furthermore, J. Pei, [83] introduced SFC-MAP, an algorithm that solves the problem in 2 steps : it places first the VNFs, and then connects them by running the shortest path in a multi-graph resulted from the transformation of the graph representing the physical infrastructure. The authors in [75] extended the eigen-decomposition algorithm to deal with the placement of VNF-FGs. Besides, X.cheng et al. [19], had put the accent on topology-aware node ranking methods, which sort substrate nodes and virtual nodes using the rules inspired by Google's Pagerank algorithm, and then use a "big-to-big, small-to-small" matching strategy to embed virtual nodes onto substrate nodes that have a similar ranking position. Authors in [36] introduced the global resource capacity (GRC) metric, a new cost metric to quantify the potential of each substrate node to host a VNF. Based on GRC, they proposed GRC-VNE to perform a greedy-based load-balancing embedding strategy. C. K. Dehury [24] also addressed the VNE problem but by proposing (DYVINE), a fitness-based dynamic virtual network embedding algorithm, and the aim was to maximize the resource utilization while considering that the requests are dynamically changing. Even game theory-based heuristics were employed to address this task of services placement. In [108], authors formulated the VNE problem as two interleaved coordination games. The first game concerns the placement of VNFs and the second one was about the placement of the virtual links (VLs). Each substrate node is considered as a player that shares the same utility function and tries to achieve a Nash Equilibrium for optimal embedding solutions.

Heuristics are a good alternative to the exact methods since they guarantee a fast computation of a feasible solution. However, this rapid resolution is usually accompanied by a high probability of finding only a sub-optimal solution. This is generally due to the fact that heuristics does not explore the solution space efficiently.

2.3.3 Meta heuristics

As the VNE problem is a combinatorial optimization problem with a large search space, there are several VNE algorithms that use metaheuristic methods to deal with the limitations of heuristics and exact methods. In [50], the authors proposed a meta-heuristic based on the Non-dominated Sorting Genetic Algorithm II (NSGA-II) to address the VNEP. (NSGA-II) is known to be effective when dealing with multi-objective optimization problems. Likewise, the authors in [15] used NSGA-II, which was modified in a greedy

manner by assigning a high priority to the nodes with high residual capacities. For the link mapping, shortest paths are prioritized. Furthermore, another evolutionary algorithm was proposed to deal with the VNEP. In [127], introduced PAME, which used an evolutionary membrane computing mechanism to perform embeddings of VNFs in parallel.

Evolutionary methods are not the only meta-heuristics that addressed the VNEP ; Zhang et al. [130] used particle swarm optimization as a stochastic global optimizer, in which each particle (i.e., possible embedding solution) iteratively improved following an embedding cost function, and a global evolution procedure was performed to obtain a complete solution. Moreover, in order to improve the exploration of the solutions' space, M. C. Luizelli [65] proposed a fix-and-optimize-based algorithm that relies on the Variable Neighborhood Search (VNS) meta-heuristic.

Meta heuristics can effectively solve the NS placement problem. Besides, and unlike heuristics, Meta-heuristics are more flexible and allow adding new constraints or new objectives without reconsidering new solutions. Although these solutions are effective, they can present some problems of convergence and hence slowness. Moreover, with meta-heuristic, each time a solution is needed, an exploration process must start from scratch. Like the methods cited above, meta-heuristics did not accumulate knowledge when solving instances of the network slicing problem.

2.3.4 Machine learning-based solutions

To remedy the limitations of the solutions discussed above, other works used Machine learning to solve the problem of slice placement, especially reinforcement learning and deep reinforcement learning. Authors in [42] modeled the problem within the Markov Decision Process (MDP) framework, and then actions were computed using Monte Carlo Tree Search (MCTS) algorithm. The limitation of MCTS is that each time a decision must be taken, a whole search procedure must be run, which badly affects the scalability of the solution. In [77], authors proposed a decentralized tabular Q-learning solution to the VNE problem. In this work, the authors proposed a discrete state representation,a simplification making their approach unrealistic. Likewise, in [60], the authors used Q-learning to solve the VNEP. They designed a reward function related to the effect of virtual link embedding, which is used to update the Q-matrix through an unsupervised learning process. Nonetheless, the feature extraction in this approach was done manually. To automate the feature extraction and solve the VNEP within the RL framework, authors in [121] used DRL to solve the network slice placement problem automatically. They

adopted a policy gradient approach to train the feed-forward neural network they employ. This work automates the feature extraction, but it does not take into consideration the topological information of the physical network or the slice to be placed. Later, several works attempt to put this into consideration. The authors in [124] used a 1-d Convolutional neural network to extract spatial features from the substrate network. However, with this approach, the relation between substrate nodes is not modeled. More recently, M. Dolati et al. [26] proposed DeepVine, an automated solution to solve the VNE problem. In this work, the authors encoded both the physical network and the slice to be placed as 2D images and then used CNNs to automate the feature extraction. This approach allows using the spatial features of the problem in an automated way, but it puts an unrealistic assumption on the shape of the networks (grid). Accordingly, a solution that combines deep reinforcement learning and graph neural networks since they are specialized in graph data is needed to overcome these limitations without putting unrealistic assumptions on the topology of the physical network. This constitutes the main focus of Chapter 3.

2.4 Network slice monitoring

Regardless of the type of communication network, monitoring the state of the services accommodated and verifying the operation of the elements of the network has been extensively studied in both academia and industry. For instance, conventional-IP networks usually rely on traditional network monitoring tools such as the Simple Network Management Protocol (SNMP) or the traffic sampling-based technologies like Netflow [29] and Sflow [84]. Despite their efficiency, these tools must be revisited in the 5G and B5G era. Indeed, with the high heterogeneity of services and equipment in 5G networks, flexible and scalable monitoring solutions are required. Traditional IP-based monitoring methods are known for their associated overhead and their significant resource consumption. Accordingly, new tools that overcome these limitations, and achieve accurate measurements, need to be developed. Following these requirements, Network Tomography (NT) has emerged as a lightweight solution for efficient monitoring. NT refers to the set of techniques used to infer internal characteristics of a communication network based on external end-to-end measurements of traffic exchanged between nodes able to launch monitoring probes (i.e., monitors)

Network tomography (NT) was introduced by Vardi [114] in 1996. Since then, NT has been extensively studied [16],[67],[72],[33]. Based on the metrics to monitor and the

external measurements to collect, NT can be classified into three main categories :

— Link-level estimation solutions : The objective is to estimate the inaccessible link metrics within the network based on external measurements. Usually, these metrics are the loss rate, delay, and jitter.

— Origin-destination traffic matrix estimation : The objective is to estimate the volume of traffic between all pairs of nodes in the network. The input, in this case, is the count of flows packets which can be collected either using SNMP or by using the flows statistics collected by the Openflow protocol.

— Topology inference : In this category, the goal is to infer the hidden network topology based on path-level measurements

Another classification of NT solutions can be considered based on the way of collecting the end-to-end measurements. We distinguish between active and passive collection. In the former, the process is done by launching probes between monitors in order to measure the end-to-end metric. While the latter focuses on analyzing existing traffic networks.

Most of the existing works related to NT fall into two main categories : statistical network tomography and algebraic network tomography.

2.4.1 Statistical network tomography

The stochastic nature of the metrics to monitor motivates the use of statistical methods. In general, inferring unknown metrics from a collection of external measurements is formulated as follows : At each time t, the relation between the unknown internal metrics denoted by X_t and the external measurement Y_t is defined using the following equation : $T(X_t) = Y_t$. Hence the objective is to solve the inverse problem and find T^{-1}. Existing statistical methods considered X_t as a random variable with some probabilistic distribution f, and the goal is to estimate this distribution from the measurements Y_t. When the metrics subject of monitoring are additive (e.g., the delay or the logarithm of the loss rate), the relation between X_t and Y_t becomes linear and defined as follows :

$$AX_t = Y_t + \epsilon \tag{2.2}$$

where A is a Boolean matrix identifying the followed paths to collect measurements, and ϵ is a noise vector. Most works consider a simple model and ignore the noise part. Then, uniquely finding X_t requires that A is a full rank matrix. In other words, finding

a unique solution for the linear system of equations presented in 2.2 is possible if and only if the number of independent equations is equal to the number of unknown metrics. Unfortunately, in real cases, this is not satisfied, A is generally non-invertible. To deal with this issue and approximate the unknown metrics, several solutions were proposed. In [59], the authors proposed using the Maximum Likelihood Estimation (MLE) using the Expectation-Maximization (EM) algorithm.

2.4.2 Algebraic network tomography

The aim of statistical approaches is to provide an approximation of the unknown metrics when solving the inverse problem discussed above. However, this category of solutions does not focus on the efficient placement of monitors and the selection of measurements paths. It is the Algebraic NT that sheds light on this level. Algebraic NT proposes solutions and necessary conditions which should be met in order to determine the unknown network metrics uniquely. This task is also known in the literature as "identifiability". Network metrics are identifiable if and only if they can be determined uniquely by the end-to-end measurements. When it comes to additive metrics, identifiability is tightly connected to the degree of independence of measurement paths which should be equal to the number of metrics to infer. Thus, the monitor placement and the chosen measurements paths should satisfy these conditions.

Several works focused on the optimal monitor placement problem to ensure the identifiability of links in a network and under different probing schemes. Authors in [43] provided the necessary and sufficient conditions to identify an additive metric while using controllable cycle-free probing paths. In other words, the authors considered the case where it is possible to control the path traffic exchanged between monitors should follow. This assumption is feasible, for example, in SDN-enabled networks. Moreover, the cycle-free assumption refers to the case of considering regular-paths probing where the source and destination of a probe must be different. Under these conditions, authors demonstrated that it is generally impossible to identify all the links' metrics by using two monitors ; Then, they highlight that under some condition of connectivity, all links not incident to any monitor can be uniquely identified using only two monitors. Finally, and based on these conditions and graph decomposition techniques, the authors proposed an algorithm to find the minimum number of monitors and their placement to identify all links in a network. Using the conditions found in this work, the authors in [67] addressed the path selection problem. Given the monitor placement obtained by the algorithm proposed in

[70], the objective was to find the minimum number of measurement paths to use to infer the link metrics.

Instead of identifying all the links of a network, the authors in [33] proposed a solution to the partial identifiability problem. The focus was only on a subset of links, and similarly, an algorithm for the monitors' placement was provided. Later, in [38], the authors studied the identifiability in the dynamic scenarios. The goal is to find an optimal and robust monitor placement that can maintain the identifiability under network changes. The authors proposed a set of algorithms in order to find optimal placement for a given network and its predicted changes. Then methods to handle unpredicted changes were also proposed. All algorithms were based on graph decomposition into connected components.

The NT solutions we presented so far usually assume that the probing paths can not be cycles. However, with the network programmability offered by SDN and the last advents of routing mechanisms such as source routing, the use of cycles becomes possible and full of promises. For instance, in [38], the authors showed that three-edge connectivity is a necessary and sufficient condition to identify link metrics using one monitoring station while employing monitoring cycles. Then, they proposed a linear polynomial algorithm to find a set of linearly independent probing cycles.

Whether they are statistical or algebraic, most of the existing NT solutions aim to estimate the metrics of links or nodes from path measurements. However, in network slicing, a slice will be represented as a chain of VNFs, and the aim is to infer the end-to-end performance of slices. Hence, it will be more interesting to infer QoS metrics of paths instead of focusing only on links. Recently, the authors in [123] were the first to address the use of NT to monitor a set of paths using another measurement path. They showed that the problem is more challenging than monitoring all links of a network, and they proposed a solution based on graph decomposition to find the optimal placement of monitors. However, they assume a cycle-free context. Therefore in chapter 4, we focus on providing a solution to monitor network slices using methods that combine both network tomography and machine/deep learning. In this chapter, we address the problem of inferring slices metrics using neural networks to avoid the high complexity of the EM-based approaches. Moreover, we address the problem of optimal monitor placement to identify slices and propose an automatic method that relies on Graph neural networks.

2.5 Conclusion

In this chapter, we have presented the expectations of 5G and B5G networks. Later, the key enablers of 5G were discussed. In addition, we presented the role of AI in the "zero-touch" vision and how some 5GPPP projects used AI to solve management challenges in 5G systems. Next, we gave an overview of the existing work in solving the placement and monitoring the 5G network slices. The analysis of the existing work shed light on their limitations. Besides, it proves that AI is an excellent candidate to solve these tasks in order to achieve the fully automated management vision.

Following this overview of the context and our contributions' related work, the first two thesis contributions are presented in the next chapter. The aim is to offer an AI-based solution for the challenge of network slice placement.

Chapitre 3

AI-based Network slices placement

3.1 Introduction

The fifth generation of cellular networks (5G) is expected to be the leading infrastructure provider for the next decade. It aims to enable new services and new technologies that were not in the fourth generation. It focuses mainly on the transformation of the mobile network ecosystem and the accommodation of heterogeneous services using a single infrastructure.

With the diversification of the services to be supported, Infrastructure Providers (InPs) are compelled to adapt to these new changes, requiring, in particular, the deployment of constrained services almost instantaneously. As hardware solutions are too rigid and unsuitable for real-time deployment, InPs are moving towards new scalable software solutions available with the advent of the Network Function Virtualization (NFV) paradigm and Software-defined networking (SDN). Thanks to these two technologies, a significant concept in 5G was born : Network slicing (NS).

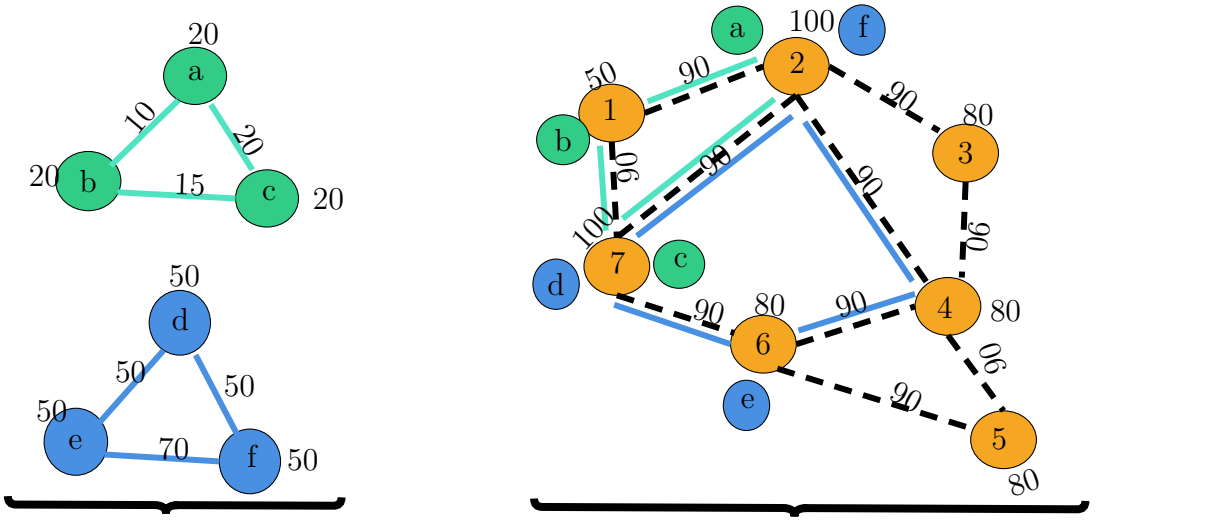
NS is one of the key technologies in 5G and beyond 5G networks (B5G). It enables the coexistence of several heterogeneous virtual networks (VNs) (i.e., slices) on top of the same physical infrastructure. In addition, NS provides network infrastructure providers (InPs) with new opportunities to increase their revenues. In fact, they will be more involved in the resource-provisioning process by leasing their infrastructure to third parties. NS holds much promise for 5G networks but not without some challenges. Network slice placement is considered one of them. It is known in the literature as Virtual Network Embedding Problem (VNEP). VNEP is known to be NP-Hard [40], and several families of solutions have been proposed to solve it : Exact, Heuristics, meta-heuristics, and finally, machine learning-based approaches.

While exact solutions ensure optimality, they are applicable only to small network

instances [49]. For large networks, heuristics may find a solution within a reasonable time frame, but they are sub-optimal [36]. To avoid this issue, meta-heuristics were proposed. They bring more flexibility, and more efficiency in the exploration process [85], but they lack efficiency in the solution computation process. Indeed, they don't accumulate knowledge, and a new search process must be started from scratch for every new instance. More recently, some works unveiled the potential of deep reinforcement learning to solve the VNEP [99, 128]. This category of approaches is more promising since we can overcome the issues stated before and learn from past experiences with reinforcement learning. With deep learning techniques, we can extract the most relevant features automatically. However, in some cases, they rely on manual feature extraction or use neural networks that are not suitable for graph-structured data. Moreover, they put particular assumptions on the shape of the substrate and virtual networks. Besides, these techniques are very slow, which could hinder their adoption to control real networks. Moreover, due to the exploration-exploitation dilemma which characterizes the training process of DRL, the learning may lead to choose an infeasible solution and not converge to an optimal solution. The thing that may cause the rejection of critical network services as well as a decrease in revenues for InPs.
In this chapter, we focus on developing deep reinforcement learning-based solutions for the VNEP problem that overcome the limitations of existing methods. The following two approaches are explored :

— GCN Based solution (Section 3.3) : Since the input of the VNEP is represented by graph-structured data, in this section, we propose a new solution for VNEP based on DRL and Graph Convolutional neural networks (GNCs). The aim is to learn to solve the VNEP automatically without putting any constraints on the shape of the physical network or the slices to place.

— Learn to improve (Section 3.4) : To deal with the exploration-exploitation challenge, in this part, we propose a new way of using DRL to solve the VNEP. Instead of learning a solution from scratch, which may result in unsafe learning, we propose to train an agent to reduce the optimality gap of VNEP heuristics. This will lead to reliable solutions based on the DRL. Also, this will ensure that the worst-case result is equal to the one obtained with the heuristic. To achieve this objective, we model the process of improving the quality of the heuristics as a reinforcement learning problem, where we train an agent to find the best strategy (policy). In each iteration of the training, the RL agent tries to improve the quality of the placement of the



Two virtual networks requests to be placed on top of the physical network

Substrate network and a feasible placement of the two virtual network requests

FIGURE 3.1 – Virtual network embedding problem

slices sequentially given by a heuristic.

3.2 Problem Formulation

A network slice is considered as a set of Virtual Network Functions (VNFs) interconnected by a set of virtual links. Note that in what follows, the terms network slices and Virtual network requests are used interchangeably. In this section, we describe the models of the Substrate Network (SN), the Virtual Network Requests (slices), and their resources. Then, we define formally the VNE problem and the metrics to optimize, and by which we quantify the quality of the placement. The notations used in this section are summarized in Table 4.3.

3.2.1 Networks and resources Models

The susbtrate network is defined by an undirected graph $\mathcal{G}^s = (\mathbb{N}^s, \mathbb{L}^s)$, where $\mathbb{N}^s$ represents the set of substrate nodes and $\mathbb{L}^s$ the set of substrate links. The number of the substrate nodes and substrate links are $|\mathbb{N}^s|$ and $|\mathbb{L}^s|$, respectively. Each substrate node n^s has a computing capacity c_{n^s} and each substrate link l^s has a bandwidth capacity b_{l^s}.

Notation	Description
$\mathcal{G}^s$	The substrate network graph
$\mathcal{G}^v$	The Virtual Network Request (VNR) graph
$\mathbb{N}^s$	Set of substrate nodes
$\mathbb{N}^v$	Set of VNFs
$\mathbb{L}^s$	Set of substrate links
$\mathbb{L}^v$	Set of Virtual Links (VLs)
$\mathbb{P}^s$	Set of substrate paths
c_{n^s}	Available CPU at substrate node n^s
b_{l^s}	Available bandwidth of substrate link l_s
c_{n^v}	CPU request of VNF n^v
b_{l^v}	Bandwidth request of VL l_v

TABLE 3.1 – Notations

Similarly, the VNR is represented by a directed graph $\mathcal{G}^v = (\mathbb{N}^v, \mathbb{L}^v)$, where $\mathbb{N}^v$ represents the set of VNFs and $\mathbb{L}^v$ represents the set of virtual links. The numbers of VNFs and VLs are $|\mathbb{N}^v|$ and $|\mathbb{L}^v|$, respectively. Each VNF n^v has a computing demand denoted as c_{n^v}, and each VL l^v has a bandwidth demand b_{l^v}.

3.2.2 Virtual Network Embedding Problem (VNEP)

The VNE problem can be divided into two stages : the virtual node mapping `VNM` and the virtual link mapping `VLM`. The former is defined as a function $f_{\texttt{VNM}}\colon\ N^v \rightarrow N^s$ that maps a virtual node to a substrate node. It must be injective. That is, two VNFs of the same VNR can not be hosted by the same substrate network. On the other hand, `VLM` can be modeled as a function $f_{\texttt{VLM}}\colon\ \mathbb{L}^v \rightarrow \mathbb{P}^s$ that maps a virtual link to a physical path (set of physical links).

At the node mapping level, a virtual node is successfully deployed if the substrate node that hosts it has sufficient CPU resources, i.e. if

$$c_{n^v} \leq c_{f_{\texttt{VNM}}(n^v)}, \quad \forall n^v \in \mathbb{N}^v. \tag{3.1}$$

At the link mapping stage, a VL is successfully deployed if its virtual nodes are deployed and if each physical link belonging to the physical path on which it is mapped has sufficient bandwidth, i.e, if

$$b_{l^v} \leq \min_{l^s \in f_{vlm(l^v)}} b_{f_{\texttt{VLM}(l^v))}}, \quad \forall l^v \in \mathbb{L}^v. \tag{3.2}$$

In this work, the aim is to maximize the number of accepted VNRs. A VNR is accepted if and only if the placement of all its VNFs and VLs satisfies, respectively, the constraints (3.1) and (3.2).

3.2.3 Optimization objectives

In this work we consider two objective functions a) maximization the number of slices placed ; and b) maximization of the revenue to the cost metric.

Maximizing the acceptance rate

The maximization of the number of accepted VNR objective function is given by the following equation :

$$\max \sum_{i} \mathcal{G}_i^v \tag{3.3}$$

where $\mathcal{G}_i^v$ represents the i-th VNR to be placed. A VNR is accepted if and only if all its nodes and links mapping satisfy, respectively the constraints (3.1) and (3.2).

Maximizing the Revenue to the Cost (R2C)

To quantify the quality of the embedding, several works used the revenue generated by embedding a request R as well as its cost C [32]. The revenue and the cost of embedding a VNR $\mathcal{G}^v$ are defined respectively as follows :

$$R(\mathcal{G}^v) = \sum_{n^v \in \mathbb{N}^v} c_{n^v} + \sum_{l^v \in \mathbb{L}^v} b_{l^v} \tag{3.4}$$

$$C(\mathcal{G}^v) = \sum_{n^v \in \mathbb{N}^v} c_{n^v} + \sum_{l^v \in \mathbb{L}^v} b_{l^v} \times |\mathbb{P}_{l^v}^s| \tag{3.5}$$

where $|\mathbb{P}_{l^v}^s|$ represents the length of the physical path that hosts the virtual link l_v. Since we want both : reducing the cost and increasing the revenue, we consider a new metric called Revenue to Cost (R2C), which is defined as follows :

$$R2C(\mathcal{G}^v) = \begin{cases} \frac{R(\mathcal{G}^v)}{C(\mathcal{G}^v)}, \text{ if } \mathcal{G}^v \text{ is accepted} \\ 0 \text{ otherwise} \end{cases}$$

3.3 GNN based solution

3.3.1 Overview

To deal with the limitations of existing solutions of the VNEP, in this section, we propose a new solution that combines DRL and a novel neural network architecture based on graph convolutional networks and the attention mechanism. In our context, the attention will let us learn how to dynamically get a graph representation based on nodes representation instead of using a fixed method. The remainder of this section is organized as follows : in Subsection 3.3.2, we present a brief background on GCNs. Next, we formulate the VNEP as an MDP. Then in Subsection 3.3.4 we present the neural network architecture we used to solve it as well as the training method we choose to update its parameters. Last, the simulation setup and the performance evaluation parts are presented.

3.3.2 Graph Convolutional neural networks

As stated earlier, Graph Neural Networks (GNNs) are a type of neural networks dedicated to graph-structured data. They were introduced in [103] and several variants of it have been proposed [53, 116, 126]. The most known one is the Graph Convolutional Networks (GCN), which generalizes the convolution operation from euclidean data (images and grid-shaped data) to non-euclidean data (graphs). The key objective of GCNs is to learn a function that generates for each node in a graph a vector representation by aggregating its own features and its neighbors' features.

Let $\mathcal{G} = (V, E)$ be a graph, where V is the set of its nodes and E the set of edges. We assume that each node $n \in V$ is characterized by an input feature vector in_n. The graph convolution operation takes as input the feature vector in_n and outputs a new vector out_n according to the following equation :

$$out_n = f\Big(\sum_{m \in \mathcal{N}(n)} \frac{1}{\sqrt{d_m d_n}} in_m W^l\Big) \tag{3.6}$$

where $\mathcal{N}(n)$ represents the set composed of node n and its neighbours, d_m is the degree of node m plus one, W^l is the weight matrix of the l^{th} GCN layer, and f is an activation function.

3.3.3 VNEP as an MDP

Reinforcement Learning (RL) is the area of machine learning that deals with sequential decision-making tasks. It is formulated as the interaction of an agent with an environment to optimize a given total amount of return. This interaction can be modelled as a Markov Decision Process (MDP) M = ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$) with a state space $\mathcal{S}$, an action space $\mathcal{A}$, transition dynamics $\mathcal{P}$ and a reward function $\mathcal{R}$. At each time step t, the agent observes a state s_t and performs an action a_t, then it receives a reward r_t as well as the next state of the environment s_{t+1}. If the agent receives a particular state called *terminal*, the interaction stops. The objective of the agent is to maximize the expected cumulative discounted return :

$$R_t = \mathbb{E}[\sum_t \gamma^t r_t], \quad \gamma \in [0, 1]. \tag{3.7}$$

The parameter γ is the discount factor, by which we ensure the convergence of the cumulative return in the infinite horizon setups. The agent's behavior is defined by a policy π, which maps a state s to a distribution over the actions A. Each state s is associated with a value function $V^\pi(s)$ that maps it to a scalar corresponding to the expected reward the agent will receive in this state and acting according to the policy π :

$$V^\pi(s) = \mathbb{E}_\pi[R_t | s_t = s]. \tag{3.8}$$

In traditional RL, both the policy and the value functions are modeled by tables. However, in most realistic scenarios, the state and action spaces are large, therefore instead of the tabular representation, the policy and the value functions are approximated by Deep Neural Networks (DNNs). DNNs are proposed for two main reasons : to overcome the slowness of the tabular-based solutions and to extract automatically valuable features from the state representation. The combination of RL with DNNs is known as Deep Reinforcement Learning (DRL). To learn the policy function, several DRL algorithms were introduced. They can be divided into three main categories : value-based methods, policy-based methods, and actor-critic methods. The policy-based solutions parameterize the policy function directly by a neural network architecture and try to find the optimal policy using gradient ascent. In contrast, the value-based solutions estimate the value of each state using a DNN and then infer the optimal policy using this estimation. Finally, the actor-critic methods combine the above methods : the agent learns both functions to find the optimal policy. One of the most widely used actor-critic algorithms is A2C [79].

Now that we defined the key parts of an RL problem, the next step will be the MDP formulation of the VNE problem and its main components, namely : the state representation, the actions, the rewards, and the transition dynamics.

State Representation : The state s is represented by the two graphs $(\mathcal{G}^s, \mathcal{G}^v)$. Each

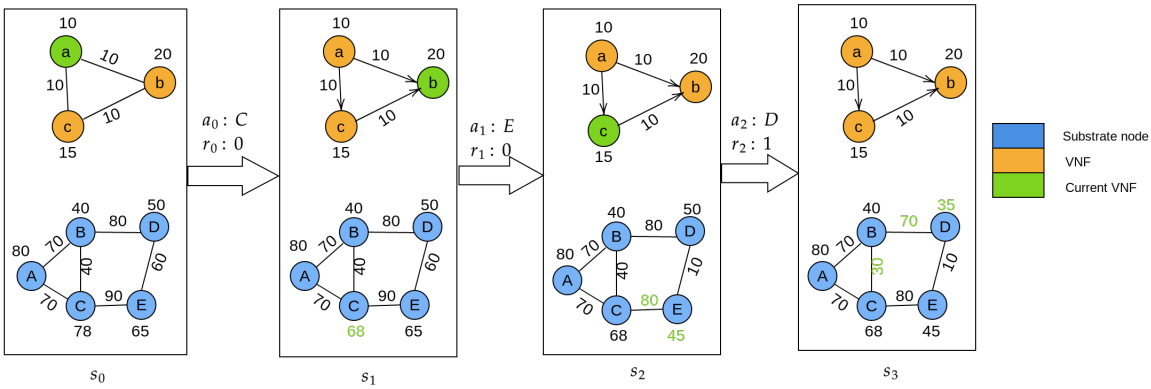


FIGURE 3.2 – An overview of the proposed iterative VNE process.

node $n^s \in \mathbb{N}^s$ has two features : (1) available CPU c_{n^s}, (2) sum of bandwidth defined as the total available bandwidth of links to which belongs n^s. On the other hand, each VNF $n^v \in \mathbb{N}^v$ is characterized by four features : (1) CPU required by the VNF, (2) total bandwidth required by the links to which belongs the VNF, (3) a flag indicating whether this node is the current node to place, (4) a flag indicating whether the VNF was placed. At the initial state s_0, the current flag of the first VNF is assigned to 1 and the others to -1, and the placement flag is set to -1 to all VNFs.

Action Space : At the time step t in the MDP, the agent has to select a substrate node to host the positive-flagged VNF (+1). Note that the VNFs of the same VNR should not be placed at the same substrate node. Hence the action space at time step t is defined as follows :

$$\mathcal{A}_t = \mathbb{N}^s \backslash \mathbb{S}_t \tag{3.9}$$

where $\mathbb{S}_t$ represents the set of substrate nodes selected before the time step t. Initially, the action space $\mathcal{A}_0$ is equal to $\mathbb{N}^s$.

State Transition Dynamics : At time step t, the agent observes the state s_t and selects an action $a_t = n^s$ to place the current VNF n^v. The MDP, then, transits to a new state s_{t+1} in which the node features of the two graphs experience the following updates :

— If the selected substrate node has sufficient CPU resource, the VNF n^v will be placed at the substrate node n^s. Then, the CPU resource of n^s is updated by $c_{n^s} := c_{n^s} - c_{n^v}$

— After each successful VNF placement, the VLs related to the VNF and having both head and tail VNFs successfully deployed are considered. The shortest path $l^s \in \mathbb{P}^s$ between the head VNF and the tail VNF is computed. If the bandwidth required by this VL is less than the one offered by all physical links belonging to the obtained physical path, then the available bandwidth of each physical link in this physical path is updated as follows :

$$b_{l^s} = b_{l^s} - b_{l^v}, \quad \forall l^s \in \mathbb{P}^s. \tag{3.10}$$

— If the previous conditions are satisfied, the placement flag of the current node is set to 1, and another non visited VNF is chosen as the current node.

The placement process of the VNR will at most ends in $|\mathbb{N}^v|$ steps.

Reward design : The objective is to maximize the number of accepted requests. To guide the agent to this objective, we introduce intermediate and final rewards. When a VNF is placed successfully, it receives an intermediate reward of 0. Final rewards quantify the quality of the final decision made by the agent. Thus, the agent obtains a unit of reward when it successfully places a VNR. Otherwise, it gets a reward of -1 if the VNR is not placed successfully, i.e., if there exists a resource violation at some step in the process.

3.3.4 The agent architecture : Design and training

DRL agent design

To solve the MDP defined above, we adopted the actor-critic framework [56]. The DRL agent has to approximate the policy and the state value functions. To implement the actor-critic framework, we can either use two distinct neural network architectures or one neural network architecture with two heads, one representing the policy function and the other one the value function. We adopt the last approach because it lets the actor and critic share the same low-level features and use them in different ways.

The two functions are parameterized by neural networks. Their inputs must then be a real-valued vector. However, as we stated earlier, the state is represented by two graphs (SN and VNR). Hence we must encode the graph-structured information of the state into a real-valued vector.

To achieve this, we proposed an end-to-end architecture represented in Fig. 3.3. It takes the state of the environment as input and encodes it to a real-valued vector which is fed to the two heads described above. Finally, the output corresponds to the prediction of the policy and the value functions. The key idea is that learning a good representation of the state will lead to a fast and accurate optimal policy prediction.

Y. Bai [10] proposed a GCN-based architecture to solve the graph similarity computation task using supervised learning. We adapt their architecture to our problem for the state encoding steps. The encoding of the state into a real valued vector involves three main steps :

— **Node-level encoding using GCNs.** This stage transforms each node of the graph into a vector encoding its features and structural properties. Based on the raw features on each node of the two graphs $\mathcal{G}^s$ and $\mathcal{G}^v$, we use GCNs to aggregate neighborhood information of each node (Eq. (3.6)). After multiple layers of GCNs, we obtain two matrices $U^s \in \mathbb{R}^{|N^s| \times D}$ and $U^v \in \mathbb{R}^{|N^v| \times D}$ that represent, respectively, the node-level encoding of SN and VNR. Each row u_n represents the encoding of node n and D is an hyperparameter representing the dimension of u_n. Note that we used two different GCNs modules to encode SN and VNR.

— **Graph level encoding using attention layers.** After getting the node level encoding matrices U^s and U^v, we can directly use them to encode the graph structure. However, this can be computationally expensive. To overcome this issue and make the representation of the graph independent of its size, we can perform a weighted sum of the nodes' representations. The choice of nodes that get higher weights depends on the current node we want to place. Thus, we use an attention mechanism to learn these weights instead of considering fixed ones. For each graph, we first compute a graph context ct, then we compute the attention weight for each node in the graph.

For the VNR we define the context ct_v as the node representation of the current VNF followed by a nonlinear transformation, $ct_v = g_v(u_c \times W_c)$, where $u_c \in \mathbb{R}^D$ is the node representation of the current VNF, $W_c \in \mathbb{R}^{D \times D}$ is a learnable weight matrix and g_v is an activation function. Based on ct_v we compute one attention weight a_{n^v} for each VNF n^v as the inner product between the context ct_v and its encoding u_{n^v} : $a_{n^v} = u_{n_v}^T ct_v$. Finally the VNR encoding h_v is given by

$$h_v = \sum_{n^v \in \mathbb{N}^v} a_{n^v} u_{n^v}. \tag{3.11}$$

Regarding the SN we define its context ct_s by a simple average of nodes encoding followed by a non linear transformation $ct_s = g_s((N^{-1}\sum_{n=1}^{|N^s|} u_n)W_s)$, where W_s is a learnable weight matrix and u_n is the encoding of the n^{th} susbtrate node. Similarly we define the encoding of the SN as a weighted sum of the substrate nodes' encodings. The weight of each node equals the inner product between its encoding and the context,

$$h_s = \sum_{n^s \in \mathbb{N}^s} a_{n^s} u_{n^s}. \tag{3.12}$$

— **State level encoding using neural tensor networks (NTN).** The previous stages let the agent learn how to model the two graphs of the state. The last step consists in learning how to model their relation. Following [107] and based on the vector representation of SN and VNR, we use Neural Tensor Networks (NTN) to achieve this :

$$g(h_s, h_v) = f_3(h_v^T W_3^{[1:K]} h_v + V\begin{bmatrix}h_s\\h_v\end{bmatrix} + b_3), \tag{3.13}$$

where $W_3^{[1:K]}$ is a weight tensor, [] denotes the concatenation operation, V is a weight vector, b_3 is a bias, and f_3 is an activation function ; K is a hyperparameter.

Training process

As stated before, we adopt the actor-critic setup. To update the parameters of the architecture we presented above, we used the Advantage Actor-Critic (A2C) algorithm [56]. A2C updates parameters of actor and critic architectures according to the following equations :

$$J(\theta) = \sum_{t=t_0}^{T} log(\pi_\theta(a_t|s_t))A^{\pi_\theta}(s_t, a_t) \tag{3.14}$$

$$\delta(\theta) = \sum_{t=t_0}^{T} H(\pi_\theta(.|s_t)) \tag{3.15}$$

$$\theta \leftarrow \theta + \beta\nabla_\theta(J(\theta)) + \phi\nabla_\theta\delta(\theta) \tag{3.16}$$

$$J(\theta_v) = \sum_{t=t_0}^{T} (r_{t+1} + v_\theta(s_{t+1}) - v_\theta(s_t))^2 \tag{3.17}$$

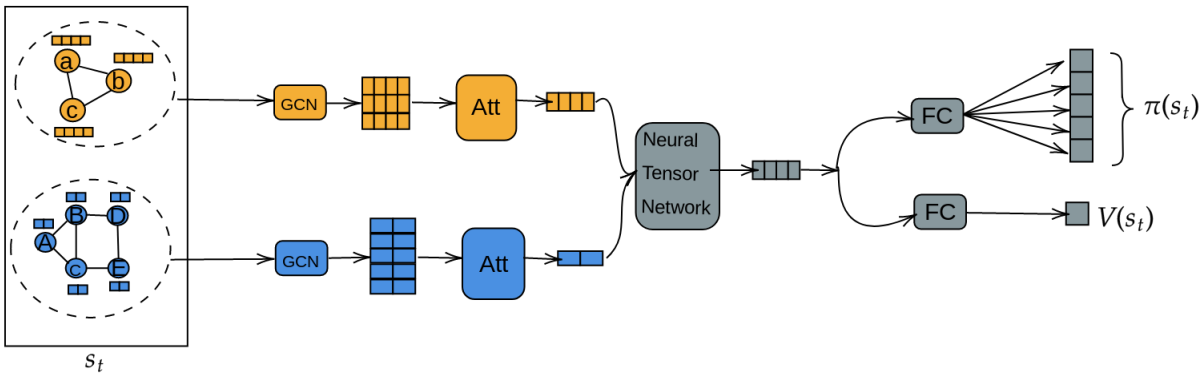


FIGURE 3.3 – An overview illustration of the proposed end-to-end agent neural network architecture. Given the state of the environment,the agent has to approximate two functions : the policy function π and the value function V.

$$\theta_v \leftarrow \theta_v + \alpha \nabla_{\theta_v} J(\theta_v) \tag{3.18}$$

θ represents the set of trainable weights in the common part as well as the trainable weight in the Actor head FC network. Similarly, θ_v represents the trainable weights in the common part and the trainable weight in the critic head FC network. The $A^{\pi_\theta}(s_t, a_t)$ represents an estimate of the Advantage function that indicates how better is the action a_t when compared to the average action from the corresponding policy under a certain state s_t. Based on the Temporal difference (TD) method we have $A^{\pi_\theta}(s_t, a_t) = r_{t+1} + v_\theta(s_{t+1}) - v_\theta(s_t)$. The $J(\theta)$ function is the performance of the Actor network. It is given by the Log-likelihood of the policy weighted by the Advantage function. The $\delta(\theta)$ is the sum of the policy entropy used as a regularization term to discourage premature convergence. The function $J(\theta_v)$ is the performance of the critic network. It is given by the squared temporal difference error of the value functions.

3.3.5 Performance evaluation and results analysis

Simulation Setup

To evaluate the performance of the proposed approach, we consider the following setup :

— The substrate network is modeled by the network topology of BtEurope [1]. It contains 24 nodes and 37 links. The capacity of the substrate nodes and links is drawn uniformly from the interval $[50, 100]$.

— To generate the virtual requests, we used the Erdős–Rényi model [28]. In this model, the generated graph is defined by the number of nodes n and the probability p of creating an edge between any pair of nodes. With, for instance, $p = 2\ln(n)/n$, the generated graph is in general connected (more precisely, the probability that it is the case goes to one as $n \to \infty$). The requested resources (CPU and BW) of VNRs are drawn randomly following a uniform distribution from the interval $[1, 10]$. The system operates in an episodic manner ; during each episode, VNRs come to the system sequentially : once a VNR is processed, the next one arrives. There are 3000 episodes and during each one 30 VNRs come to the system. At the end of each episode, the VNRs leave the system. We execute 5 runs with different seeds.

— For the model architecture, it is written in `Python` with the `Pytorch` library [82], and the DGL library [117]. The neural architecture is constructed with the following hyperparameters. We set the numbers of GCN layers to 3, and we use *tanh* as the activation function. For the NTN layer, we set K to 8. We then use two fully connected layers, and finally, we have two heads, each one modeled by a fully connected layer that represent, respectively, the policy network and the value network. The learning rate is set to 10^{-3}, and the discount factor γ is set to 0.99. To train the model, the Adam optimizer was used [52]. These hyperparameters were chosen based on the results of a grid search [58]. We tested different combinations of these hyperparameters and we kept the combination leading to the best performance.

Impact of the hidden units

The number of hidden units used in each layer of the agent's neural architecture impacts both the quality of the solution obtained and the computational cost to get it.

In this section, we study the effect of the number of hidden units on the performance of the DRL agent in order to find a trade-off between these two metrics. We consider four different agents : agent16, agent32, agent64, and agent128, which have, respectively, 16, 32, 64, and 128 hidden units in all their hidden layers.

The results illustrated in Figures 3.4, 3.5 and 3.6 show, respectively, the performance

1. http ://www.topology-zoo.org/dataset.html

in terms of the number of accepted requests, number of VNFs deployed, and number of VLs deployed. As we see in these figures, the agents with 64 and 128 hidden units are better than the other ones, and their performance is more stable.

Figure 3.7 shows the required time to finish one episode for each configuration. The more hidden units we have, the more time we need to finish the episode. The time required to finish one episode with the agent with 128 hidden units is greater than the one with 64 hidden units, while the performance is quite the same. Consequently, we use the agent with 64 hidden units to compare its performance against other solutions.

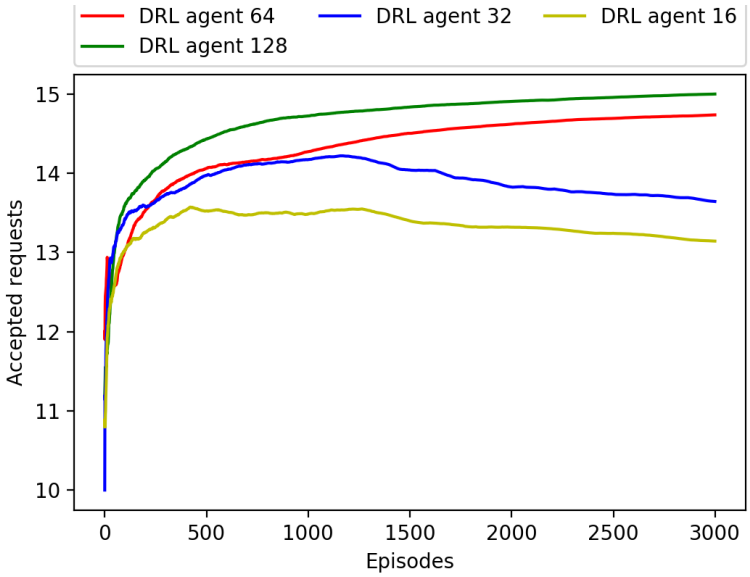


FIGURE 3.4 – Number of accepted requests vs Number of hidden units.

Performance Benchmarking

To benchmark the performance of our DRL-based solution, we compare it against the following approaches :

— **Random agent :** It performs randomly to choose the substrate nodes that will host the VNFs. Based on the obtained VNF mapping, the agent uses the shortest path algorithm to find the link mapping of the Vls,

— **First Fit :** It adopts the First-Fit (FF) algorithm to allocate the VNFs and runs the shortest path algorithm to find the physical path mapping for each VL. For each VNF, FF selects the first substrate node with enough resources.

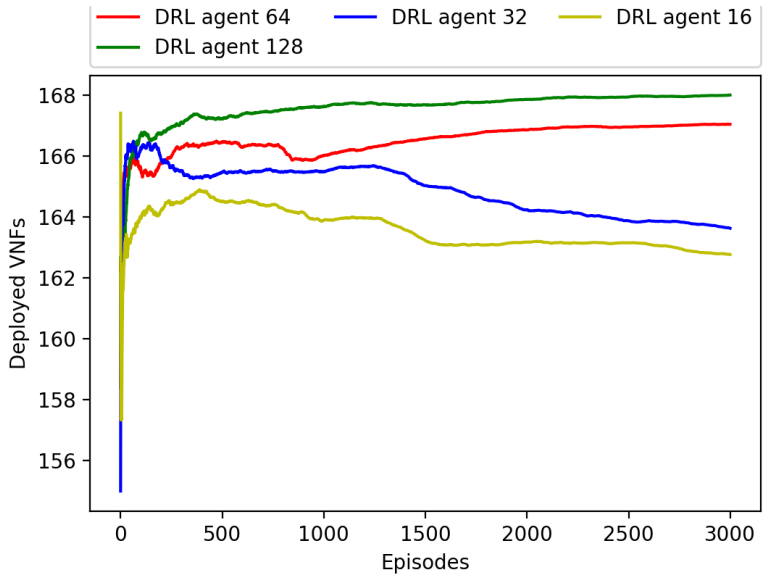


FIGURE 3.5 – Number of accepted VNFs vs Number of hidden units.

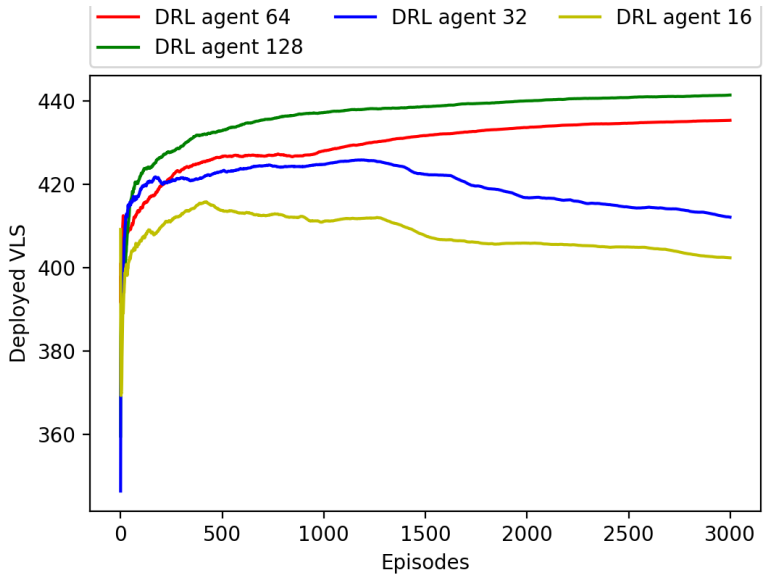


FIGURE 3.6 – Number of accepted VLs vs Number of hidden units.

Figures 3.8, 3.9 and 3.10 show, respectively, the number of accepted requests as well as the number of deployed VLs and VNFs. With respect to the three metrics, our approach outperforms the random agent and the First Fit strategies. Unlike the latter algorithm, our approach efficiently explores the possible actions until it finds the optimal ones, while First Fit tries to place the VNFs on the first substrate nodes with enough resources.

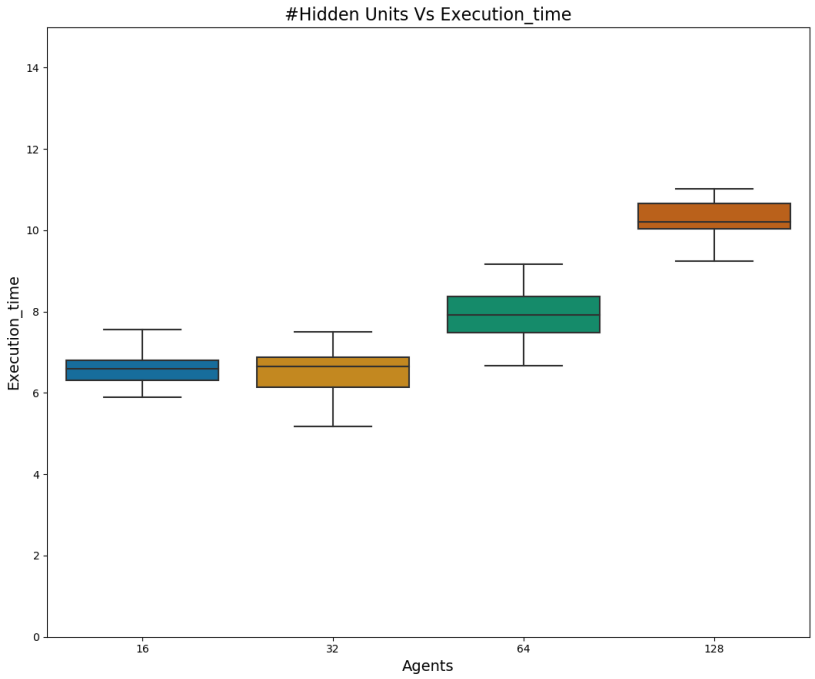


FIGURE 3.7 – Execution time(s) VS Number of hidden units.

Therefore it explores just a part of the possible solutions, resulting in huge consumption of the residual bandwidth of the physical links and then in an increased probability of rejection of new arriving requests.

3.3.6 Summary

In this section, we have proposed a methodology to address the VNEP. The proposed approach is based on the use of a Deep Reinforcement Learning technique. In particular, we have proposed the use of GCNs, which allow us to take into account the structuring of the data of the problem, and showed how to adapt the tool to the considered VNE problem. The solution also exhibits a new strategy of placement, significantly improving the performance of this type of task. The proposed solution has proven to be very effective compared to existing conventional approaches.

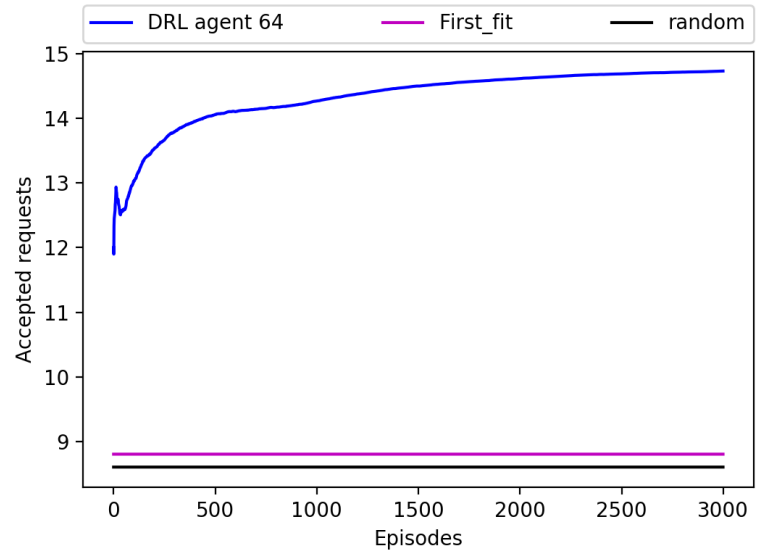


FIGURE 3.8 – Number of accepted requests.

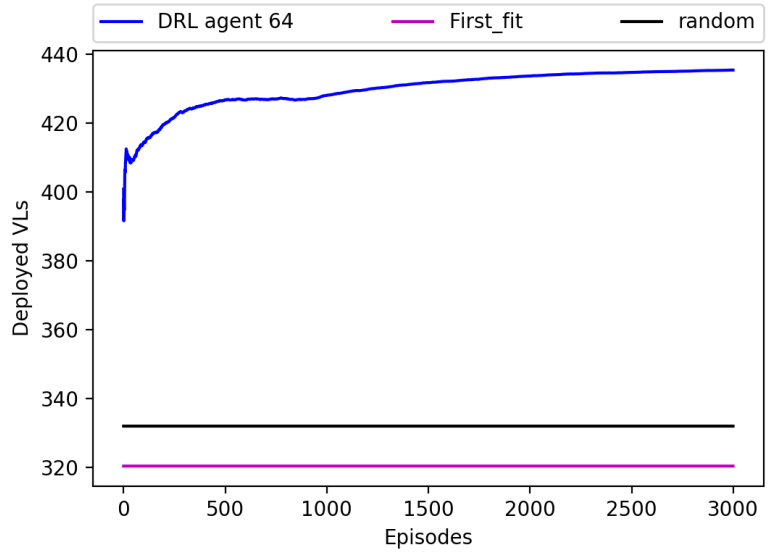


FIGURE 3.9 – Number of deployed VLs.

3.4 Improving Heuristics

3.4.1 Overview

In the previous section, we described our new method that combines DRL and GCNs to solve the VNEP. Despite its advantages, some challenges remain. Because of the

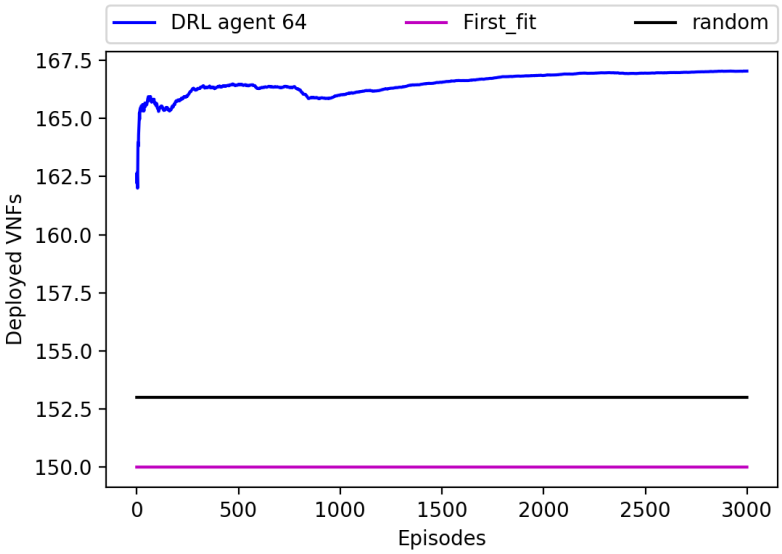


FIGURE 3.10 – Number of deployed VNFs.

exploration-exploitation dilemma, the learning may lead to unsafe actions and, therefore, the rejection of a slice placement. To cope with this matter, in the following section, we propose a safe DRL-based solution for the VNEP. We mean by a safe training, a training which does not violate the constraints of the placement. Our solution trains a DRL agent to reduce the optimality gap of the heuristics sequentially. Our main contributions are the following :

- We propose a safe DRL-based solution to the problem of the placement of network slices by learning how to reduce the optimality gap of heuristic-based solutions,
- We formalize the task as a Markov Decision Process (MDP),
- We represent the states of the system in a novel way using heterogeneous graphs,
- In order to automate the features extraction process and let the agent benefit from the structural information, we model its policy using a Relational Graph Convolutional Neural network-based architecture,
- We design a specific reward function in order to guide the agent to improve the quality of the solutions.

.

The rest of this section is organized as follows. In subsection 3.4.2we define what is a hetergeneous graph, and then we continue by a presentation of Relational Graph Convolutional neural networks. Next, the subsection 3.4.3 presents the MDP formulation of the

problem. Later the DRL Agent architeture as well as its training strategy are presented respectively in the subsections 3.4.4 and 3.4.4. Last, the subsecion 3.4.5 illustartes the simulation setup and the numercial results.

3.4.2 Heterogeneous graphs and Relational Graph Convolutional neural networks (RGCN)

Heterogeneous graphs

A heterogeneous graph and heterograph for short [106], is defined as a directed graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{T}_n, \mathcal{T}_l)$ associated with two functions ϕ_n and ϕ_l. $\mathcal{V}$, $\mathcal{E}$, $\mathcal{T}_n$, $\mathcal{T}_l$ represent, respectively, the set of nodes, the set of edges, the set of types of nodes and the set of types of links of the heterograph. $\phi_n \colon \mathcal{V} \to \mathcal{T}_n$ maps each node $v \in \mathcal{V}$ to a single node type $\phi_n(v)$, while $\phi_l \colon \mathcal{E} \to \mathcal{T}_l$ maps each link $e \in \mathcal{E}$ to a link type $\phi_l(e)$.

Besides, $\mathcal{T}_n$ and $\mathcal{T}_l$ satisfy the following condition :

$$|\mathcal{T}_n| + |\mathcal{T}_l| > 2 \tag{3.19}$$

Relational Graph Convolutional Neural Networks (RGCN)

Graph Convolutional Neural Networks(GCN)[53] were proposed as a generalization of the convolution operation on arbitrary graph-structured data. The aim of GCN is to extract features automatically from graphs. However, GCN can deal only with homogeneous graphs where we have only one type of edges between nodes (i.e., $|\mathcal{T}_l| = 1$). Hence, RGCN [104] was defined as an extension of GCN to extract features from heterographs.

The key idea behind RGCN is that the encoding of a node is based on the encoding of its neighbors under all types of links.

Let $\mathcal{H}$=($\mathcal{V}$,$\mathcal{E}$,$\mathcal{T}_n$,$\mathcal{T}_l$) be a heterograph. We assume that each node $v \in \mathcal{V}$ has an input vector in_v. Each layer of an RGCN network takes as input the feature vector in_v and outputs a new vector out_v according to the following equation :

$$out_v = \sigma\Big(\sum_{r \in \mathcal{T}_l} \sum_{j \in \mathcal{N}_v^r} \frac{1}{c_{v,r}} W_r in_j\Big) \tag{3.20}$$

where $\mathcal{N}_v^r$ represents the neighbors of node v under the link type r, $c_{v,r} = |\mathcal{N}_v^r|$ represents a normalization constant and finally W_r denotes the trainable weight matrix associated with the link type r.

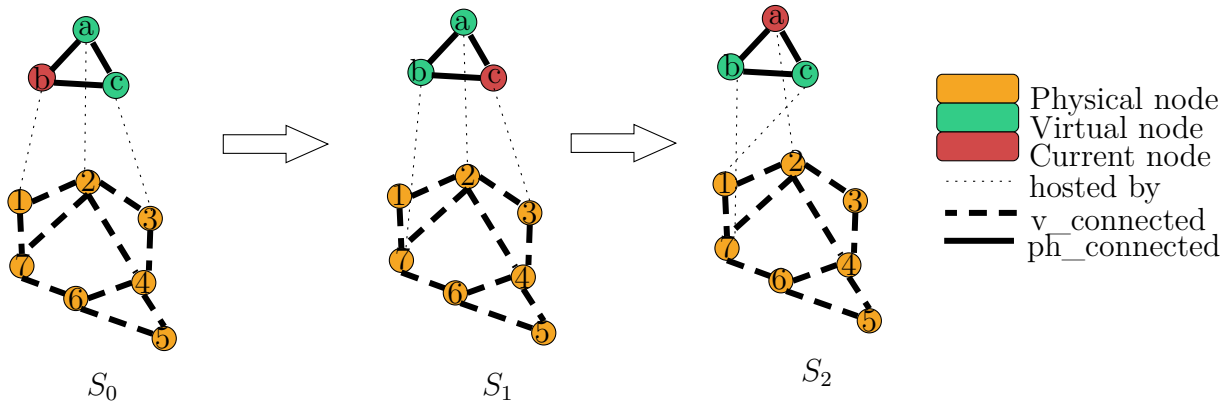


FIGURE 3.11 – Illustration of the sequential process of Improvement of quality of VNE heuristics. The state is represented using an heterograph with two types of nodes and three types of links. At each step either the placement of a virtual node is modified or kept.

3.4.3 Improving the Quality of VNEP Heuristics (IQH) as an MDP

Given a VNE problem and a heuristic to solve it, the process of improving the quality of heuristics consists in starting with this initial solution and then modify it in order to increase its R2C score. Instead of modifying the solution on a one-shot basis, we do it node by node. We define IQH as a sequential MDP where the states, actions, and rewards are defined as follows :

— **States :** In RL, the state represents the raw information the agent gets from the environment and based on which it chooses an action. In this work we define the state as a hetero-graph that illustrates the solution of the heuristic. The hetero-graph is composed of 2 types of nodes and 3 types of edges : *'v_connected'*, *'hosted_by'* and *'ph_connected'*. The two categories of nodes represent the virtual and substrate nodes, while the categories of edges represent the type of possible relation between these nodes. Each virtual node is connected to another virtual node under the relation *v_connected* and each substrate node is connected to another substrate node under the relation *s_connected*, and to represent the solution given by the heuristic, each VNF may be connected to a substrate node via *hosted_by* type relation.
Each node has 3 initial features. For the virtual nodes : (1) the CPU required by

the VNF c_{n^v}, (2) the sum of requested bandwidth defined as the total bandwidth requested by the virtual links to which belongs the node ; and (3) a flag indicating if the VNF is the current VNF to process. On the other hand each substrate node has the following features : (1) the remaining amount of CPU c_{n^s}, (2) the sum of bandwidth defined as the total remaining bandwidth of links to which belongs the substrate node ; and (3) the number of its neighbors. Initially, a random VNF (virtual node) is selected as the current node.

— **Actions :** At each time step t, the agent has to select either to keep the same placement of the current VNF or to modify it into another substrate that does not host any other VNF from the same request. Hence the action space at time step t is defined as follows :

$$\mathcal{A}_t = \mathbb{N}^s \backslash \mathbb{S}_t \tag{3.21}$$

where $\mathbb{S}_t$ represents the set of substrate nodes that host a VNF of the current slice at time step t excepting the current node. For example, as depicted in Figure 3.11 at step 0, $\mathcal{A}_0 = \{1, 7, 6, 4, 5\}$.

Once the action for the current node is computed, we check its feasibility (i.e., chosen node has enough resources) and then we try updating the link mapping of edges, where the current VNF is head or tail. We run the shortest path algorithm between the new chosen substrate node and the placement of its neighbors under the relation '*connected_v*'. If the link mapping is feasible the agent will receive a new state representation, where the '*hosted_by*' link of the current VNF is updated according to the new computed node and a new virtual node is marked as the current node. If a failure occurs either on the node or on the link mapping stages, the action is not applied and the agent gets a new state representation, where only the current VNF is changed. Besides, after each step the agent gets a reward related to the action it made.

— **Rewards :** The reward is the signal by which an RL agent can measure the correctness of the action it made. It can be positive in the successful cases or negative if the action leads to an undesirable behavior. The process to compute the reward obtained at each time step t is presented in Algorithm 1. The objective is to improve the R2C score of the solution given by the heuristic by updating the placement of virtual nodes sequentially, hence at each time step t if the action a_t leads to an unfeasible solution the agent gets a negative reward of -100 and we keep the last

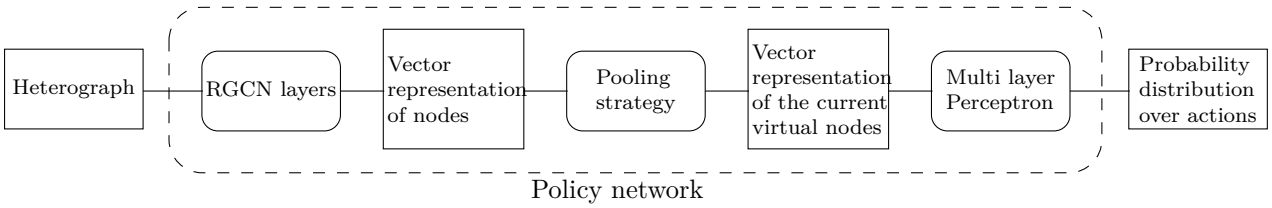


FIGURE 3.12 – The neural architecture of the policy function.

feasible placement. Otherwise, the reward obtained is $r_t = rc_t - bp_t$, where rc_t is the new R2C score after updating the placement of the t^{th} node and bp_t is the best R2C score found before time step t. Initially bp_0 is the R2C score of the placement of the heuristic.

Algorithm 1 Compute reward

```
1: function GETREWARD(bp, r2c)
2:     reward ← 0
3:     if r2c = 0 then
4:         reward ← −100                          ▷ unfeasable solution
5:     else
6:         reward ← (r2c − bp)
7:     if r2c > bp then
8:         bp ← r2c                               ▷ new best score
9:     return reward, bp
```

3.4.4 DRL Agent : Architecture and training

In this section we present the neural network-based architecture of the policy as well as its training process.

Agent architecture

To choose actions, the policy of the agent is parametrized using deep neural networks and trained using the policy gradient descent algorithm.

The architecture of the agent is depicted in Figure 3.12. The policy is a function that maps each state's representation to a probability distribution over the actions. At each time step t given the state representation, the policy network outputs a distribution

probability over the actions. However, since the state is represented by a heterograph, the first part of the policy network is in charge of encoding this information. This step generates a vector representation of all nodes in the heterograph using an RGCN network, then the vector of the current virtual node is selected and fed to a Multi-Layer Perceptron (MLP).

Training

To optimize the neural network architecture of the policy network and update its trainable parameters, we adopt the Reinforce [119] algorithm. Reinforce is a policy-gradient algorithm, in other words it updates the policy network parameters to optimize an objective function using gradient descent. The objective is to maximize the cumulative discounted return :

$$R_t = \mathbb{E}[\sum_t \gamma^t r_t], \quad \gamma \in [0, 1]. \tag{3.22}$$

where γ is the discount factor, by which we ensure the convergence of the cumulative return in the infinite horizon setups. The gradient of this objective is defined by :

$$\nabla R_t = \mathbb{E}[\log(\pi_\theta(a_t|s_t))Q_{\pi_\theta}(a_t, s_t)] \tag{3.23}$$

where $Q_{\pi_\theta}(a_t, s_t)$ represents the expected cumulative reward by choosing a_t in s_t and following π_θ. θ represents the set of trainable parameters of π
With Reinforce, this gradient is estimated using a Monte-Carlo approach. During each episode a rollout (s_t, a_t, r_t) is performed using the current policy π_θ. At the end of each episode a gradient ascent step is performed and θ are updated as follows :

$$\theta \leftarrow \theta + \sum_{t=0}^{T} \alpha \log(\pi_\theta(a_t|s_t))(\sum_{k=t}^{T} r_k - b_k) \tag{3.24}$$

b_t represents a baseline for reducing the variance of the estimate and it is equal to the average of the cumulative reward starting from time step t, $b_t = \sum_{k=t}^{T} r_k - b_k$. This way, the trainable parameters are updated such that the probability of actions leading to a cumulative rewards higher than the average rewards is increased.

Parameters	Values
Learning rate α	5×10^{-3}
Discount Factor γ	0.99
Optimizer	Adam [52]

TABLE 3.2 – DRL agent Hyperparameters

3.4.5 Performance evaluation and results analysis

Simulation setup

In this section, we present the simulation setup and an analysis of the obtained results. To assess the performance of our approach we consider the following simulation setup. For the substrate network, we consider a random topology generated following the Waxman random graph model [118] , using the parameters $\alpha = 0.5$ and $\beta = 0.2$; this methodology for topology generation has been commonly used in previous works [32]. Given this configuration, the generated substrate network will contain 100 nodes and 500 edges. The CPU and bandwidth of the substrate nodes and links are drawn uniformly from the interval [50,100].

To generate the virtual requests topology, we adopt the Erdős–Rényi model [28]. The generated topology is defined by the number of nodes n and the probability p for the creation of the edge. With $p = 2\frac{\ln(n)}{n}$, the generated graphs are in general connected. The CPU and bandwidth requested by the VNR are drawn uniformly from the interval [10,20]. The number of VNFs in each VNR is randomly chosen from the interval [8,10].

For the DRL agent, the policy architecture is set up with the parameters depicted in Table 3.2. The RGCN part is represented by a 2-layer RGCN network. The first layer contains 16 hidden units and the second one 32 hidden units. The fully connected layer that maps the vector encoding the current node to the actions contains 100 units. We adopt a LeakyReLu activation for all layers except for the last one, where we used softmax activation so as to get probabilities. These hyperparameters were chosen based on the results of a grid search [58]. We tested different combinations of these hyperparameters and we kept the combination leading to the best performance.

Each time a VNR arrives, the heuristic finds first the initial placement, and then an episode of quality improvement starts. There are 30000 episodes. We execute ten runs with different seeds. The model architecture is written in Python using the Pytorch [82] and the DGL [117] libraries.

Heuristics

We evaluated our method in learning to improve the following heuristics :

— **First-Fit** : An heuristic that maps a virtual node into the first substrate node with sufficient CPU.

— **Best-Fit** : Heuristic that maps a virtual node to the substrate node with the smallest sufficient CPU.

For both heuristics, the virtual links mapping is computed by running the shortest path algorithm between the placement of VNFs.

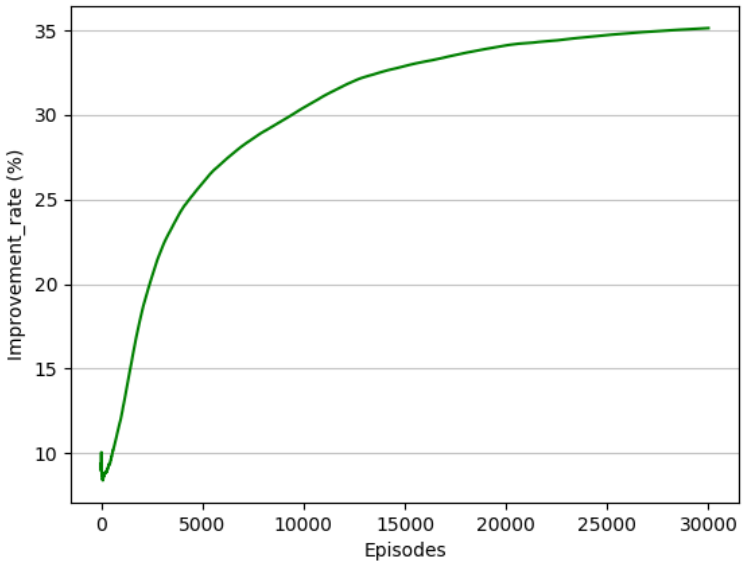


FIGURE 3.13 – Improvement rate while using First Fit.

Simulation results

Figures 3.13 and 3.14 show, respectively, the improvement rates reached by the agent, for First-Fit and Best-Fit. For the First-Fit, we see that the agent could reach an improvement rate of 35%. Moreover, the performance of the agent gets better with time. On the other hand, with the Best-Fit strategy, the task was harder for the agent in the beginning, but with the negative rewards it got, the agent learned to avoid bad decisions

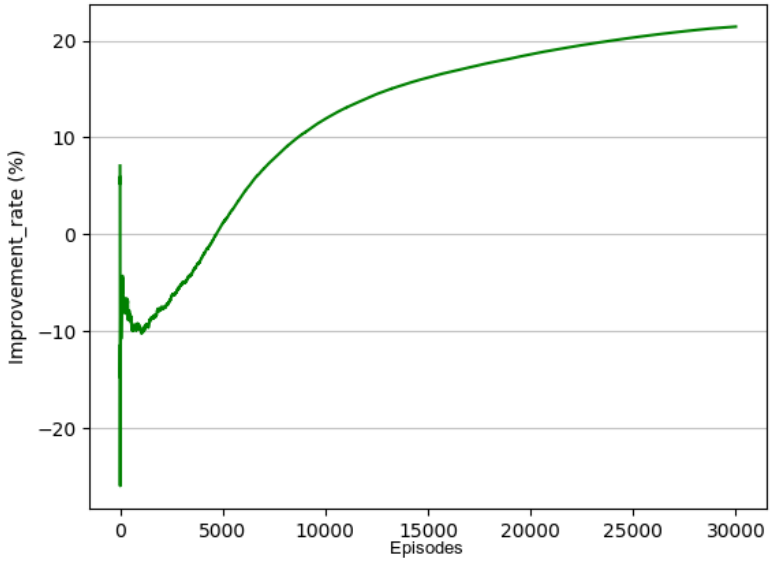


FIGURE 3.14 – Improvement rate while using Best Fit.

and converged to an improvement rate of 20%. Note that when the agent fails to improve the heuristic solution (the worst case), we don't apply its decision, but it learns from it.

To assess the robustness of the proposed approach, we modify the interval from which the number of VNFs per VNR is drawn from [8,10] to [12,14]. Figures 3.15 and 3.16 show the results using the First-Fit and the Best-Fit strategies under this setting. Even the task becomes harder, the agent learns and finds a way to improve both heuristics under this new setting. For First-Fit, the agent reached an average improvement rate of 16%, while for Best-Fit, the improvement rate was of 13%.

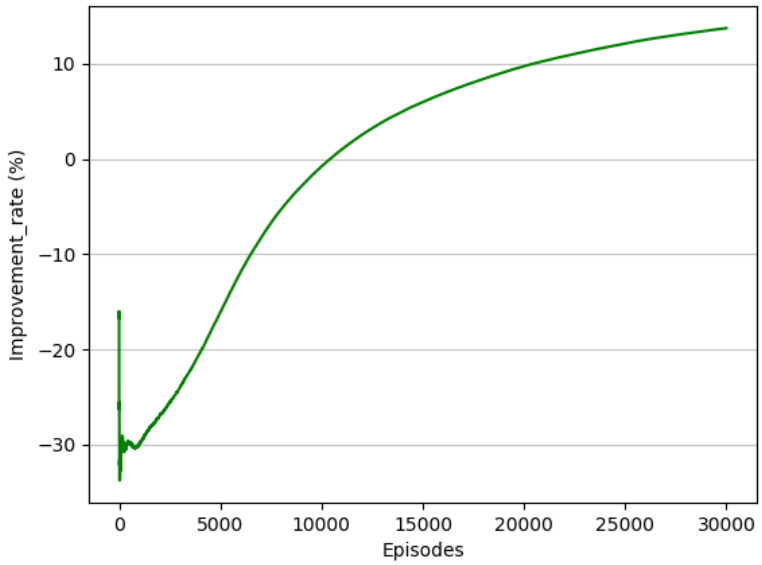


FIGURE 3.15 – Improvement rate while using Best Fit .

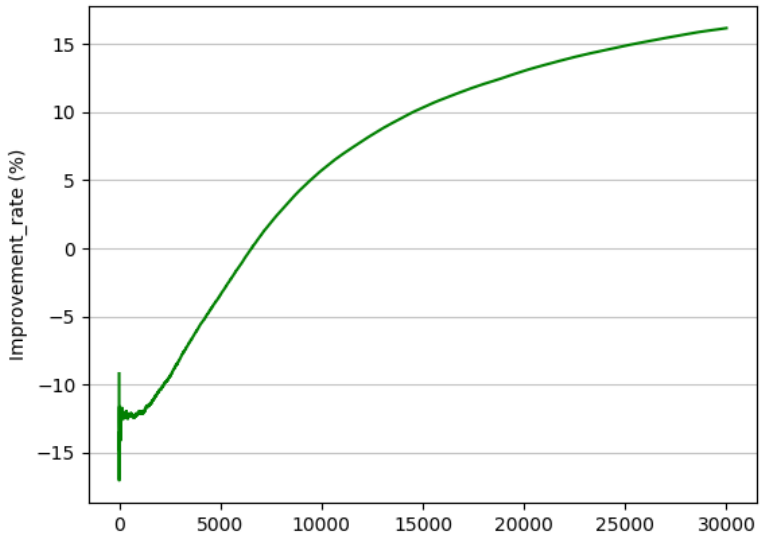


FIGURE 3.16 – Improvement rate while using First Fit.

3.4.6 Summary

In this second contribution, we unveiled the potential of deep reinforcement learning to solve the VNEP in a novel way. Instead of learning a solution from scratch using DRL, we trained a reinforcement learning agent to reduce the optimality gap of heuristics dedicated to solving this problem. We modeled the task as a sequential Markov decision process, where the states are represented using heterogeneous graphs. To exploit the graph information, we exploited RGCN to parameterize the policy network, and in order to guide the agent to the desired behavior, we defined a reward function that gives the agent a positive reward each time it could improve the quality of the heuristic. To validate our approach, we learned to improve two heuristics : First-Fit and Best-Fit. the simulation results showed that we could reach an improvement rate of 35% for First-Fit and 20% for Best-Fit.

3.5 Conclusion

Virtual network embedding in 5G and beyond 5G networks is known to be an NP-hard problem. In this chapter, we presented two learning-based solutions for the VNEP that overcome some of the limitations of existing methods. In a first solution, we proposed an approach that combines deep reinforcement learning with a new neural architecture based on graph convolutional neural networks to allow the DRL agent to automatically take into account the structure data of the problem. The proposed solution has proven to be very effective compared to existing conventional approaches.

In the second solution, we trained a reinforcement learning agent to reduce the optimality gap of heuristics dedicated to solving this problem instead of learning a solution from scratch. We modeled the task of improving the quality of heuristics as a sequential Markov decision process, in which heterogeneous graphs are used to represent states. To automatically extract the features from the heterograph information, we exploited RGCN to parameterize the policy network. In addition, in order to guide the agent to achieve the desired behavior, we define a reward function, each time it can improve the quality of the heuristic, it will give the agent a positive reward. To assess the performance of our approach, we learned to improve two heuristics : First-Fit and Best-Fit. The simulation results showed that we could reach high rates of improvement for both heuristics.

In the future, for the first solution, we suggest evaluating the approach in more constrained conditions and take into consideration the QoS parameters as well as the

placement of services in multiple domains. For the second one, we are planning to learn to improve other heuristics and under different settings.

Chapitre 4

AI-BASED SLICES MONITORING

4.1 Introduction

The placement of the slices is not the only technical challenge network slicing raises. Indeed, once network slices are leased, network slice providers have to ensure that Service Level Agreements (SLAs) negotiated with the slices tenants are not violated. Moreover, with the easy access to network performance measurement tools such as FAST [48] and SpeedTest [112] offered by Netflix and M-Lab [74], which Google offers, internet users can now measure and make sure that the quality of the services they are subscribing to is met. Therefore, monitoring the state of the slices and verifying their performance will be a priority for both network slices providers and tenants, especially with the increase of the number of slices deployed on top of physical networks. Besides, monitoring the state of the slices is essential for various operations such as resource allocation and fault diagnosis. Notwithstanding, it is usually prohibitive and not feasible to directly measure the performance of all the slices due to the overhead caused by the measurement traffic or the lack of support at internal network elements to make these measurements. To deal with these limitations, a promising solution is to use Network Tomography (NT).

NT aims to infer the network measurements based on limited end-to-end direct measurements. Hence, it reduces the complexity of the monitoring and the network diagnostic process. Another motivation for using NT approaches for the slices monitoring is their perfect fit with the SDN and NFV paradigms which are one of the key enablers of network slicing. Indeed, NT methods require a holistic view of the state of the network, which is guaranteed through SDN technologies.

Most existing works in Network Tomography had put the accent on inferring all links metrics [68] or a subset of links in a network [27] from end-to-end measurements. In this chapter and as in [123], we focus on inferring metrics for a set of paths of interest (i.e., slices).

The metrics we focus on can be classified as either additive or non-additive. In this

work, we focus on additive metrics, but our methods can be extended easily to non-additive metrics. An example of additive metrics is the delay on slices or the logarithm of the loss rates. In general, the construction of an NT solution is done in three main steps : the selection of nodes for traffic data collection (i.e., monitors), the deployment of a probing strategy, and the inference of metrics.

In this chapter, we focus on two main challenges : First, on the inference of slices metrics based on some end-to-end measurements between monitors as well as on the efficient monitor placement.

— For the inference, we focus on it because it is a key part of building an NT-based monitoring system. It enables the quantification of the quality of the placement of monitors as well as the probing strategy. We model the inference task as a multi-output regression problem which we solve using neural networks. We propose to train on synthetic data to augment the diversity of the training data and avoid the overfitting issue. Moreover, to deal with the changes that may occur either on the slices we monitor or the topology on top of which they are placed, we use transfer learning techniques.

— Next, we consider the monitor's placement problem. We consider a special case where only cycles probes are allowed. The probing cycle schemes have a significant advantage compared to regular paths since the source probe is actually the destination which reduces the synchronization problems. The objective is to find the minimum number of monitors and their placement such that we cover all the slices by probing cycles of limited length. We formulate the problem as a variant of the minimum set cover problem. Owing to its complexity, we introduce a standalone solution based on Graph Neural Networks (GNNs) and genetic algorithms to find a trade-off between the quality of monitors placement and the cost to achieve it.

The remainder of this chapter is organised as follows : In section 4.2, we will present our first contribution relatd to the inferring part. Then, in section 4.3 we will show the details of our approach in solving the monitor placement challenge. Finally, in section 4.4 we will conclude this chapter.

4.2 On the use of machine learning and network tomography for network slicing

4.2.1 Overview

Monitoring the state of the slices and getting an accurate knowledge of their health is vital for network operators to ensure their correct functioning and that their SLAs are met. However, directly monitoring all the slices deployed on top of the physical network is costly and not always feasible due to the lack of support of internal nodes. Such limitations shed light on the usefulness of indirect methods to monitor slices. Indirect monitoring methods are known in the literature as Network Tomography (NT).

Y.Vardi introduced network tomography (NT) in 1996 [114]. Since then, NT has been extensively studied. NT refers to the bunch of methods that aim to predict unmeasured network metrics using an end-to-end measurement between monitors. Hence, it avoids all the limitations of direct methods and makes the monitoring and the network diagnostic less complex.

Several existing works in Network Tomography had focused on deducing all links metrics [68] or a subset of links in a network [27] from end-to-end measurements. In this work and as in [123], we aim to infer the metrics of a set of paths of interest (i.e., slices). In fact, to infer the slices metrics, we do not necessarily need to monitor all the links involved in the slices. Moreover, we may need to monitor links that do not belong to these slices. Hence, inferring the slices' metrics directly is more efficient and uses less monitoring resources than the approaches that focus on links metrics inferring.

In this work, we focus on identifying additive metrics. Therefore, inferring the slices' metrics is equivalent to solving a system of linear equations, where the unknown variables are the slice metrics, and the known constants are the end-to-end path measures. However, in practice, this system of linear equations is non-invertible [41], [18]. In other words, in most cases, slices metrics cannot be uniquely identified.

To deal with these drawbacks, we formulate the task of solving the linear system of equations into a regression problem, which we solve using neural networks. By doing so, our approach can also be used in the case where the system of equations can be uniquely solved. Indeed, we can determine the values in an efficient way. We avoid the matrix inversion process that takes too much time in large networks. The use of Neural Networks (NNs) is also motivated by the fact that NNs can approximate any function as long as

the hidden layers contain sufficient numbers of hidden neurons [46].

Unfortunately, formulating the problem as a regression problem and the use of neural networks to infer slices metrics raise two main challenges. First, the training process requires a large and diverse amount of real labeled data to avoid over-fitting. Hence, a real large labeled dataset of metrics must be collected in a small window of time. This will introduce an important overhead. The second issue is that with a learning-based approach if a change occurs in the network topology or the slices we monitor, the training of the neural network architecture must start from scratch to take into consideration the new changes. Repetitive training will then also introduce an important overhead. In response to the first concern, we train our neural network using simulated data [92]. While for the second one, we unveil the potential of transfer learning techniques. [131]

Overall the main contributions of this section can be summarized as follows :

— We propose a one-step solution to infer the slices' metrics using end-to-end measurements. We formulate it as a regression problem, which we solve using a neural network architecture.

— Since neural network-based methods requires a vast and diverse amount of data. To avoid the generation of overhead in the collection process, we propose a self-training approach to solve the regression problem mentioned above.

— We avoid frequent retraining from scratch when a change occurs in the network topology or in the monitored slices using transfer learning techniques. We retrain only a part of the neural network architecture instead of updating all the weights. Hence we reduce the complexity of the approach.

— We conduct extensive simulations, and we compare our approach against benchmark solutions. The results show the superiority of our approach in terms of error approximation of slices' metrics as well as in terms of the computational complexity.

The remainder of this section is structured as follows. In the subsection.4.2.2, we present the related work on the inference part in network tomography monitoring. Then, we provide an overview of the context of our work and the notations we have used in the subsection 4.2.3 . Next, in subsection 4.2.4, we delve into the details of our proposed method. Later, in 4.2.5, we describe both the simulation setup and the results analysis. Finally, in Sec 4.2.6, we conclude and summarize this part.

4.2.2 Related Work

Network slices monitoring can be defined as a particular case of overlay network monitoring [17]. Several works focused on the use of network tomography techniques to monitor overlay networks. For example, in [17], authors proposed an algebraic solution to select the necessary paths in an overlay network to fully describe an additive metric on the rest of the available paths. Moreover, M. Demirci et al. focused on the placement of overlay networks for diagnosability of the underlay network [25]. Compared to these works, our work has a different scope. We aim to identify the state of the slices and not use the slices to diagnose the underlay network.

When the target is additive metrics, the approaches suggested in the literature to infer link metrics can be used to infer the metrics of the slices. To infer the slice metrics from end-to-end measurements, we can infer the metric related to each link in the slice or all links in the underlay network and then sum them up.

Several works based on statistical methods focused on inferring links' metrics. In [64], the authors presented a solution based on multicast probing and the expectation-maximization algorithm (EM) to estimate the loss rate on links. Moreover, in [59], authors proposed *Flexicast*, a method based on a mix of unicast and multicast probing to infer links' delays in a tree topology. They formulate the problem as a likelihood maximization problem, which was solved using the EM algorithm. This kind of approach is characterized by its slow convergence time. In [93], there is a statistical algorithm called ESA merging the EM technique with an evolutionary approach that outperforms previously mentioned proposals.

More recently, machine learning-based solutions have been introduced for network monitoring. Sirnivasan and al used supervised learning, in particular a mix of classification and regression, to locate the failed links [109]. Compared to our work, we also use supervised learning, but we train a model using an automatically generated dataset. In [92], M.Rahali and al, used machine learning algorithms and, more particularly, neural networks. They trained them using a simulated training dataset. However, their focus was on the inference of underlying metrics using information collected in the overlay networks. Suppose a change occurs in the underlay topology or the overlay networks. In that case, their approach has to repeat the training from the beginning, introducing an important overhead in the monitoring process.

4.2.3 Context and problem formulation

As mentioned earlier, in this work, we aim to monitor a set of network slices placed on a physical network using NT and machine learning techniques.

In general, the construction of an NT solution is done in three main steps : the selection of nodes for traffic data collection (i.e., monitors), the deployment of a probing strategy, and the inference of metrics.

In this work, we focus on the inference part. It is a key part because it is the one that quantifies the quality of the first and the second steps. Hence, we consider that the placement of monitors and the list of probed paths as known. Then, we collect the monitoring data and feed it to our model to estimate the slices' metrics.

The notations used in this subsection are summarized in Table 4.1.

Notation	Description
$\mathcal{G} = (\mathbb{V}, \mathbb{L})$	The physical topology
$\mathbb{V}, \mathbb{L}$	Set of physical nodes and links
$\mathbb{P}$	Set of measurement paths
$\mathbb{S}$	Set of slices to monitor
Y_s	Unknown slices metrics vector l_s
Y_p	measurements paths metrics vector
X	Unknown links' metrics vector

TABLE 4.1 – Main notations

We assume that the topology of the physical network is known and we model it as an undirected graph $\mathcal{G} = (\mathbb{V}, \mathbb{L})$ where $\mathbb{V}$ and $\mathbb{L}$ represent the sets of physical nodes and of the physical links, respectively. Let $\mathbb{P}$ denote the set of the probed paths and $\mathbb{S}$ the set of slices we want to monitor and which are deployed on top of the physical network $\mathcal{G}$. The cardinality of the sets $\mathbb{V}$, $\mathbb{L}$, $\mathbb{P}$ and $\mathbb{S}$ are respectively, $m = |\mathbb{V}|$, $n = |\mathbb{L}|$, $p = |\mathbb{P}|$ and $s = |\mathbb{S}|$. Let X denote the vector of size n, the unknown links' metrics. Similarly, we denote by Y_s the vector of the unknown slices metrics, with size s. Finally, we define Y_p as the vector of the measured or observed probed paths metrics, whose size is p.

To model the relation between the physical links and the slices being monitored, as well as the relationship between the physical links and the measured paths, we define two matrices A_s and A_p. These are Boolean matrices, and their shapes are respectively equal to (s, n) and (p, n). For A_s the entry $A_s(i, j)$ is equal to 1 if the jth physical link belongs to the ith slice, and to 0 otherwise. Likewise, the $A_p(k, l)$ equals one if the lth link is on

the kth path, 0 otherwise. Finally, we assume that the links' metrics have a known upper bound B.

Using the notation stated before, we derive two main equations :

$$A_s X = Y_s, \tag{4.1}$$

$$A_p X = Y_p. \tag{4.2}$$

The objective here is to infer the Y_s vector knowing A_s, A_p and Y_p. To solve this problem, one may start by determining X solving Equation (4.2) and then, a simple multiplication with A_s will give Y_s. However, in practice, the linear system depicted in Equation (4.2) is undetermined (that is, $p < m$). Moreover, even in determined cases we have another issue here. Solving Equation (4.2) involves inverting matrix A_p. This takes $O(n^3)$ time, which can be too much for large graphs.

4.2.4 Neural network-based tomography for slices monitoring

Inferring slices' metrics using neural networks

To solve the problems discussed in the previous section, we model the metrics' inference as a multi-output regression problem, where the features are represented by Y_p and the targets by Y_s. Thus, solving the equation is equivalent to learning the f function defined by

$$Y_s = f(Y_p). \tag{4.3}$$

To learn f, we use neural networks (NNs). NNs can approximate any function as long as the hidden layers contain sufficient numbers of hidden neurons [46].

By doing so, our solution can approximate the slices metrics in the case of an undermined system. Moreover, the direct learning of the relation between Y_s and Y_p makes the task independent of the size of the physical network. Hence it can scale easily.

The neural network architecture we used is a Multi Layer Perceptron (MLP)[96]. As presented in Fig 4.1, it consists of one input layer with p neurons, one output layer with s neurons and h hidden layers. The number of neurons in each hidden layer is a hyperparameter that we fine-tune to reach an optimal performance. To train this neural network architecture, a set of pairs (Y_s, Y_p) must be collected. Since we know the topology of the physical network, we can simulate the distribution of traffic in the network in many

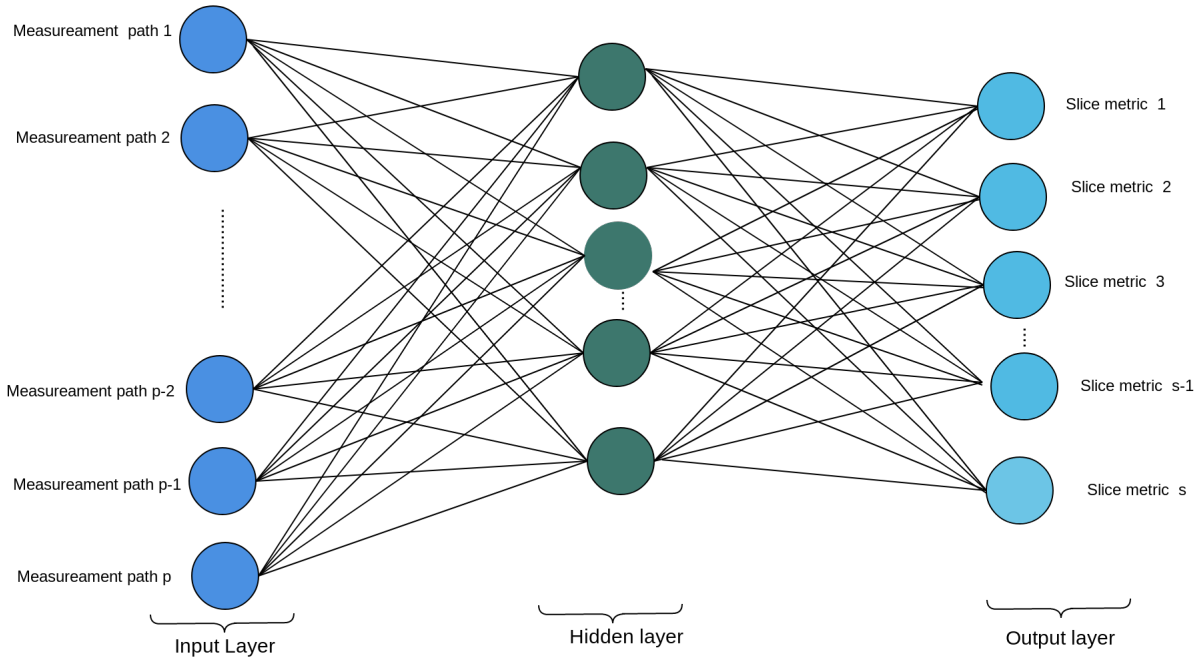


FIGURE 4.1 – Neural network to infer slices metrics

configurations and observe the pairs (Y_s, Y_p).

Neural networks are efficient once trained. However, as the case for any data-driven approach, it requires a huge and diverse pool of samples and, therefore, a large amount of labeled data. Moreover, to avoid the over-fitting problem [125], this process must be done under diverse network conditions. Unfortunately, the latter process will generate an overhead and may disturb the network's functioning. To solve such a challenge, we propose to train our neural network using simulated data. We propose to generate a large amount of random pairs $(Y_p^{'}, Y_s^{'})$.

The dataset is generated as follows : we generate a large number of random samples of links metrics denoted as $X^{'}$. We associate with each link metric a random value between 0 and an upper bound B to generate one sample of $X^{'}$. Then, given A_s and A_p and based on Equations (4.1) and (4.2), we generate one pair $(Y_p^{'}, Y_s^{'})$.

After collecting the simulated dataset, we feed it to the neural network and train the latter using the backpropagation algorithm in order to minimize the mean square error (MSE) between the predicted and the target values [102].

Once the training of the neural network is finished, we use it to estimate the slice

metrics from paths' measurements.

Transfer learning

In real scenarios, the number of slices to monitor may vary with time. In addition, the paths' measurements can also change, either due to a failure on the physical network or to an update of the monitors in charge of launching the probes. When these changes occur, and in order to take them into consideration, we have to update the neural network architecture and start the training process from scratch. However, changing the neural network architecture and restart training from scratch each time a new slice is added to the set of slices to monitor may be inefficient and may introduce an important overhead in the monitoring process. To deal with this challenge, we unveil the potential of transfer learning.

Transfer learning is a machine learning technique that allows to *transfer* what a model learned in one task to another one. Thus, when a change occurs, instead of building a new neural network (NN) and start the training process from the beginning, we take the last version of our model, and we modify only the input or the output layer. If the update concerns the slices we monitor, it is the output layer that will be changed. In contrast, it will be the input layer that will be changed if the update occurs in the measurements' paths. Moreover, once we make this slight update, we train the new version of the model efficiently. We only update the weights of the new layer and freeze the other ones. This way, the model transfers what it learned before and then starts to learn how to adapt to the new changes.

4.2.5 Performance evaluation

In this part, we evaluate the performance of our monitoring approach. To evaluate its effectiveness, we compare it to two main solutions. In what follows, we present a description of the simulation setup, then we define the approaches against which we compare our solution. Finally, we present and discuss the results.

Simulation setup

In order to test the performance of our method, we tested it using two different topologies. The first one was taken from [76] and the second one from [54]. They are depicted,

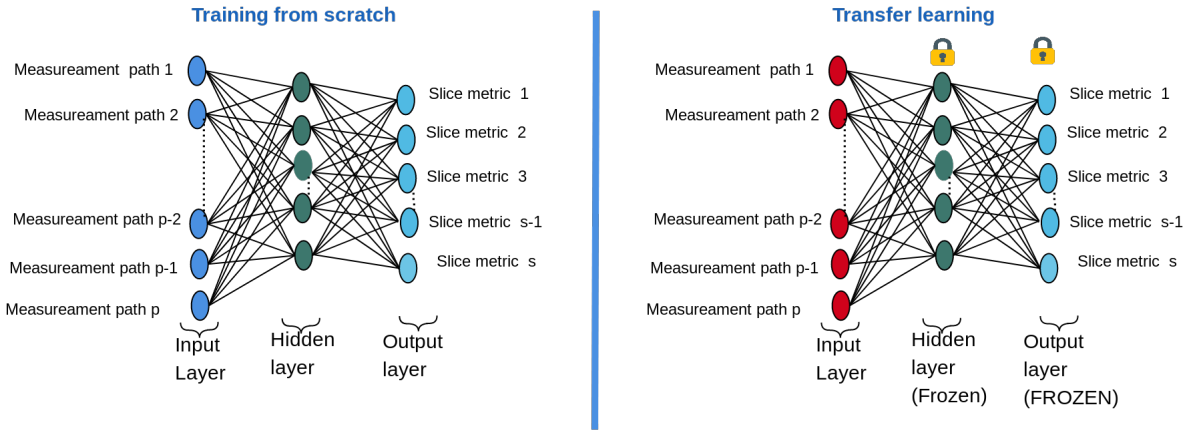


FIGURE 4.2 – Transfer learning paradigm

respectively, in Fig. 4.3 and Fig. 4.4. The first topology contains 9 nodes and 22 links, while the second one contains 54 nodes and 68 links.

For each network, two nodes are selected to start probes and exchange the monitoring traffic. Besides, we considered a predefined list of measurement paths, as well as the slices to monitor. The first topology is provided with a dataset of multiple samples of delay measurements performed on its different links. To get this dataset, the traffic was simulated with the Omnet++4 network emulator [115]. Regarding the second topology, we generate a random dataset of link delays. Based on these datasets, we computed the end-to-end delay on each path and on each monitored slice according to the equations (4.1) and (4.2). For the first topology, we monitor 3 slices, while for the second topology, we monitor 20 slices. The main parameters of our simulation setup are summarized in Table 4.2.

In order to assess the quality of the estimates, we define, in the following, the error formula :

$$Error = 100 \times |Y_s^{estimated} - Y_s^{real}|. \tag{4.4}$$

In this equation, $Y_s^{estimated}$ represents the estimated delay on the slices, while Y_s^{real} is the exact value we measured.

Baselines

In order to benchmark the performance of our proposed method, we compare it against two well-known state-of-the-art solutions. They both aim to estimate the links' metrics

Topology	m	n	p	s	Test set
Topology 1	9	22	$[16, 20]$	3	Emulation
Topology 2	54	68	$[20, 60]$	20	Simulation

TABLE 4.2 – Topologies specifications

and then use them to estimate the slices' metrics.

They can be described as follows :

— **Singular Value Decomposition-based method** : This method relies on the singular value decomposition method (SVD) [35] to compute the pseudo-inverse of the matrix A_p, denoted A_p^+. Then, the slices metrics are obtained by the following equation :

$$Y_s^{estimated} = A_s \, A_p^+ \, Y_p \tag{4.5}$$

— **Link level metric prediction** : In this method, we apply the learning method we propose but to estimate the link metrics. Therefore, in this case, we define a neural network whose input is the path metrics and the output is the estimate of the link metrics. Once the training is complete, we replace the estimation of the link metrics in Equation (3.6) to get $Y_s^{estimated}$.

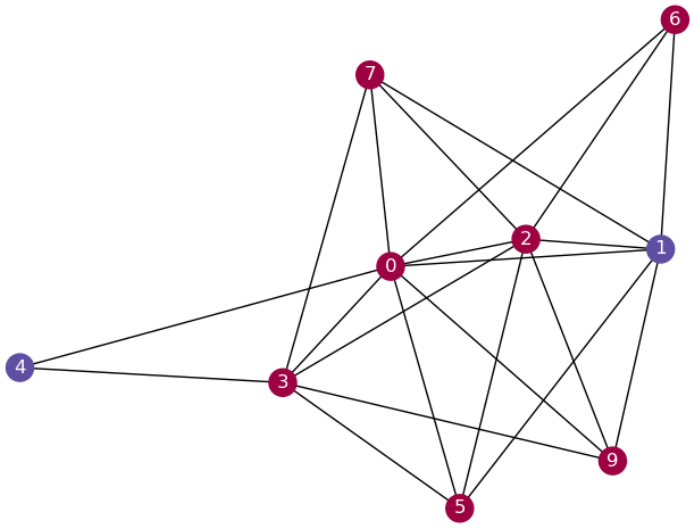


FIGURE 4.3 – Topology 1.

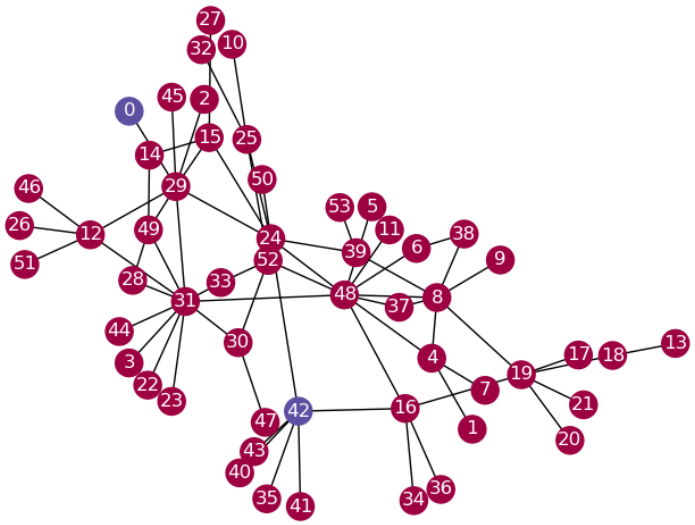


FIGURE 4.4 – Topology 2.

Simulation Results

To compare the performance of our method with the above described approaches, we consider a neural network with only one hidden layer. In each case, we generate a simulated training set with 5×10^5 samples. The hidden layer contains 50 hidden units. These parameters were fixed after several tests when we saw that after increasing the number of hidden units, we only increase the computation time while the performance is quasi the same. Moreover, the training lasts for five epochs.

Estimation error vs number of paths used

With each topology, and based on the monitor placement, we fix a list of measurements' paths. For the first topology, we fixed 20 paths, while we considered 60 paths for the second one.

To test the relation between the number of measurements paths used and the accuracy of the estimation, we made multiple tests. In each test, we select k paths from the predefined list, and we compare the performance of our approach to the SVD and link-based estimations as a function of the number of paths used.

For example, for the first topology, first, we use only 16 paths among the 20, then 16, and so on, in order to analyze the effect of the number of paths on the estimation error.

Figures 4.5 and 4.6 illustrate the dispersion of the percentage error of the three methods with both topologies.

Increasing the number of measurements paths reduces the percentage error of the solutions. Our method always has a better estimation accuracy. With the first topology and while using 20 paths, the median of the absolute error is 3.45% with our method, while it was 3.71% for the link-based method and 10.76% for the SVD based method. With the second topology, when we use, for example, 20 paths, the median error is 4.34% for our approach, while it is 7.13% for the link based method and 12.07% for SVD.

Overall, we find that learning-based solutions have taken the lead from the SVD method. In addition, direct estimation of slice metrics is more precise and efficient. In fact, by dividing the task into link estimation and then using that estimation to obtain slice metrics, the error of the first step is propagated. Therefore, the performance is lower than the direct method. With our approach, the neural network focuses directly on learning the solution that minimizes the error.

Finally, when using the first topology, we observe that the difference between our method and the one that learns to estimate link metrics is narrow. However, in terms of temporal complexity, our solution is more efficient since we only predict for three slices, while the other approach focuses on 22 links.

Transfer learning evaluation

In this section, we study the performance of the transfer learning methodology we used. Without loss of generality, in these tests, we highlight results related to the second topology. We start with a case where we monitor only ten slices. Once the training is done, we suppose that new slices are added and need to be monitored. We added 5, 10 and 15 new slices to the existing 10 slices.

For each case, we used the model trained to monitor the first 10 slices. Then we freeze all its layers except the last one ; we change it with a new layer with 15 slices, for example, for the first case. Then, we compare the performance of these models with the ones that are retrained from scratch and do not use any weights from the original model, which was in charge of monitoring the first 10 slices.

Figures 4.7 and 4.8 illustrate the comparison in terms of the median error of the estimation and the time to finish one epoch of training after a change occurs. We observe that pre-trained models are less accurate than the whole trained ones. However, this difference is tight. Besides, when it comes to the training time to get these models, we observe that pre-trained models are three times faster than the other models.

To summarize, the use of transfer learning allows us to find a trade-off between the

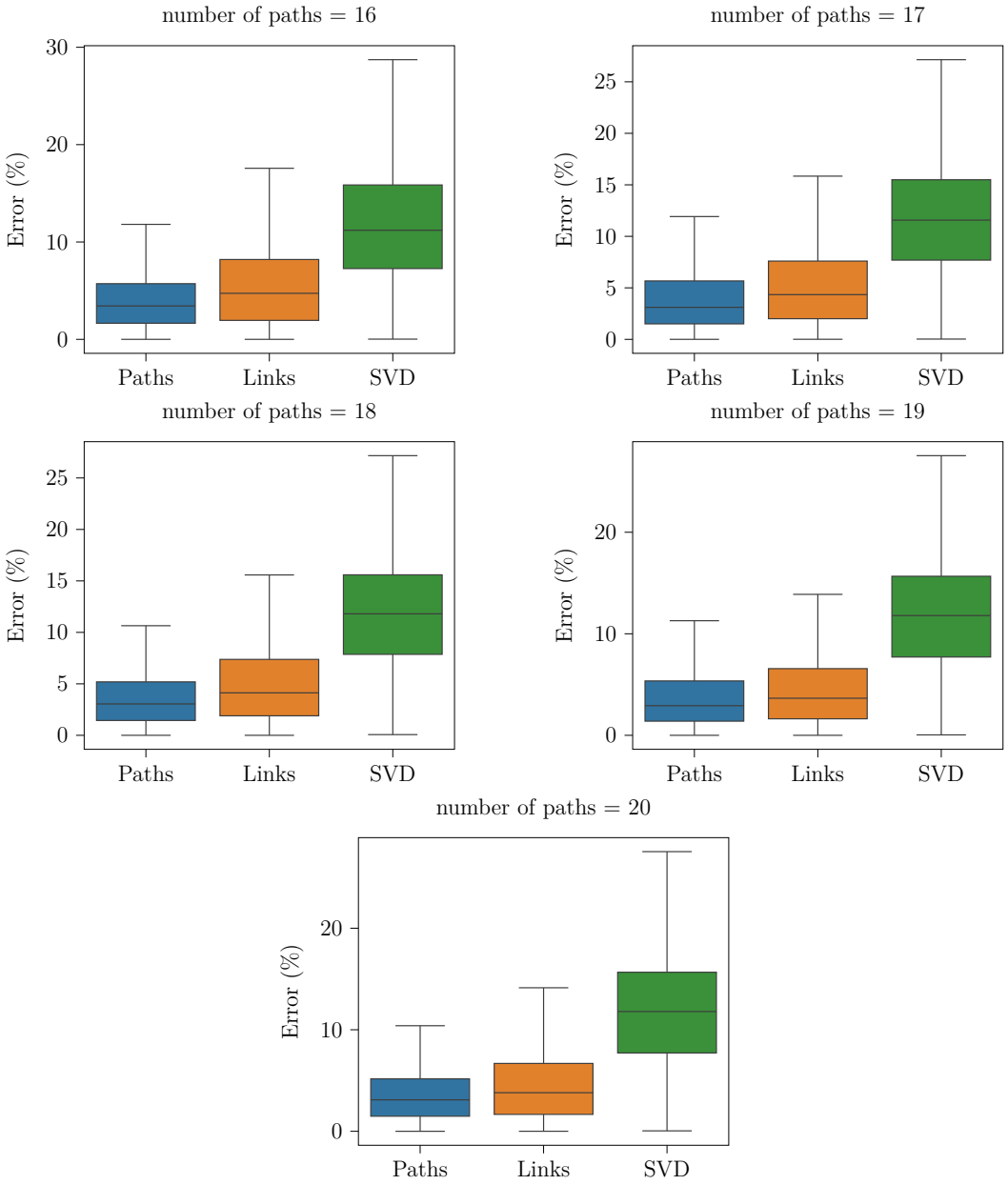


FIGURE 4.5 – Error vs # of measurements paths for **Topology 1**

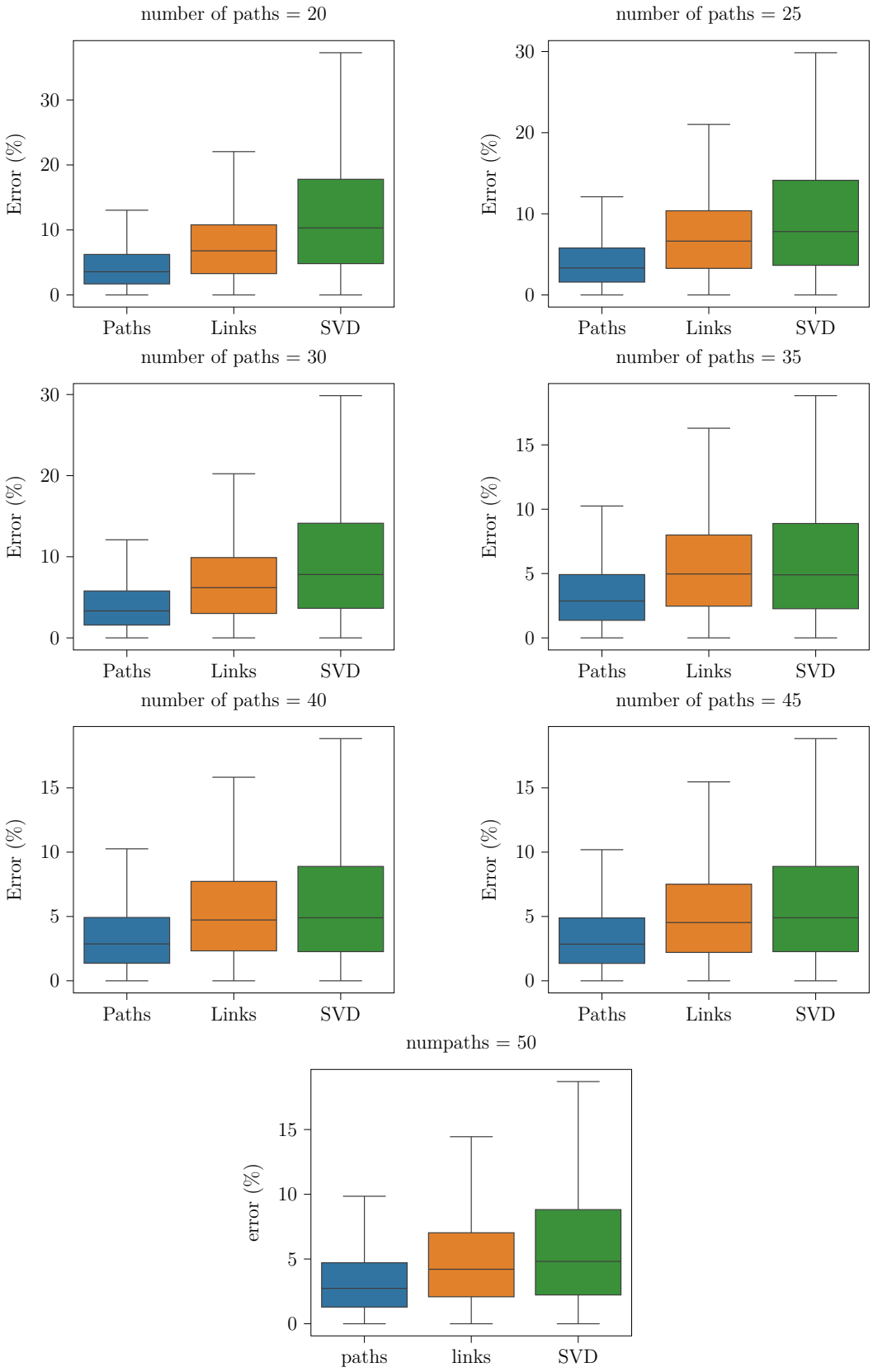


FIGURE 4.6 – Error VS # of measurements paths for **Topology 2** :

quality and the complexity of the monitoring solution. It transfers the knowledge acquired during past experiences to get a fast adaptation to the new changes.

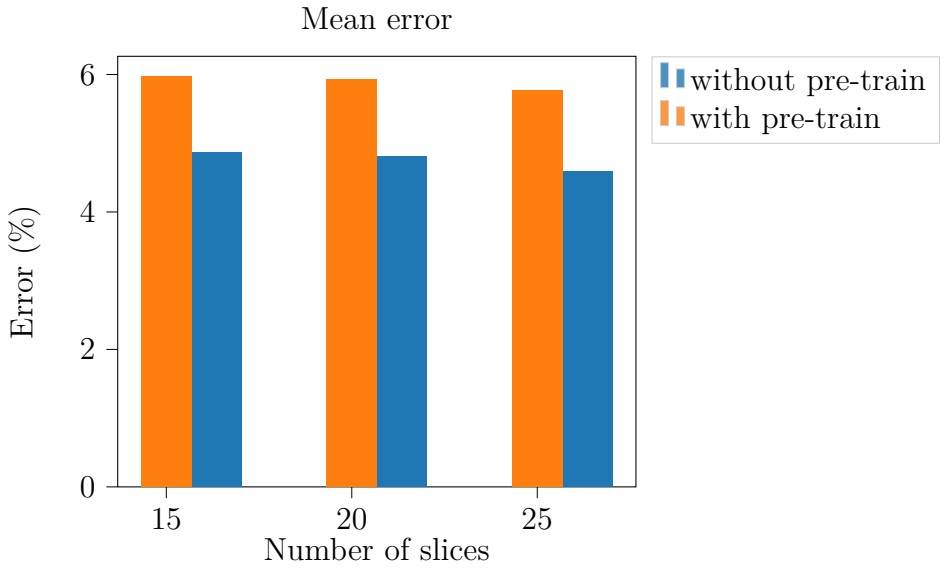


FIGURE 4.7 – Topology 2. Error : Pre training vs Retraining from scratch

4.2.6 Summary

In this section, we have shed light on the potential of machine learning methods and network tomography techniques for network slices monitoring. We focused on additive metrics inference, but the approach can be used for the non-additive metrics as well. Unlike existing work, we focused on estimating the slices metrics and not the metrics of the physical links. Hence, we modeled the inference problem as a regression problem, and we solved it by training neural networks. To deal with the challenges associated with the training of neural networks, we used simulated data in the training step. Finally, we proposed a transfer learning technique in order to handle changes in the physical topology as well as the slices subject of monitoring. We evaluated the performance of our approach using an emulated and simulated network traffic. The results show that our approach outperforms the methods with which we compared in terms of both accuracy and computational complexity. For future work, we plan to extend this work by incorporating the first and second stages of network tomography monitoring. Namely, an intelligent selection of measurements.' paths as well as an intelligent monitor placement for slices' monitoring. Furthermore, we plan to use neural network architectures suitable for graph

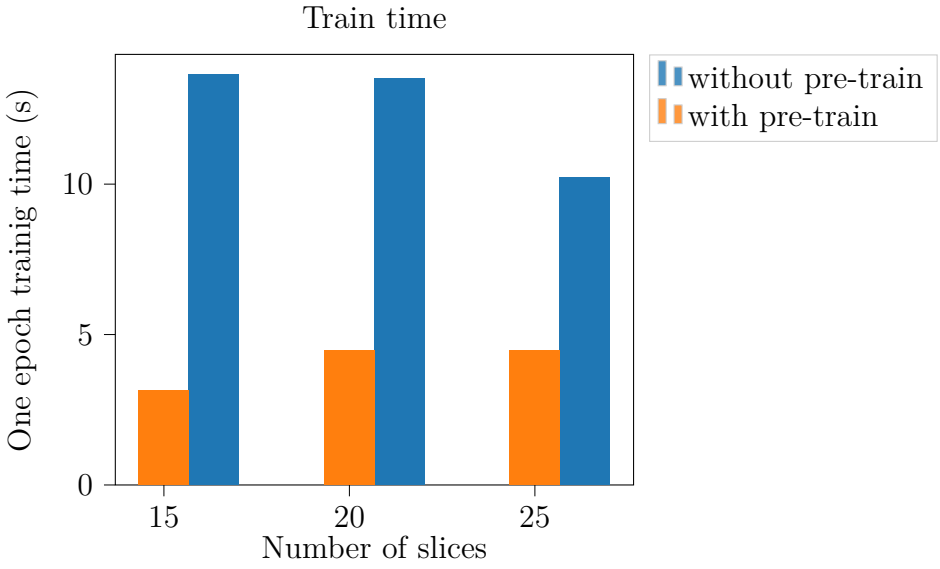


FIGURE 4.8 – Topology 2. One Epoch time : Pretrain vs retraining from scratch

structured data. The aim is to generalize for new topologies.

4.3 MonGNN : A neuroevolutionary-based solution for 5G network slices' monitoring

In the previous section, we demonstrated how the combination of network tomography and machine learning led to an efficient solution to monitor the health of network slices. The focus was put mainly on the inference part. We assumed that the probes to collect the end-to-end measurements follow a regular path. Hence to collect the end-to end metrics following a certain path, both the sender and the receiver need to synchronize. In order to reduce the complexity of the monitoring process and avoid all synchronization related issues, we propose to monitor the network slices using a cycle probing scheme. In other words the path followed by the monitoring traffic will be defined as a cycle and the monitor will play the roles of the sender and the receiver in a probe.

To implement a cycle-based monitoring scheme, the network needs to forward packets along arbitrary cycles. Segment routing (SR) is one of the solutions that allows to constrain traffic to follow a well specified path (or cycle). SR is based on the concept of source routing. The main idea of SR is that the traffic source encodes the forwarding path, or some parts of the path, in the packet header. The path is encoded in the packet header as

a list of segment IDs (SID). Then, the packet is forwarded according to the instructions corresponding to the stacked SIDs.

SR is a new mechanism of routing, it brings new advantages such as flexibility and scalability. By stacking multiple segment labels in the packet header at the network boundary, segment routing can flexibly control the packet transmission along a specific path or specific nodes according to administrative demand. Moreover, since in SR the route information is stored in the labels and determined at the boundary of the network, the per-flow state information is only needed at the edge routers of the network, and the memory consumption for forwarding entries at the core nodes can be significantly reduced. Thus, the segment routing can be applied in a larger network

Despite the flexibility offered by segment routing in monitoring, this technology presents many challenges. Indeed, most commercial switches only support a limited number of SIDs in packet headers [39]. This adds a constraint on the length of the probed cycles.

In this section we focus on finding the optimal number of monitors and their placement such that we cover all the slices by probing cycles of limited length. The task becomes then equivalent to the minimum set covering problem, which is known as an NP-Complete problem [13].

Several solutions have been proposed to solve the problem of monitors' placement, which can be classified into two classes : exact and heuristics. Exact methods ensure optimality, but they are applicable only for small networks instances. On the other hand, heuristics can be applied to large instances, but they are suboptimal.

To find a trade-off between the quality of the solution and the cost to obtain it, we propose MonGNN, an autonomic solution based on Graph Neural Networks (GNN), to learn to predict the optimal number of monitors and then use genetic algorithms to reveal their identity.

Overall, the main contributions of this section can be summarized as follows :

— We propose an Integer Linear Programming (ILP) formulation of the following problem : given a set of slices running some services, determine the optimal placement of monitors in order to cover all these slices using cycles whose length is less than a certain amount k.

— We propose a learning-based approach to find a trade-off between the quality and the cost of the solution.

— We unveil the potential of Graph Neural Networks to predict the optimal number of monitors.

— We define a Genetic Algorithm (GA) to identify the monitors based on the predicted value of the objective function.

— We enhance the GA by a local search method to deal with the estimation error that can occur in predicting the number of monitors.

— We conduct extensive simulations, and we compare our approach against benchmark solutions. The results show the superiority of our technique in terms of optimality gap as well as computational complexity.

The remainder of this section is organized as follows. We introduce background information in the subsection 4.3.1. Then, a related work on monitors' placement is presented in subsection 4.3.2. Next, in subsection 4.3.3, we provide an overview of the context of the work and the notation we have used as well a formal definition of the problem we aim to solve. Then, in 4.3.5, we delve into the details of our proposed solution. Next, in subsection 4.3.6, we describe both the simulation setup and the results' analysis. Finally, in subsection 4.3.7, we conclude and give an overview of our future work.

4.3.1 Background

Graph Neural Networks (GNN)

Graph Neural Networks (GNNs), are a type of neural networks dedicated to graph structured data. They were introduced in [103] and several variants of it have been proposed [120]. Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs) are among the most known variants.

Let $\mathcal{G} = (V, E)$ be a graph, where V is the set of nodes and E the set of edges. We assume that each node $n \in V$ is characterized by an input feature vector in_n.

GATs [116] have been proposed as an improvement of GCNs. Instead of doing a normalized sum of the features of the neighbors, GATs include the attention mechanism in the propagation step to generate the nodes' encoding, which is computed using the following equations :

$$out_n = f\Big(\sum_{m \in \mathcal{N}(n)} \alpha_{nm} in_m W^l\Big), \tag{4.6}$$

$$\alpha_{nm} = \frac{\exp\Big(\text{LeakyReLU}\Big(a^T\Big[in_m W^l || in_n W^l\Big]\Big)\Big)}{\sum_{j \in \mathcal{N}(n)} \exp\left(\text{LeakyReLU}\left(a^T\left[in_n W^l || in_j W^l\right]\right)\right)}, \tag{4.7}$$

where || denotes the concatenation operator, a is a fully connected neural network and $.^T$ the transposition. $\mathcal{N}(n)$ represents the set composed of the node n and its neighbours

To stabilize the training process, the authors of GAT proposed to use multi head attention. It applies H independent attention head matrices to compute the hidden states and then concatenates their features. With the multi-head attention mechanism the node encoding becomes governed by the following equations :

$$out_n = \|_{h=1}^{h=H} f\Big(\sum_{m \in \mathcal{N}(n)} \alpha_{nm}^k in_m W^l\Big), \tag{4.8}$$

$$\alpha_{nm}^h = \frac{\exp\Big(\text{LeakyReLU}\Big(a_h^T\Big[in_m W^l || in_n W^l\Big]\Big)\Big)}{\sum_{j \in \mathcal{N}(n)} \exp\left(\text{LeakyReLU}\left(a_h^T\left[in_n W^l || in_j W^l\right]\right)\right)}. \tag{4.9}$$

Segment routing

In segment routing, each probing cycle is encoded by a list of segment IDs (SID). There are two types of SIDs : nodes-SIDs and adjacency-SIDs. The former identify a shortest-path while the latter identify a link segment. The use of nodes-SIDs is feasible ; however it may lead to an ambiguous definition of the probing cycle and accordingly introduce a bias in the measurement process [95]. In fact, when using nodes-SIDs to identify a path between a source s and a destination d and due to the load balancing mechanism deployed, we will not be sure which path the traffic followed exactly.

In this work, we consider the adjacency-SIDs and we assume that each probing cycle is encoded only using a list of adjacency-SIDs of a maximum length of k.

4.3.2 Related work

Segment routing (SR) is considered as a promising candidate for monitoring network slices. It provides, indeed, fast and efficient tools to deploy traffic forwarding strategies along arbitrary paths without any additional protocol or signaling procedures. It enables to constrain traffic to pass through particular places and, thus, build specific probing schemes. Besides, SDNs allow arbitrary routing, but SR is more scalable and practical [23]. In SR, the path is encoded in the packet header as a list of segment IDs (SIDs). Then, the packet is forwarded according to the instructions corresponding to the stacked SIDs.

Thanks to the advantages it brings, SR was proposed as a key tool for network monitoring in several works. In [9], the authors introduce an SR-based monitoring solution where only a single monitor is deployed, and monitoring probes follow the cycles starting and ending at that monitor. They propose *Scmon*, a framework that minimizes the number of cycles covering the physical topology and efficiently encodes the probing cycles with segment IDs. The use of a single monitor reduces the monitoring cost and avoids synchronization issues. However, this requires the use of very long cycles to cover all the resources, which can introduce a bias in the measurements. Similarly, X.Li and al. [62], focused on using a single monitor and proposed an improvement of the Scmon framework. They emphasize minimizing the total length of all probing cycles that can be generated from a single monitor. Furthermore, the authors of [61] proposed several ILP formulations of the minimum cycles cover problem. The common point between our proposal and these works is the use of segment routing and cycle probing. However, what distinguishes us is that we focus on finding a subset of nodes as monitors and not a single monitor.

In [38], the authors theoretically studied the problem of monitor placement to identify additive links metrics. They gave necessary and sufficient conditions to identify an additive metric on all the network links for a given topology. Then, they proposed a solution that minimizes the number of monitors to exchange probes with regular paths or cycles. However, no constraint on the length of the path is considered. Similarly, in [44], authors built necessary and sufficient conditions on identifying additive metrics but using only regular paths. They followed an algebraic approach, and they proposed an algorithm based on the graph decomposition into strongly connected components to identify the minimum number of monitors that guarantee the identifiability of all links. In another work [71] the authors focused on a different vision of the problem of monitoring placement, where the number of monitors is fixed first. Then they aim to find their optimal placement to identify as many links as possible.

More recently, M.Rahali and al. [95] focused on the same problem we consider here. They propose a Greedy solution to which we compare our learning-based method.

4.3.3 Problem Formulation

In this work, we focus on the optimal monitor placement problem, but under certain constraints : the probing scheme will consist of cycles that we deploy using segment routing. The use of probing cycles eliminates the synchronization problem that one might have when using regular monitoring paths. It should be noted, however, that the use

of segment routing adds a constraint to the length of the probing cycles. The notation used in this subsection are summarized in Table 4.3. We assume that the topology of the

Notation	Description
$\mathcal{G} = (\mathbb{V}, \mathbb{L})$	The physical topology
$\mathbb{V}$	Set of physical nodes
$\mathbb{L}$	Set of physical links
$\mathbb{C}$	Set of measurement cycles
$\mathbb{S}$	Set of slices to monitor
k	Maximum length of probing cycles
$\mathcal{C}_v^k$	Set of cycles starting at node v
$\mathcal{L}_v^k$	Links covered by $\mathcal{C}_v^k$

TABLE 4.3 – Main notations

physical network is known, and we model it as an undirected graph $\mathcal{G} = (\mathbb{V}, \mathbb{L})$, where $\mathbb{V}$ and $\mathbb{L}$ represent the sets of physical nodes and of the physical links, respectively. Let $\mathbb{C}$ denote the set of the probed cycles and $\mathbb{S}$ the set of slices we want to monitor and which are deployed on top of the physical network $\mathcal{G}$. The cardinality of the sets $\mathbb{V}, \mathbb{L}, \mathbb{S}$ and $\mathbb{C}$ are respectively : $n = |\mathbb{V}|$, $m = |\mathbb{L}|$, $p = |\mathbb{C}|$ and $s = |\mathbb{S}|$

The objective of collecting the end-to-end measurements is to detect failures and to monitor the state of the links on top of which slices are deployed. Therefore, the state of a link is defined by the fact that the monitored metric (e.g., delay, loss rate) is above or below a threshold. Hence, the metric of interest becomes non-additive (i.e., its value on a link is 1 if and only if there is an anomaly on that link).

Covering all the physical links on top of which slices are deployed will allow then the detection of any anomaly. In this work, we focus on covering all physical links, not just the subset of links hosting slices, as this will be more robust in the event of slice migration and will not necessitate recalculating the monitors' placement. From this observation, we focus on finding the minimum number of monitors to cover the maximum number of physical links using cycles.

Given the physical network, we denote the cycles formed by at most k links that can be generated from each node v as $\mathcal{C}_v^k$. Similarly, for each vertex v the set of links covered by $\mathcal{C}_v^k$ is denoted as $\mathcal{L}_k^v$.

The first step in solving the monitor placement is to compute for each node v, the set of cycles whose length is less than k : $\mathbb{C} = \{\mathcal{C}_{v_1}^k, \mathcal{C}_{v_2}^k, \cdots, \mathcal{C}_{v_n}^k\}$. Algorithm 2 summarizes this process.

Algorithm 2 Compute Cycles whose length less or equal to k [95]

Input : a graph $\mathcal{G} = (\mathbb{V}, \mathcal{L})$, maximum length of cycles k
Output : The set of cycles $\mathbb{C}$

$\mathbb{C} \leftarrow \emptyset$
for all node v in $\mathbb{V}$ **do**
    $v\_start \leftarrow v$
    $\mathcal{C}_v^k \leftarrow \emptyset$
    $visited \leftarrow \emptyset$
    ComputeCycles$(v, visited, v_start)$
    $\mathbb{C} \leftarrow \mathbb{C} \cup \mathcal{C}_v^k$
return $\mathbb{C}$
function ComputeCycles$(v, visited, vstart)$
    if $|visited| > k$ **then**
        return
    if $v = vstart$ **then**
        $\mathcal{C}_v^k.add(visited)$
        return
    for $u \in \mathcal{N}(v)$ **do**
        if $u \notin visited$ **then**
            ComputeCycles$(v, visited, v_start)$

Once the list of cycles is obtained, the next step is to determine the minimum set of monitors $\mathcal{M} \subset \mathbb{V}$ such that : $\bigcup_{v \in \mathcal{M}} \mathcal{L}_v^k = \mathbb{L}$.

In other words, the task of finding the optimal set of monitors which cover all links can be considered as a particular case of the Minimum set cover problem, which is known to be NP-Complete.

To summarize, to determine the optimal set of monitors we need to solve the following integer linear program (ILP) :

$$\begin{aligned} \min \quad & \sum_{v \in \mathbb{V}} x_v \\ s.t \quad & \sum_{v : l \in \mathcal{L}_k^v} x_v \geq 1, \qquad \forall l \in \mathbb{L} \\ & x_v \in \{0, 1\} \end{aligned}$$

x_v is a binary varibale that indicates if a node v is selected as monitor or not.

To solve the problem presented in previous section and due to its NP-completeness, we introduce MonGNN, a GNN based framework that divides the problem into two tasks : determining the number of monitors and then identifying them.

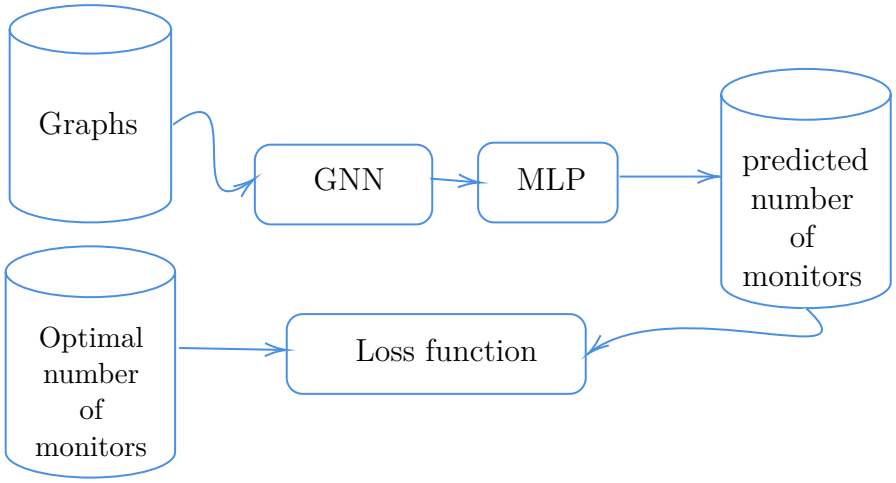


FIGURE 4.9 – The training process to learn to predict the number of monitors.

4.3.4 Learn to predict the number of monitors

The main process of the first task is depicted in Figure 4.9. The objective of this task is to learn a function f that maps a graph and a maximum cycle length to the needed number of monitors. It is defined as follows :

$$f(\mathcal{G}, k) = m_n \tag{4.10}$$

To learn this function, we model it using neural networks (NNs). In fact, NNs can approximate any function as long as the hidden layers contain sufficient numbers of hidden neurons [46].

Since f takes as input a graph, we used a Graph Neural Network (GNN). GNNs takes as input a graph and outputs a node-level representation : a matrix, where each row is a vector that encodes both the attributes and the structure of a vertex in the input graph. Then, to obtain a graph-level representation (a vector that encodes the whole graph), we choose to apply a simple average of the node-level representation. Figure 4.10 summarizes the main parts of the GNN module represented in Figure 4.9.

Next, this graph encoding is fed into a Multilayer Perceptron (MLP) [96] to obtain the number of monitors.

To train this neural architecture, we consider a supervised setting. We start by the data collection process, which is summarized below :

— We generate N graphs.

— For each graph, we compute the cycles map using Algorithm 2.

— Based on the cycles map, we generate a link map that models the relationship between the nodes and the links they can cover.

— The problem becomes a set covering problem that we solve using the ILP defined in the previous section (using Gurobi [1]) to obtain the optimal labels.

At the end of this process, we obtained a dataset of pairs (graphs, optimal number of monitors needed to cover all links). This dataset was then used for the training of the neural network architecture minimizing the mean square error (MSE) between the true labels and predicted ones.

Once the training step was done, we got a model that, given a graph, predicts the optimal number of monitors that we need to cover all links.

Apart from the fast prediction, the obtained model does not need to compute neither the cycles map nor the link cover bipartite graph in order to determine the number of monitors. It learns how to directly map the raw physical topology to that optimal number of monitors.

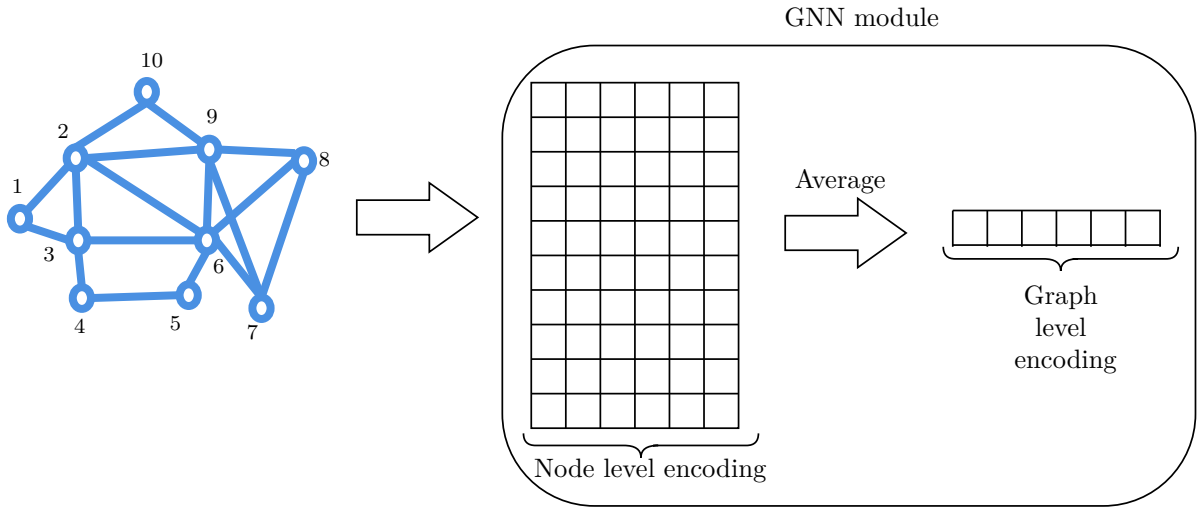


FIGURE 4.10 – GNN Module

1. https ://www.gurobi.com/

4.3.5 Monitors identification

Until now, our model is able just to predict the number of monitors we need. In this step, we aim to answer the following question : given the optimal number of monitors, how can we determine their identity ?

The answer to the question above is depicted in Figure 4.12. The GNN module refers to a trained model.

The process of revealing the identity of the optimal monitors is done as follows : we feed a graph to the trained module to obtain a prediction of the optimal number of monitors. Then we compute the bipartite link graph, and we feed them with the graph to the genetic algorithm in charge of revealing the identity of monitors. The genetic algorithm (GA) has 5 phases : initialization, fitness score, selection, cross-over, and Mutation. The general workflow of the GA is presented in Figure 4.11.

— **Initialization step** : In the initialization step, we generate a random generation of solutions (individuals). Each solution is a boolean vector whose size equals the number of nodes in the graph. The j^{th} node is selected as a monitor if and only if the j^{th} of the boolean vector equals one. The generation of this initial population is based on the predicted number of monitors. For example, let's suppose that the number of nodes in the graph is set to N and the predicted number is m ; in this case, all random vectors will be of size N and they all contain m ones and $N - m$ zeroes.

— **Selection step and fitness score** : The objective of the selection step is to select individuals (solutions) that pass their genes to the next generation. We consider a simple selection process, only keeping the best elements and killing the rest.

— **Fitness score** : To quantify the fitness of an individual, we define a fitness score which equals to the number of links covered by the selected elements of the solution.

— **Mutation and cross over :** In the cross-over, the aim is to combine the genes of two individuals in the hope of obtaining a better solution. In the context of our problem, the cross-over is done by taking two individuals and swapping a single different element between them. Finally, the Mutation is done by randomly changing the state of two nodes, while keeping the number of ones in each individual equal to n.

As we will show in the performance evaluation part, the trained model may underestimate or overestimate the optimal number of monitors. In spite of this misestimation, the

prediction remains close to the optimal solution because it will add one node at most to the number of monitors. To handle this error, we propose a local search approach to enhance the process of determination of the identity of the optimal monitors. The local search process checks if the solution with n nodes cover all links. If it does, it will try to find a solution with $n-1$; otherwise, it will increase the number to search with $n+1$ nodes using the genetic algorithm explained before.

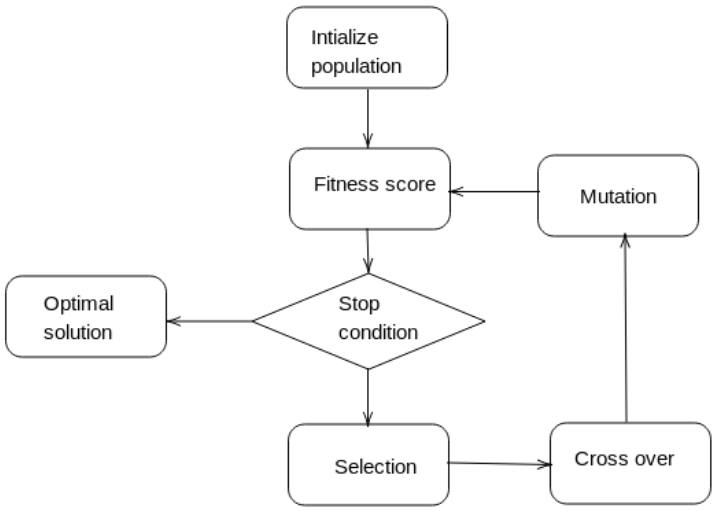


FIGURE 4.11 – Genetic algorithm workflow

In what follows, we will assess the performance of our approach based on extensive simulations. First we describe the simulation setup and the baselines, to which we compare the performance of our model. Then, we present and discuss the results.

4.3.6 Performance evaluation and results analysis

Simulation setup

In order to measure the performance of our method, we tested it using different topologies. We consider two families of random graphs : Erdos Reyni(ER) [28] and Barbasi-Albert (BA) [11].

For ER topologies, they are defined by two parameters, the number of nodes n and the probability pr of edge creation. To ensure that the generated graph is connected we

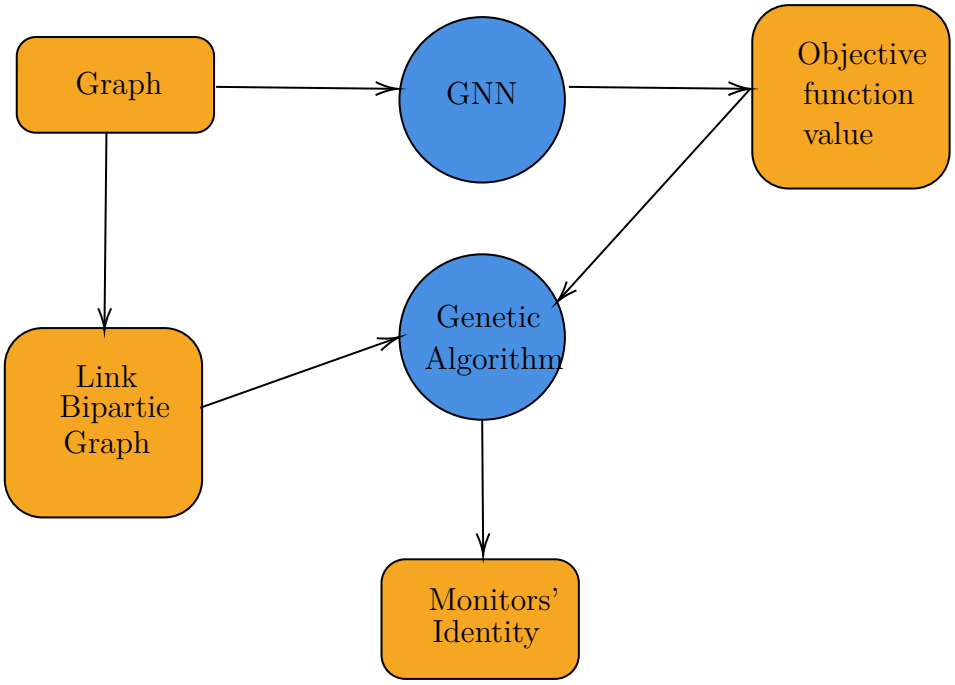


FIGURE 4.12 – Process to determine the identity of the monitors

selected $pr = 2 \times \frac{\ln(n)}{n}$. On the other hand, the BA networks are characterized by the number of nodes n and the number of edges to attach from a new node to existing nodes m_ba, we select $m\_ba = 2$. For each family we consider six values for the number of nodes n : 20, 30, 40, 50, 60 and 70. For space constraint, and without loss of generality, in these tests, we fix the maximal length of cycles k to 6. Other values of k gave similar results.

For each type of graph, and for each number of nodes, we generate 20000 graphs that we split into 15000 graphs for training and 5000 for the test.

For the neural network architecture, in the GNN part, we use a two layers GAT network with 3 heads and we use the Exponential Linear Unit (ELU) as an activation function. Then, we consider a seven-layer MLP where each hidden layer contains, respectively, 256, 128, 64, 32, 16, 8 hidden units and the output layer with only a single neuron. For the training, we set the batch size to 8, and to learn, we use Adam optimizer [52] with a learning rate of 0.0001. Like in the previous section, these hyperparameters were chosen based on the results of a grid search [58]. We tested different combinations of these hyperparameters, and we kept the combination leading to the best performance.

Baselines

We compare our approach, with two methods :

— A greedy solution that takes as input the link maps and selects iteratively the nodes that cover the highest number of links uncovered.

— A genetic algorithm combined with a binary search (GABS) : this is a variant of our proposal but without the GNN part. We directly run the GA to find the optimal solution.

Simulation results

Prediction of the number of monitors

Fig. 4.13 and Fig. 4.14 illustrate the performance of the neural network architecture in predicting the number of monitors. We put the accent on the distribution of the prediction error. The prediction was conducted on the test graphs that the model had never seen during training. In both types of graphs and in all sizes, we observe that the model correctly predicts the optimal number of monitors with a high frequency. Furthermore, when it misses the optimal values, it stays close to it. The absolute value of the error is either zero or one.

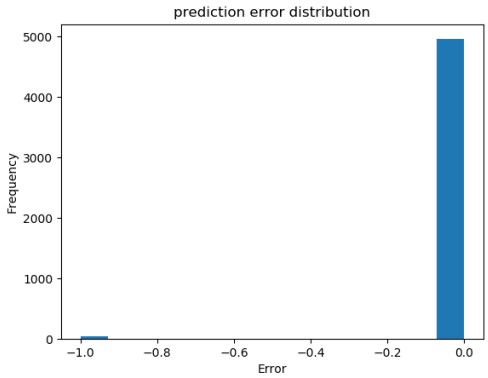
Note that, once the model is trained, our model can reach this performance without recalculating neither the cycles of the graphs nor the link map. The model learned how to directly map the topology of the network into the number of required monitors.

where *optimal_value* represents the optimal number of monitors and *method_value* refers to the number of monitors given by a certain method.

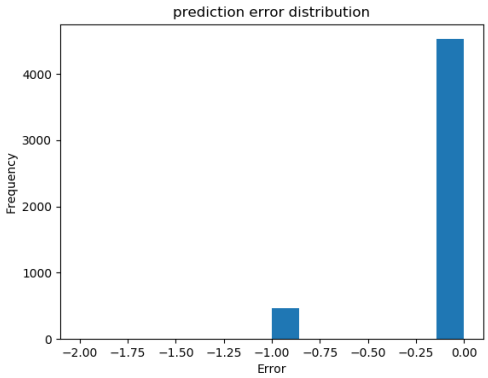
The results for ER and BA graphs are shown, respectively, in Fig. 4.15 and Fig. 4.16. We can observe that our methods outperform the greedy solution in all cases, especially with the ER graphs. In the BA graphs, all solutions have the same performance on small graphs. Our method conserves the same behavior on large graphs while the optimality gap of the greedy method starts to increase.

MonGNN and GABS have the same performance. We, then, compare them in terms of execution time. Fig. 4.17 and Fig. 4.18 illustrate, respectively, the obtained results for the ER and BA graphs.

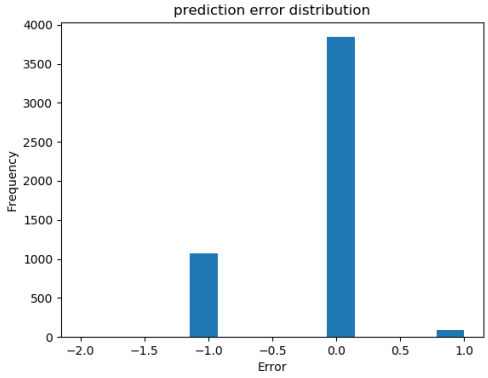
Although both methods have quasi the same performance, we can see that MonGNN is faster than GABS. In fact, with the learning part, our method can predict the number



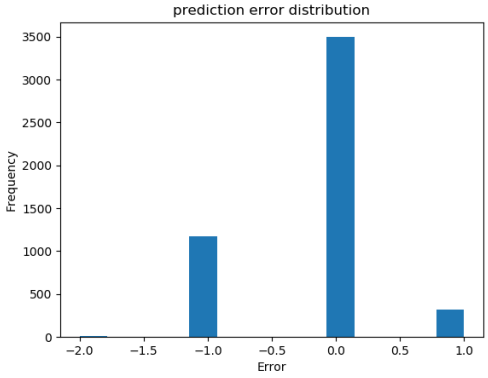
(a) $n = 20$



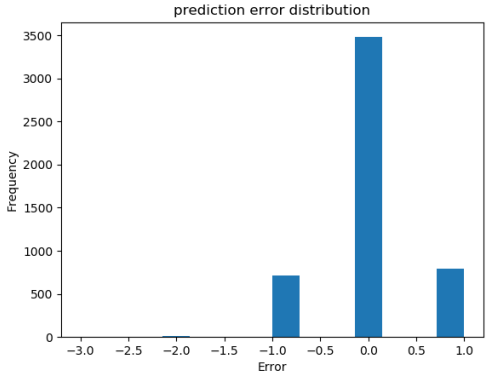
(b) $n = 30$



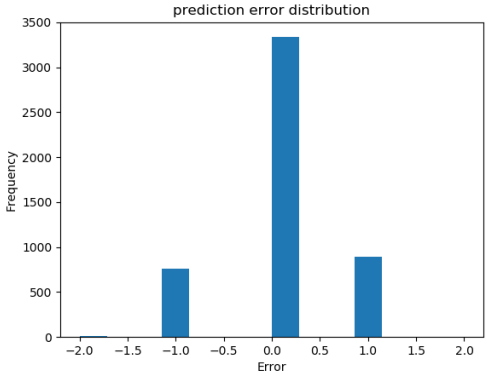
(c) $n = 40$



(d) $n = 50$



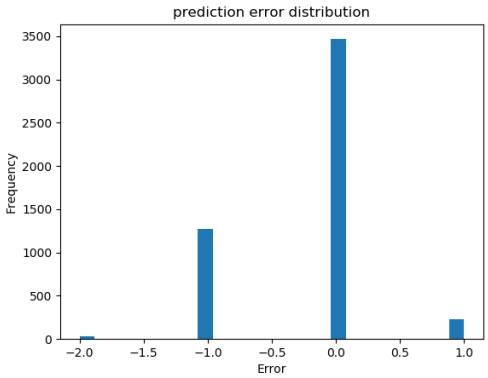
(e) $n = 60$



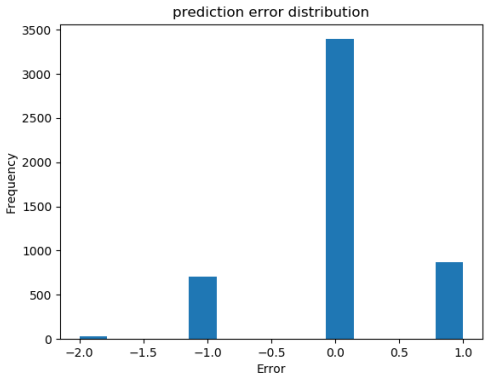
(f) $n = 70$

FIGURE 4.13 – Error prediction of number of monitors with BA graphs

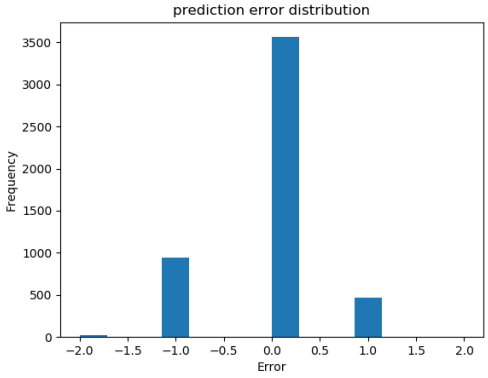
of monitors efficiently. Hence, it effectively reduces the search space and then reduces the time wasted in exploring trivial and unfeasible solutions.



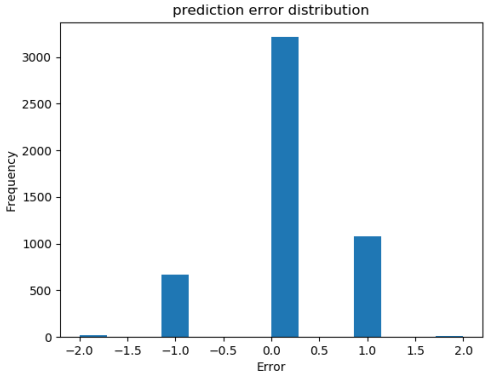
(a) $n = 20$



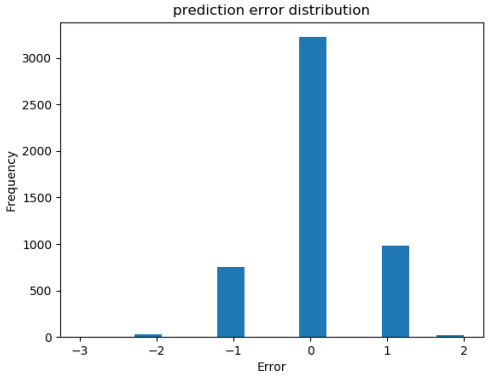
(b) $n = 30$



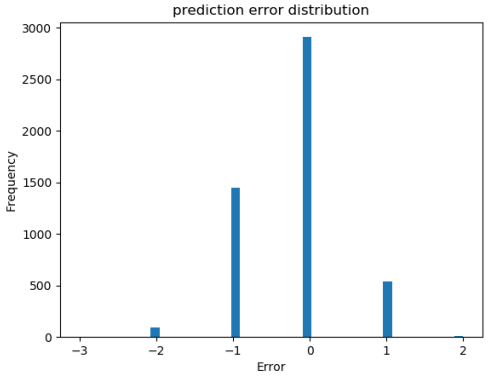
(c) $n = 40$



(d) $n = 50$

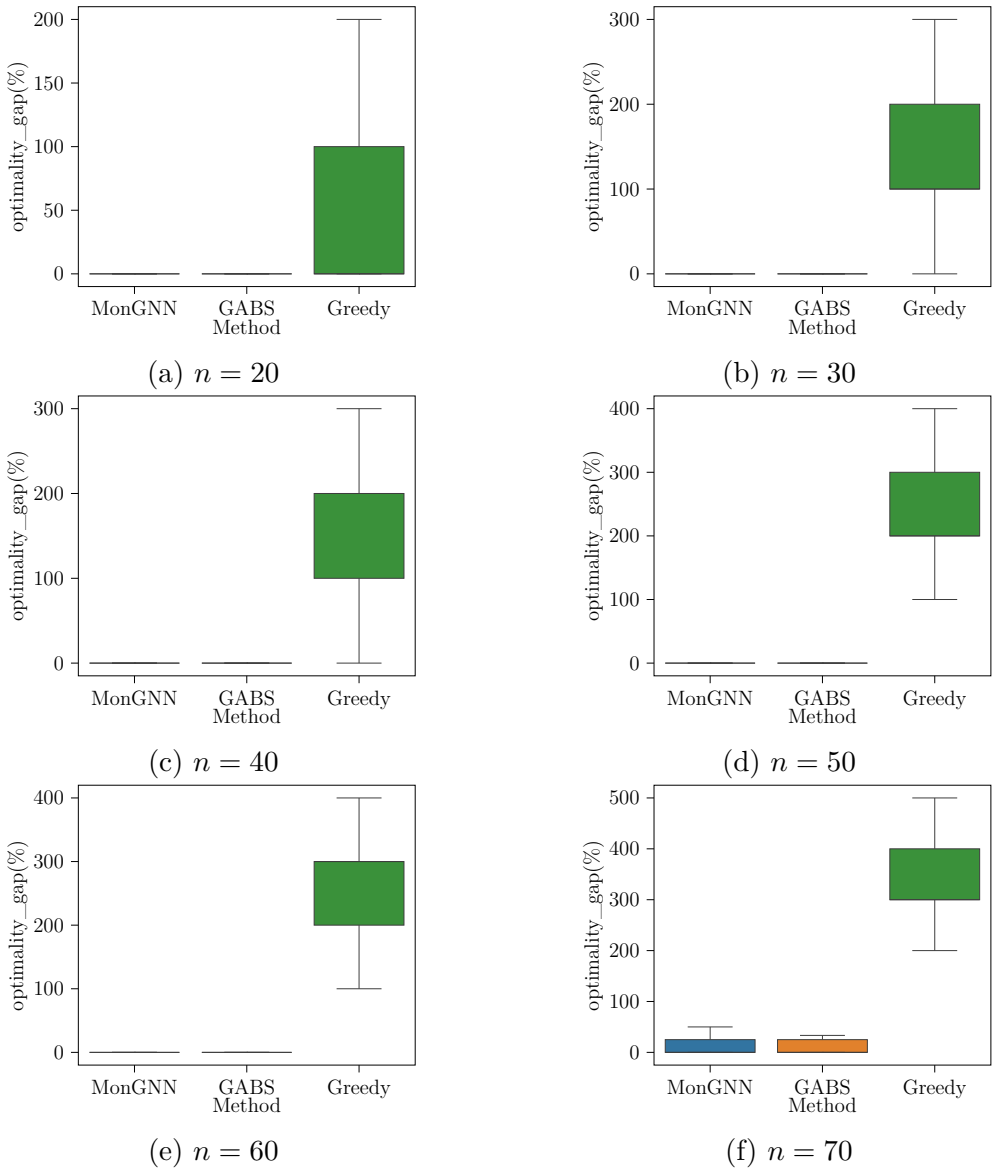


(e) $n = 60$



(f) $n = 70$

FIGURE 4.14 – Error prediction of number of monitors with ER graphs



(a) $n = 20$ (b) $n = 30$

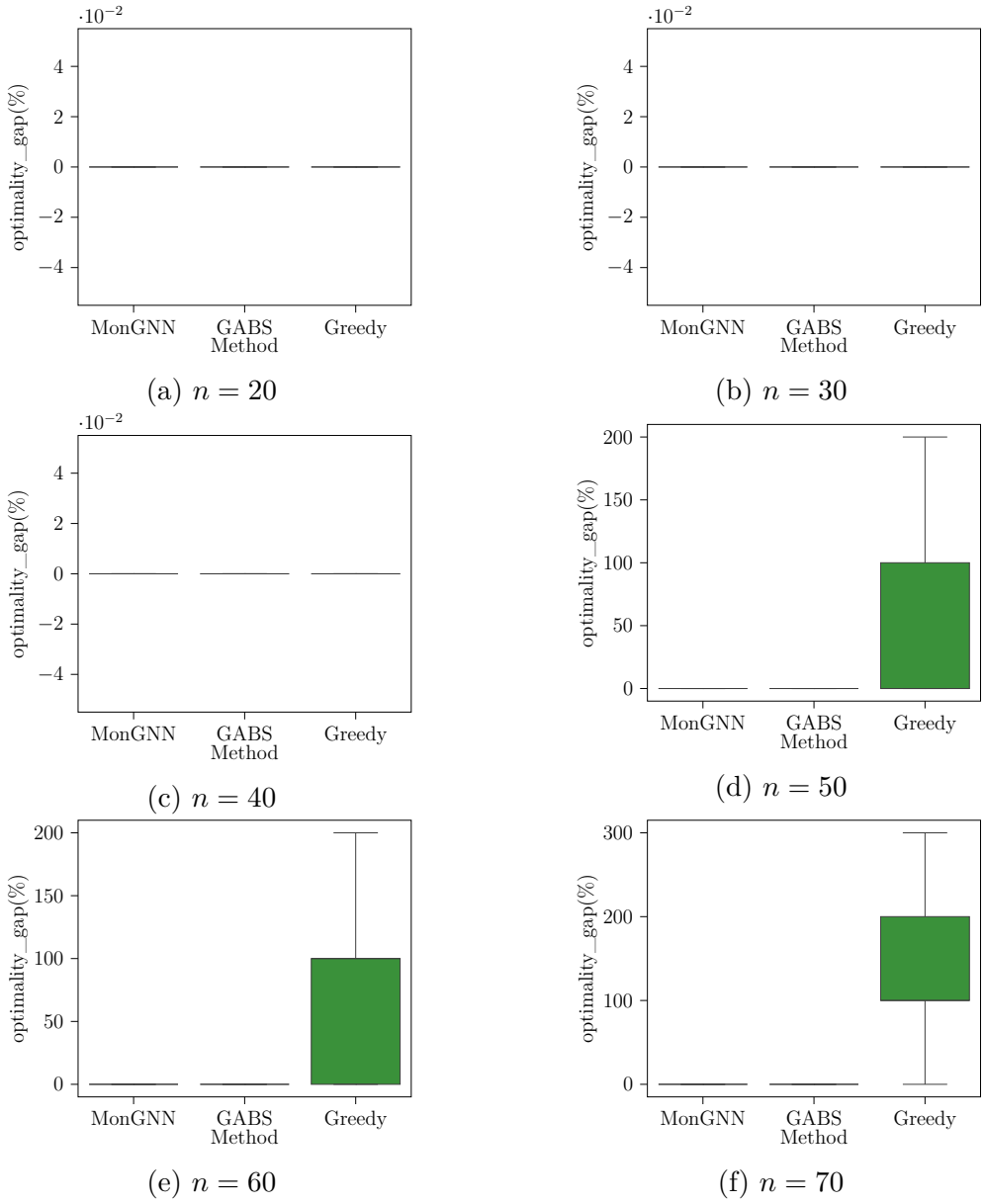
(c) $n = 40$ (d) $n = 50$

(e) $n = 60$ (f) $n = 70$

FIGURE 4.15 – Optimality gap : ER graphs

4.3.7 Summary

In this section, we proposed a new solution to find the optimal monitors' placement to cover all slices with probing cycles. We formulated the task as a set covering problem, and we proposed a learning-based approach to solve it. We divided the task into two subtasks : the prediction of the number of monitors and their identification. We model the neural network architecture using GNNs to solve the first sub-task. Then, we proposed a genetic algorithm to deal with the second one. When compared to existing approaches, the results



(a) $n = 20$ (b) $n = 30$

(c) $n = 40$ (d) $n = 50$

(e) $n = 60$ (f) $n = 70$

FIGURE 4.16 – Optimality gap : BA graphs

demonstrate that MonGNN produces promising outcomes in both stages and can reduce the optimality gap.

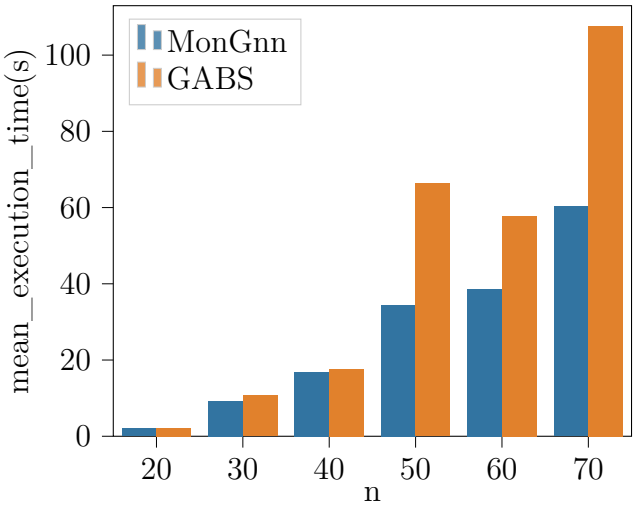


FIGURE 4.17 – ER graphs : Mean execution time(s)

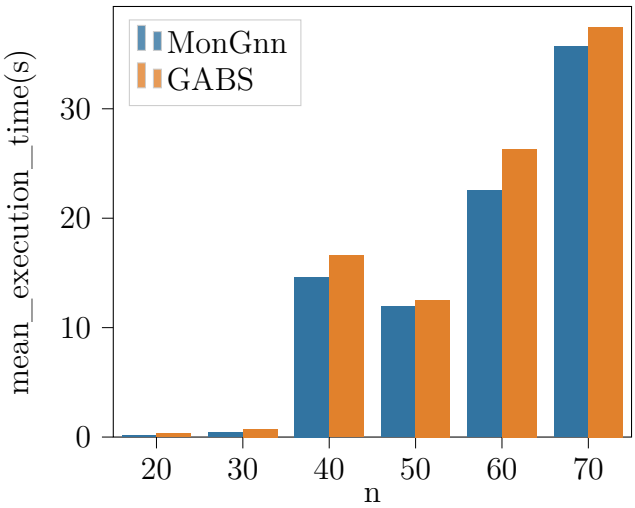


FIGURE 4.18 – BA graphs : Mean execution time (s)

4.4 Conclusion

In this chapter, we focused on monitoring the state of the slices. We proposed two solutions that combine machine learning and network tomography. The first solution concerned the inferring of slices metrics using end-to-end measurements between monitors. On the other hand, the second contribution was about the optimal placement of monitors to cover all slices. In the first contribution, we focused on additive metrics inference. Unlike existing work, we focused on estimating the slices metrics and not the metrics of the physical links that host these slices. Hence, we modeled the inference problem as a regression problem, and we solved it by training neural networks. To overcome the challenges associated with the training of neural networks, we used simulated data in the training step. Finally, we proposed a transfer learning technique in order to handle changes in the physical topology as well as the slices subject of monitoring. We evaluated the performance of our approach using an emulated and simulated network traffic. The results show that our approach outperforms the methods with which we compared in terms of both accuracy and computational complexity.

In the second contribution, we provided a new method for determining the best monitor placement for probing all slices. We rephrased the problem as a set covering problem and provided a learning-based solution. The problem was broken into two parts : predicting the number of monitors and identifying them. To answer the first sub-problem, we use GNNs to represent the neuronal architecture. Then, to deal with the identification phase, we employed genetic algorithms combined with a local search to remedy any errors that may have occurred during the prediction section. The results show that MonGNN achieves promising results when compared to existing methods.

For the first contribution, we intend to expand on this work in the future by combining the first and second stages of network tomography monitoring. Specifically, an intelligent path selection for measurements as well as an intelligent monitor placement for slice monitoring. Furthermore, we intend to employ neural architectures that are appropriate for graph-structured data and apply our method to other real topologies.

When it comes to the second contribution, there are numerous directions to go for the future work :(1) Even if we avoid the process of computing cycles in the prediction phase, we are obliged to do so in order to reveal the identity of monitors, thus, finding a solution that avoids this is useful ; (2) Given that the optimal solution for large instances can not be determined, it would be useful to provide ways to improve the generalization on large

graphs of the model that was trained on small instances.

Chapitre 5

Conclusion and future perspectives

5.1 Conclusions

In order to meet all their expectations, 5G networks will be based on the last advents of network function virtualization, software-defined networking, and network slicing. Flexibility is not the only objective in the 5G and B5G networks. Indeed, a "zero-touch" network strategy is envisioned. The aim is to automate the management and orchestration processes in this new generation of wireless networks. To deal with these complex tasks, and motivated by the success of Machine Learning algorithms in solving complex tasks in several domains, a plethora of works considered ML as a key enabler for 5G, 6G, and beyond networks. Among the challenges to be addressed using AI are the placement of network slices and the monitoring of their states.

In this thesis, we attempted to answer the two following questions :

— How can deep learning models help in solving the placement of network slices ?

— How can a combination of AI and network tomography help in proposing autonomous and flexible ways to monitor 5G and B5G network slices ?

To answer the first question, in Chapter 3, we proposed two deep reinforcement learning-based solutions for the task of placement of network slices. In the first solution, we formulated the task as an RL problem where the state is represented using graphs. Accordingly, we designed a neural network architecture that combines graph neural networks and attention layers to learn from this data type. The neural architecture was trained in an RL manner. The objective was to learn how to map the state to the actions representing the chosen placement for each virtual network function in the slice. These contributions showed that our proposal performs better than other existing methods by exploring the solution space intelligently.

With this approach, the challenge that arises is the exploitation-exploration dilemma. Indeed, when using RL, each time it has to choose an action, the agent is faced with a spot of exploring new options or exploiting the already acquired knowledge. Sometimes this leads to choosing unfeasible actions and hence causes the rejection of a service. To deal with this challenge, our second contribution was to propose a safe AI-based solution for the network slice placement task. Instead of starting without any prior and waste time in exploring all states (which may lead to unsafe exploration), we train an agent to reduce the optimality gap of heuristics. In this contribution, we defined the improvement task as a Markov decision process where the states were represented using heterogeneous graphs. Next, we leveraged the Relational Graph Convolutional Neural Networks (RGCN) to learn automatically from this type of graph. Numerical simulations were conducted in order to test our approach in learning to improve the quality of two heuristics. The obtained results indicated that our proposal was able to increase the performance of the first heuristics by up to 20% and the second heuristic by up to 35%.

Later, to respond to the second question, in Chapter 4, we addressed the task of monitoring the network slices. We tackled two main problems in this context : Inferring the metrics of the slices and optimal monitor placement. In the third contribution, we studied the inference of additive link metrics using neural networks. We defined the problem as a multi-output regression one, and to solve it, we used a Feed-Forward neural network. The neural network was trained in a supervised fashion based on a synthetic dataset. Moreover, to increase the robustness of our approach, we proposed to leverage the techniques of transfer learning to deal with the changes of the topology or the slices we monitor. Simulation results showed that our approach outperforms the existing methods based on pseudo inverse methods.

For the last contribution, we targeted the monitoring placement problem, which is as important as the inference task. Indeed, an efficient monitor placement strategy leads to efficient estimation of the slices metrics. In this contribution, the objective was to find optimal placement for monitors in a way that all slices can be covered by probes. Moreover, we considered the particular case of : cycle probing. The motivation behind this choice is that we will not have to deal with synchronizations issues anymore. Cycle probing can be implemented using Segment Routing (SR). Despite all its advantages, SR puts a constraint on the length of cycle probing. Therefore our problem becomes a variant of the set cover problem. To solve it, we have proposed MonGNN, a solution that divides the task into two subproblems : First, it predicts the number of monitors using Graph

neural networks. Then, it uses genetic algorithms to identify their placement. Simulation results demonstrated that our proposal outperforms some existing methods in terms of both the optimality gap and the computational complexity.

5.2 Limitations

This section summarizes the limitations of the contributions made in this thesis

5.2.1 Network slice placement

In Chapter 3, when placing network slices, the constraints related to links were put only on the bandwidth. However, and as our future work suggests, more work can be done by considering QoS constraints and also by considering that the requirements of slices may evolve and that the placement has to consider a possibility to migrate totally or partially the slices. Furthermore, in this thesis, we focused only on a single domain placement. For the second contribution, the solution was proposed to improve only two heuristics. Therefore it will be more interesting to apply it to new heuristics while taking QoS constraints into consideration.

5.2.2 Network slice monitoring

Later, in Chapter 4, we proposed two solutions to monitor network slices using AI and network tomography. In both solutions, we consider only additive metrics such as delay, jitter, and loss rates. However, non-additive metrics such as bandwidth, failures are also important. Our proposed methods can be extended and adapted to deal with this type of metric. Moreover, in the first section of that chapter, to train the agent to solve the inference problem, we had to collect diverse data. Hence we generated a synthetic training dataset via emulation. However, we can enhance this process to imitate the behavior of real data. Finally, in the second contribution, the simulation tests were not conducted on large instances.

5.3 Future works

Based on the work introduced in this thesis and the limitations we cited above, in this section, we will summarize the future directions of our contributions.

5.3.1 Network slice placement

— **AI-based network slice placement in a multi-domain setup** : The first direction that one may take to extend our proposed solutions in the context of the placement of network slices is to consider the multi-domain setup in both cases, cooperative and non-cooperative. We imagine defining the problem in the context of Multi-agent Reinforcement learning (MARL). In MARL, the objective is to learn a policy for each agent such that the final decision made by all agents maximizes the ultimate goal of the system. Another option is to explore how to combine reinforcement learning with federated learning to deal with this task. Federated learning is a learning paradigm that trains an algorithm across different distributed devices without exchanging data between them. Therefore, each agent will be trained in reinforcement learning based on the domain it belongs to. Hence, it will be useful in the case of non-cooperative domains.

— **AI-based solution to take into consideration dynamicity of slices** : In order to take into consideration the dynamicity of the physical network and the slices deployed on top of it, one solution is to perform reactively and change the placement of slices each time a reconfiguration is needed. However, this solution may lead to a frequent reconfiguration and instability of the services run by the slices. Therefore, a proactive solution is needed. Our idea to deal with this challenge is to encode the state of networks with their potential reconfiguration. The graph neural networks we used to solve the network slice placement do not capture the evolution of a graph with time. A solution to explore is to use Temporal Graph Neural (TGN) networks introduced in [101].

5.3.2 Network slice monitoring

— **Enhance the quality of synthetic data :** In order to avoid the overfitting problem, in the third contribution of the thesis, we train our model using synthetic data that was collected using emulation. A future direction from this point is to enhance the quality of this synthetic data in order to be as close as possible to the real one. The use of Generative Adversarial Networks (GANs) is a promising solution. GANs were proposed by I. Goodfellow et al. [37]. They are a neural architecture composed of two neural networks : The Generator and the Discriminator, which are trained adversarially. The former is in charge of the generation of samples as similar to the

ones seen in the training set. While the role of the latter, is to predict whether an instance is real or fake(produced by the generator). The process consists of training the Generator and the Discriminator alternatively until that the Discriminator can not distinguish between real and fake instances. Hence, to generate synthetic data as similar as the real one, we can use the GANs and avoid the whole process of collecting data several times.

— **Enhancement of MonGnn** : In the last contribution, we presented our proposal MonGnn that divided the task of monitors placement into the prediction of the number of monitors and their identification. As a next step we suggest, unifying the tasks and defining the problem as a node classification challenge that can be solved either in a reinforcement learning manner or a supervised one.

Alongside all these propositions, an important direction that concerns both tasks is the explainability of the deep learning models used to solve these tasks. Indeed, we used the models as black boxes, and we can not explain exactly which features influenced on the final decision made by the neural networks. Hence, in order to build safe and trustworthy models, we suggest extending the proposed methods of this thesis and make them both explainable and interpretable. As an illustration of what we have in mind, we can cite the work [122]

Chapitre 6

Publications

Conference Papers

— Anouar Rkhami, Pham Tran Anh Quang, Yassine Hadjadj-Aoul, Abdelkader Outtagarts, Gerardo Rubino : **On the use of graph neural networks for virtual network embedding.** - *2020 International Symposium on Networks, Computers and Communications (ISNCC)* October 2020, Montreal, Canada [99]

— Anouar Rkhami, Yassine Hadjadj-Aoul, Abdelkader Outtagarts : **Learn to improve : A novel deep reinforcement learning approach for beyond 5G network slicing.** - *18th Annual Consumer Communications and Networking Conference (CCNC))* January 2021, Las Vegas, NV, USA [97]

— Anouar Rkhami, Yassine Hadjadj-Aoul, Abdelkader Outtagarts, Gerardo Rubino : **On the use of machine learning and network tomography for network slices monitoring** - *22nd International Conference on High Performance Switching and Routing (HPSR)* June 2021, Paris, France - **Best Paper Award** [100]

— Anouar Rkhami, Yassine Hadjadj-Aoul, Abdelkader Outtagarts, Gerardo Rubino : **MonGNN : A neuroevolutionary-based solution for 5G network slices monitoring** - *46th Conference on Local Computer Networks (LCN)* October 2021, Edmonton, CANADA [98]

— Ghina Dandachi, Anouar Rkhami, Yassine Hadjadj-Aoul, Abdelkader Outtagarts : **A Robust Deep Learning strategy for VirtualNetwork Embedding using Monte-Carlo** - *Pending*

Posters

— Anouar Rkhami, Pham Tran Anh Quang, Yassine Hadjadj-Aoul, Abdelkader Outtagarts, Gerardo Rubino : **GCN-based VNE placement in 5G and post-5G networks** - *Journées du GDR Réseaux et Systèmes Distribués (Poster Session)* January 2020, Nantes, France

BIBLIOGRAPHIE

[1] *3GPP, "Telecommunication management ;study on management and orchestration of network slicing for next generation network*, Online : https://bit.ly/2qeZd1s.

[2] *5G-HEART : 5G HEalth AquacultuRe and Transport validation trials*, http://5gheart.org/.

[3] *5G-TOURS*, https://5gtours.eu/.

[4] *5G Vertical Innovation Infrastructure, Project 5G-VINNI*, https://www.5g-vinni.eu/.

[5] *5G-ZORRO*, https://www.5gzorro.eu/wp-content/uploads/2021/03/5GZORRO_D4.1_v1.0_final-with-WM.pdf.

[6] I. Afolabi et al., « Network Slicing and Softwarization : A Survey on Principles, Enabling Technologies, and Solutions », in : *IEEE Communications Surveys Tutorials* 20.*3* (2018), p. 2429-2453.

[7] *AI and ML – Enablers for Beyond 5G Networks*, Online : http://doi.org/10.5281/zenodo.4299895.

[8] M.D. Ananth et Rinki Sharma, « Cloud Management Using Network Function Virtualization to Reduce CAPEX and OPEX », in : *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2016, p. 43-47, doi : 10.1109/CICN.2016.17.

[9] François Aubry et al., « SCMon : Leveraging segment routing to improve network monitoring », in : *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, p. 1-9, doi : 10.1109/INFOCOM.2016.7524410.

[10] Yunsheng Bai et al., *SimGNN : A Neural Network Approach to Fast Graph Similarity Computation*, 2020, arXiv : 1808.05689 [cs.LG].

[11] Albert-László Barabási, « Network science », in : *Philosophical Transactions of the Royal Society A : Mathematical, Physical and Engineering Sciences* 371.*1987* (2013), p. 20120375.

[12] Gianni Barlacchi et al., « A multi-source dataset of urban life in the city of Milan and the Province of Trentino », in : *Scientific Data* 2 (2015).

[13] J.E. Beasley, « An algorithm for set covering problem », in : *European Journal of Operational Research* 31.*1* (1987), p. 85-93, issn : 0377-2217, doi : https://doi.org/10.1016/0377-2217(87)90141-X, url : https://www.sciencedirect.com/science/article/pii/037722178790141X.

[14] Y. Bengio et Yann Lecun, « Convolutional Networks for Images, Speech, and Time-Series », in : (nov. 1997).

[15] Jiuyue Cao et al., « VNF-FG design and VNF placement for 5G mobile networks », in : *Science China Information Sciences* 60 (2016), p. 1-15.

[16] Aiyou Chen, Jin Cao et Tian Bu, « Network tomography : Identifiability and fourier domain estimation », in : *IEEE Transactions on Signal Processing* 58.*12* (2010), p. 6029-6039.

[17] Yan Chen, David Bindel et Randy H. Katz, « Tomography-Based Overlay Network Monitoring », in : *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, Miami Beach, FL, USA : Association for Computing Machinery, 2003, 216–231, isbn : 1581137737, doi : 10.1145/948205.948233, url : https://doi.org/10.1145/948205.948233.

[18] Yan Chen et al., « An algebraic approach to practical and scalable overlay network monitoring », in : t. 34, oct. 2004, p. 55-66, doi : 10.1145/1015467.1015475.

[19] Xiang Cheng et al., « Virtual Network Embedding through Topology-Aware Node Ranking », in : *SIGCOMM Comput. Commun. Rev.* 41.*2* (avr. 2011), 38–47, issn : 0146-4833, doi : 10.1145/1971162.1971168, url : https://doi.org/10.1145/1971162.1971168.

[20] Kyunghyun Cho et al., « Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation », in : *CoRR* abs/1406.1078 (2014), arXiv : 1406.1078, url : http://arxiv.org/abs/1406.1078.

[21] Po-Yu Chou et al., « Deep Reinforcement Learning for MEC Streaming with Joint User Association and Resource Management », in : *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, p. 1-7, doi : 10.1109/ICC40277.2020.9149086.

[22] Mosharaf Chowdhury, Muntasir Raihan Rahman et Raouf Boutaba, « ViNEYard : Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping », in : *IEEE/ACM Transactions on Networking* 20.*1* (2012), p. 206-219, doi : 10.1109/TNET.2011.2159308.

[23] Cisco, *Making networks sdn-ready with segment routing*, https ://www.segment-routing.net/images/lightreading$_r$*eport.pdf*, 2017.

[24] Chinmaya Kumar Dehury et Prasan Kumar Sahoo, « DYVINE : Fitness-Based Dynamic Virtual Network Embedding in Cloud Computing », in : *IEEE Journal on Selected Areas in Communications* 37.*5* (2019), p. 1029-1045, doi : 10.1109/JSAC.2019.2906744.

[25] M. Demirci et al., « Overlay network placement for diagnosability », in : *2013 IEEE Global Communications Conference (GLOBECOM)*, 2013, p. 2236-2242, doi : 10.1109/GLOCOM.2013.6831407.

[26] Mahdi Dolati et al., « DeepViNE : Virtual Network Embedding with Deep Reinforcement Learning », in : *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, p. 879-885, doi : 10.1109/INFCOMW.2019.8845171.

[27] W. Dong et al., « Optimal Monitor Assignment for Preferential Link Tomography in Communication Networks », in : *IEEE/ACM Transactions on Networking* 25.*1* (2017), p. 210-223, doi : 10.1109/TNET.2016.2581176.

[28] Paul Erdős et Alfréd Rényi, « On the evolution of random graphs », in : *Publ. Math. Inst. Hung. Acad. Sci* 5.*1* (1960), p. 17-60.

[29] C. Estan et al., *Building a Better NetFlow : Technical Report*, rapp. tech., Cooperative Association for Internet Data Analysis (CAIDA), 2004-06.

[30] *Evolution of Mobile generation from 1G to 6G*, Online : https://5glearning.org/evolution-of-mobile-generation-from-1g-to-6g/.

[31] Jamal Fattah et al., « Forecasting of demand using ARIMA model », in : *International Journal of Engineering Business Management* 10 (oct. 2018), p. 184797901880867, doi : 10.1177/1847979018808673.

[32] Andreas Fischer, « An evaluation methodology for virtual network embeddING », in : (2017).

[33] Y. Gao et al., « Scalpel : Scalable Preferential Link Tomography Based on Graph Trimming », in : *IEEE/ACM Transactions on Networking* 24.*3* (2016), p. 1392-1403, DOI : 10.1109/TNET.2015.2411691.

[34] Amitava Ghosh et al., « 5G Evolution : A View on 5G Cellular Technology Beyond 3GPP Release 15 », in : *IEEE Access* PP (sept. 2019), p. 1-1, DOI : 10.1109/ACCESS.2019.2939938.

[35] Gene H Golub et Christian Reinsch, « Singular value decomposition and least squares solutions », in : *Linear Algebra*, Springer, 1971, p. 134-151.

[36] Long Gong et al., « Toward profit-seeking virtual network embedding algorithm via global resource capacity », in : *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, p. 1-9, DOI : 10.1109/INFOCOM.2014.6847918.

[37] Ian J. Goodfellow et al., *Generative Adversarial Networks*, 2014, arXiv : 1406.2661 [stat.ML].

[38] Abishek Gopalan et Srinivasan Ramasubramanian, « On Identifying Additive Link Metrics Using Linearly Independent Cycles and Paths », in : *IEEE/ACM Transactions on Networking* 20.*3* (2012), p. 906-916, DOI : 10.1109/TNET.2011.2174648.

[39] Rabah Guedrez et al., « Label encoding algorithm for MPLS Segment Routing », in : *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, 2016, p. 113-117, DOI : 10.1109/NCA.2016.7778603.

[40] A. Gupta et al., « A Scalable Approach for Service Chain Mapping With Multiple SC Instances in a Wide-Area Network », in : *IEEE Journal on Selected Areas in Communications* 36.*3* (2018), p. 529-541, ISSN : 0733-8716, DOI : 10.1109/JSAC.2018.2815298.

[41] O. Gurewitz et M. Sidi, « Estimating one-way delays from cyclic-path delay measurements », in : *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, t. 2, 2001, 1038-1044 vol.2, DOI : 10.1109/INFCOM.2001.916297.

[42] Soroush Haeri et Ljiljana Trajković, « Virtual Network Embedding via Monte Carlo Tree Search », in : *IEEE Transactions on Cybernetics* 48.*2* (2018), p. 510-521, DOI : 10.1109/TCYB.2016.2645123.

[43] Ting He et al., *Network Tomography*, Cambridge, England, UK : Cambridge University Press, 2021, ISBN : 978-1-10842148-5, URL : https://www-cambridge-org.passerelle.univ-rennes1.fr/fr/academic/subjects/engineering/communications-and-signal-processing/network-tomography-identifiability-measurement-design-and-network-state-inference?format=HB&isbn=9781108421485.

[44] Ting He et al., « Robust and Efficient Monitor Placement for Network Tomography in Dynamic Networks », in : *IEEE/ACM Transactions on Networking* 25.*3* (2017), p. 1732-1745, DOI : 10.1109/TNET.2016.2642185.

[45] Sepp Hochreiter et Jürgen Schmidhuber, « Long Short-Term Memory », in : *Neural Computation* 9.*8* (nov. 1997), p. 1735-1780, ISSN : 0899-7667, DOI : 10.1162/neco.1997.9.8.1735, eprint : https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf, URL : https://doi.org/10.1162/neco.1997.9.8.1735.

[46] K. Hornik, M. Stinchcombe et H. White, « Multilayer Feedforward Networks Are Universal Approximators », in : *Neural Netw.* 2.*5* (juil. 1989), 359–366, ISSN : 0893-6080.

[47] *IMT Traffic estimates for the years 2020 to 2030*, Online : https://www.itu.int/pub/R-REP-M.2370-2015/.

[48] *Internet Speed Test — Fast.com*, Online : https://fast.com/.

[49] Insun Jang et al., « Joint Optimization of Service Function Placement and Flow Distribution for Service Function Chaining », in : *IEEE Journal on Selected Areas in Communications* 35.*11* (2017), p. 2532-2541, DOI : 10.1109/JSAC.2017.2760162.

[50] Selma Khebbache, Makhlouf Hadji et Djamal Zeghlache, « A multi-objective non-dominated sorting genetic algorithm for VNF chains placement », in : *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2018, p. 1-4, DOI : 10.1109/CCNC.2018.8319250.

[51] Selma Khebbache, Makhlouf Hadji et Djamal Zeghlache, « Scalable and cost-efficient algorithms for VNF chaining and placement problem », in : *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, 2017, p. 92-99, DOI : 10.1109/ICIN.2017.7899395.

[52] Diederik P Kingma et Jimmy Ba, « Adam : A method for stochastic optimization », in : *arXiv preprint arXiv :1412.6980* (2014).

[53] Thomas N. Kipf et Max Welling, « Semi-Supervised Classification with Graph Convolutional Networks », in : *CoRR* abs/1609.02907 (2016), arXiv : 1609.02907, url : http://arxiv.org/abs/1609.02907.

[54] S. Knight et al., « The Internet Topology Zoo », in : *Selected Areas in Communications, IEEE Journal on* 29.*9* (2011), p. 1765 -1775, issn : 0733-8716, doi : 10.1109/JSAC.2011.111002.

[55] Haneul Ko et al., « Optimal placement of service function in service function chaining », in : *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2016, p. 102-105, doi : 10.1109/ICUFN.2016.7536993.

[56] Vijaymohan R Konda et Vivek S Borkar, « Actor-Critic–Type Learning Algorithms for Markov Decision Processes », in : *SIAM Journal on control and Optimization* 38.*1* (1999), p. 94-123.

[57] D. Kreutz et al., « Software-Defined Networking : A Comprehensive Survey », in : *Proceedings of the IEEE* 103.*1* (2015), p. 14-76.

[58] Steven M LaValle, Michael S Branicky et Stephen R Lindemann, « On the relationship between classical grid search and probabilistic roadmaps », in : *The International Journal of Robotics Research* 23.*7-8* (2004), p. 673-692.

[59] Earl Lawrence, George Michailidis et Vijayan N. Nair, « Network delay tomography using flexicast experiments », in : *B* (2006), p. 785-813.

[60] Earl Lawrence et al., « Network Tomography : A Review and Recent Developments », in : (juil. 2006), doi : 10.1142/9781860948886_0016.

[61] Xiaoqian Li et Kwan L.Yeung, « ILP Formulation for Monitoring-Cycle Construction Using Segment Routing », in : *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, 2018, p. 485-492, doi : 10.1109/LCN.2018.8638040.

[62] Xiaoqian Li et Kwan L. Yeung, « Designing Network Monitoring Schemes Based on Segment Routing », in : *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, p. 1-6, doi : 10.1109/GLOCOM.2017.8254486.

[63] Yong Li et Min Chen, « Software-defined network function virtualization : A survey », in : *IEEE Access* 3 (2015), p. 2542-2553.

[64] C. Liu et al., « Multicast vs. unicast for loss tomography on tree topologies », in : *MILCOM 2015 - 2015 IEEE Military Communications Conference*, 2015, p. 312-317, DOI : 10.1109/MILCOM.2015.7357461.

[65] Marcelo Caggiani Luizelli et al., « A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining », in : *Computer Communications* 102 (2017), p. 67-77, ISSN : 0140-3664, DOI : https://doi.org/10.1016/j.comcom.2016.11.002, URL : https://www.sciencedirect.com/science/article/pii/S0140366416305485.

[66] Marcelo Caggiani Luizelli et al., « Piecing together the NFV provisioning puzzle : Efficient placement and chaining of virtual network functions », in : *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, p. 98-106, DOI : 10.1109/INM.2015.7140281.

[67] L. Ma et al., « Efficient Identification of Additive Link Metrics via Network Tomography », in : *2013 IEEE 33rd International Conference on Distributed Computing Systems*, 2013, p. 581-590, DOI : 10.1109/ICDCS.2013.24.

[68] L. Ma et al., « Inferring Link Metrics From End-To-End Path Measurements : Identifiability and Monitor Placement », in : *IEEE/ACM Transactions on Networking* 22.4 (2014), p. 1351-1368, DOI : 10.1109/TNET.2014.2328668.

[69] Liang Ma, Ziyao Zhang et Mudhakar Srivatsa, *Neural Network Tomography*, 2020, arXiv : 2001.02942 [cs.NI].

[70] Liang Ma et al., « Identifiability of Link Metrics Based on End-to-End Path Measurements », in : *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, Barcelona, Spain : Association for Computing Machinery, 2013, 391–404, ISBN : 9781450319539, DOI : 10.1145/2504730.2504738, URL : https://doi.org/10.1145/2504730.2504738.

[71] Liang Ma et al., « Link identifiability in communication networks with two monitors », in : *2013 IEEE Global Communications Conference (GLOBECOM)*, 2013, p. 1513-1518, DOI : 10.1109/GLOCOM.2013.6831288.

[72] Ratul Mahajan et al., « Inferring Link Weights Using End-to-End Measurements », in : *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, Marseille, France : Association for Computing Machinery,

2002, 231–236, ISBN : 158113603X, DOI : 10.1145/637201.637237, URL : https://doi.org/10.1145/637201.637237.

[73] Nick McKeown et al., « OpenFlow : Enabling innovation in campus networks », in : *Computer Communication Review* 38 (avr. 2008), p. 69-74, DOI : 10.1145/1355734.1355746.

[74] *Measurement lab*, Online : https://www.measurementlab.net/.

[75] Marouen Mechtri, Chaima Ghribi et Djamal Zeghlache, « A Scalable Algorithm for the Placement of Service Function Chains », in : *IEEE Transactions on Network and Service Management* 13.*3* (2016), p. 533-546, DOI : 10.1109/TNSM.2016.2598068.

[76] Albert Mestres et al., « Knowledge-Defined Networking », in : *ACM SIGCOMM Computer Communication Review* 47.*3* (2017), 2–10, ISSN : 0146-4833, DOI : 10.1145/3138808.3138810, URL : http://dx.doi.org/10.1145/3138808.3138810.

[77] Rashid Mijumbi et al., « Design and evaluation of learning algorithms for dynamic resource management in virtual networks », in : *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, p. 1-9, DOI : 10.1109/NOMS.2014.6838258.

[78] Tom M. Mitchell, *Machine Learning*, New York : McGraw-Hill, 1997, ISBN : 978-0-07-042807-2.

[79] Volodymyr Mnih et al., *Asynchronous Methods for Deep Reinforcement Learning*, 2016, arXiv : 1602.01783 [cs.LG].

[80] *NGMN, 5G White Paper 2, accessible at*, https://www.ngmn.org/work-programme/5g-whitepaper-2.html.

[81] Jose Ordonez-Lucena et al., « Network Slicing for 5G with SDN/NFV : Concepts, Architectures, and Challenges », in : *IEEE Communications Magazine* 55.*5* (2017), p. 80-87, DOI : 10.1109/MCOM.2017.1600935.

[82] Adam Paszke et al., « PyTorch : An Imperative Style, High-Performance Deep Learning Library », in : *Advances in Neural Information Processing Systems 32*, sous la dir. de H. Wallach et al., Curran Associates, Inc., 2019, p. 8024-8035, URL : http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[83] Jianing Pei et al., « Efficiently Embedding Service Function Chains with Dynamic Virtual Network Function Placement in Geo-Distributed Cloud System », in : *IEEE Transactions on Parallel and Distributed Systems* 30.*10* (2019), p. 2179-2192, DOI : 10.1109/TPDS.2018.2880992.

[84] P. Phaal, S. Panchen et N. McKee, *RFC3176 : InMon Corporation's SFlow : A Method for Monitoring Traffic in Switched and Routed Networks*, USA, 2001.

[85] Quang Pham Tran Anh et al., « Virtual network function-forwarding graph embedding : A genetic algorithm approach », in : *International Journal of Communication Systems* 33 (août 2019), e4098, DOI : 10.1002/dac.4098.

[86] Marius-Constantin Popescu et al., « Multilayer Perceptron and Neural Networks », in : *WSEAS Trans. Cir. and Sys.* 8.*7* (juil. 2009), 579–588, ISSN : 1109-2734.

[87] *Project 5G-CLARITY - Beyond 5G multi-tenant private networks integrating cellular, Wi-Fi, and LiFi, powered by artificial intelligence and intent-based policy*, https://www.5gclarity.com/.

[88] *Project 5G-COMPLETE - A unified network, computational and storage resource management framework targeting end-to-end performance optimization for secure 5G multi-technology and multitenancy environments*, https://5gcomplete.eu/.

[89] *Project 5GENESIS - 5th Generation End-to-end Network, Experimentation, System Integration, and Showcasing*, https://5genesis.eu/.

[90] *Project 5GROWTH - 5G-enabled Growth in Vertical Industries*, https://5growth.eu/.

[91] *Project MONB5G - 5th Generation End-to-end Network, Experimentation, System Integration, and Showcasing*, https://www.monb5g.eu./.

[92] M. Rahali, G. Rubino et J.-M. Sanner, « TOM : a self-trained Tomography solution for Overlay networks Monitoring », in : *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC'20)*, 2020, p. 1-6, DOI : 10.1109/CCNC46108.2020.9045301.

[93] M. Rahali, G. Rubino et J.-M. Sanner, « Unicast inference of additive metrics in general network topologies », in : *Proc. of the 27th IEEE Int. Symp. on the Modeling, Analysis, and Simulation of Comp. and Telecomm. Sys. (MASCOTS'19)*, 2019.

[94] Mohamed Rahali, « SDN and NFV networks : a new boost and opportunity for network tomography », Theses, Université Rennes 1, déc. 2020, URL : https://tel.archives-ouvertes.fr/tel-03261578.

[95] Mohamed Rahali, Jean-Michel Sanner et Gerardo Rubino, « FEAL : A source routing Framework for Efficient Anomaly Localization », in : *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, p. 1-7, DOI : 10.1109/ICC40277.2020.9148725.

[96] Hassan Ramchoun et al., « Multilayer Perceptron : Architecture Optimization and Training. », in : *IJIMAI* 4.1 (2016), p. 26-30.

[97] Anouar Rkhami, Yassine Hadjadj-Aoul et Abdelkader Outtagarts, « Learn to improve : A novel deep reinforcement learning approach for beyond 5G network slicing », in : *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, 2021, p. 1-6, DOI : 10.1109/CCNC49032.2021.9369463.

[98] Anouar Rkhami et al., « MonGNN : A neuroevolutionary-based solution for 5G network slices monitoring », in : *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 2021, p. 185-192, DOI : 10.1109/LCN52139.2021.9524880.

[99] Anouar Rkhami et al., « On the Use of Graph Neural Networks for Virtual Network Embedding », in : *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, p. 1-6, DOI : 10.1109/ISNCC49221.2020.9297270.

[100] Anouar Rkhami et al., « On the use of machine learning and network tomography for network slices monitoring », in : *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, 2021, p. 1-7, DOI : 10.1109/HPSR52026.2021.9481795.

[101] Emanuele Rossi et al., « Temporal Graph Networks for Deep Learning on Dynamic Graphs », in : *CoRR* abs/2006.10637 (2020), arXiv : 2006.10637, URL : https://arxiv.org/abs/2006.10637.

[102] « Mean Squared Error », in : *Encyclopedia of Machine Learning*, sous la dir. de Claude Sammut et Geoffrey I Webb, Boston, MA : Springer US, 2010, p. 653, ISBN : 978-0-387-30164-8, DOI : 10.1007/978-0-387-30164-8_528, URL : https://doi.org/10.1007/978-0-387-30164-8{_}528.

[103] Franco Scarselli et al., « The graph neural network model », in : *IEEE Transactions on Neural Networks* 20.*1* (2008), p. 61-80.

[104] Michael Schlichtkrull et al., « Modeling relational data with graph convolutional networks », in : *European Semantic Web Conference*, Springer, 2018, p. 593-607.

[105] Nashid Shahriar et al., « Virtual Network Survivability Through Joint Spare Capacity Allocation and Embedding », in : *IEEE Journal on Selected Areas in Communications* 36.*3* (2018), p. 502-518, doi : 10.1109/JSAC.2018.2815430.

[106] Chuan Shi et al., « A survey of heterogeneous information network analysis », in : *IEEE Transactions on Knowledge and Data Engineering* 29.*1* (2016), p. 17-37.

[107] Richard Socher et al., « Reasoning With Neural Tensor Networks for Knowledge Base Completion », in : *Advances in Neural Information Processing Systems*, sous la dir. de C. J. C. Burges et al., t. 26, Curran Associates, Inc., 2013, url : https://proceedings.neurips.cc/paper/2013/file/b337e84de8752b27eda3a12363109e80-Paper.pdf.

[108] Oussama Soualah et al., « A reliable virtual network embedding algorithm based on game theory within cloud's backbone », in : *2014 IEEE International Conference on Communications (ICC)*, 2014, p. 2975-2981, doi : 10.1109/ICC.2014.6883777.

[109] S. M. Srinivasan, T. Truong-Huu et M. Gurusamy, « Machine Learning-Based Link Fault Identification and Localization in Complex Networks », in : *IEEE Internet of Things Journal* 6.*4* (2019), p. 6556-6566, doi : 10.1109/JIOT.2019.2908019.

[110] Mohammad M. Tajiki et al., « Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining », in : *IEEE Transactions on Network and Service Management* 16.*1* (2019), p. 374-388, doi : 10.1109/TNSM.2018.2873225.

[111] Nicolas Tastevin, Mathis Obadia et Mathieu Bouet, « A graph approach to placement of Service Functions Chains », in : *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, p. 134-141, doi : 10.23919/INM.2017.7987273.

[112] *The ISP speed index from netflix*, Online : https : https://ispspeedindex.netflix.com/.

[113] A. Tomassilli et al., « Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints », in : *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, p. 774-782, DOI : 10.1109/INFOCOM.2018.8486275.

[114] Yehuda Vardi, « Network tomography : Estimating source-destination traffic intensities from link data », in : *Journal of the American statistical association* 91.*433* (1996), p. 365-377.

[115] András Varga, « Discrete event simulation system », in : *Proc. of the European Simulation Multiconference (ESM'2001)*, 2001, p. 1-7.

[116] Petar Veličković et al., « Graph attention networks », in : *arXiv preprint arXiv :1710.10903* (2017).

[117] Minjie Wang et al., *Deep Graph Library : A Graph-Centric, Highly-Performant Package for Graph Neural Networks*, 2020, arXiv : 1909.01315 [cs.LG].

[118] B. M. Waxman, « Routing of multipoint connections », in : *IEEE Journal on Selected Areas in Communications* 6.*9* (1988), p. 1617-1622.

[119] Ronald J. Williams, « Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning », in : *Mach. Learn.* 8.*3–4* (mai 1992), 229–256, ISSN : 0885-6125, DOI : 10.1007/BF00992696, URL : https://doi.org/10.1007/BF00992696.

[120] Zonghan Wu et al., « A Comprehensive Survey on Graph Neural Networks », in : *IEEE Transactions on Neural Networks and Learning Systems* 32.*1* (2021), 4–24, ISSN : 2162-2388, DOI : 10.1109/tnnls.2020.2978386, URL : http://dx.doi.org/10.1109/TNNLS.2020.2978386.

[121] Yikai Xiao et al., « NFVdeep : Adaptive Online Service Function Chain Deployment with Deep Reinforcement Learning », in : *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, 2019, p. 1-10, DOI : 10.1145/3326285.3329056.

[122] Ning Xie et al., *Explainable Deep Learning : A Field Guide for the Uninitiated*, 2020, arXiv : 2004.14545 [cs.LG].

[123] R. Yang et al., « On the Optimal Monitor Placement for Inferring Additive Metrics of Interested Paths », in : *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, p. 2141-2149, DOI : 10.1109/INFOCOM.2018.8486423.

[124] Haipeng YAO et al., « A novel reinforcement learning algorithm for virtual network embedding », in : *Neurocomputing* 284 (2018), p. 1-9, ISSN : 0925-2312, DOI : https://doi.org/10.1016/j.neucom.2018.01.025, URL : https://www.sciencedirect.com/science/article/pii/S0925231218300420.

[125] Xue YING, « An Overview of Overfitting and its Solutions », in : *Journal of Physics : Conference Series* 1168 (fév. 2019), p. 022022, DOI : 10.1088/1742-6596/1168/2/022022.

[126] Jiaxuan YOU et al., *GraphRNN : Generating Realistic Graphs with Deep Auto-regressive Models*, cite arxiv :1802.08773Comment : ICML 2018, 2018, URL : http://arxiv.org/abs/1802.08773.

[127] Chunyan YU et al., « PAME : Evolutionary membrane computing for virtual network embedding », in : *J. Parallel Distributed Comput.* 111 (2018), p. 136-151.

[128] Ying YUAN et al., « A Q-learning-based approach for virtual network embedding in data center », in : *Neural Computing and Applications* 32 (2019), p. 1995-2004.

[129] Chaoyun ZHANG et Paul PATRAS, *Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks*, 2017, arXiv : 1712.08083 [cs.NI].

[130] Zhongbao ZHANG et al., « A unified enhanced particle swarm optimization-based virtual network embedding algorithm », in : *Int. J. Commun. Syst.* 26 (2013), p. 1054-1073.

[131] Fuzhen ZHUANG et al., *A Comprehensive Survey on Transfer Learning*, 2020, arXiv : 1911.02685 [cs.LG].

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Du placement des services à la surveillance des services dans les réseaux 5G et post-5G

Mot clés : 5G, Apprentissage profond, Découpage du réseau,réseaux de neurones des graphs, tomographie des réseaux, apprentissage profond par renforcement

Résumé : Les réseaux 5G et au-delà sont destinés à servir un large éventail de services réseau aux besoins très disparates tout en utilisant la même infrastructure physique. En scindant l'infrastructure physique en un ensemble de réseaux virtuels, chacun exploitant un service spécifique, le Network Slicing (NS) permettra la coexistence de ces services. En dépit de ses avantages, le NS est complexe d'un point de vue technique puisqu'il s'agit d'un problème NP-hard. La première section de la thèse explore le potentiel de l'apprentissage par renforcement profond (DRL) basé sur des graphes de réseaux neuronaux pour résoudre le problème du placement des tranches de réseau et remédier aux limites des techniques existantes. Deux approches sont proposées : la première consiste à apprendre à résoudre automatiquement le problème du placement. Plutôt que de se limiter à la topologie de l'infrastructure physique ou à extraire manuellement des caractéristiques, le problème est formulé sous la forme d'un processus de décision markovien qui est résolu à l'aide d'un réseau de neurones convolutif à base de graphes pour apprendre à découvrir une solution optimale. Ensuite, plutôt que de former un agent DRL de zéro pour identifier la meilleure solution, ce qui pourrait entraîner un défaut de fiabilité, un agent est présenté pour réduire l'écart d'optimalité des heuristiques existantes. Une fois les tranches placées, la surveillance de l'état des tranches de réseau devient une priorité pour s'assurer que les SLAs sont respectés. Ainsi, dans la deuxième partie de la thèse, il est proposé d'utiliser des techniques d'apprentissage automatique et la tomographie réseau (NT) pour surveiller les tranches de réseau. Il y a deux problèmes majeurs à prendre en compte. Premièrement, les métriques de slices sont déduites sur la base de diverses mesures de bout en bout entre les moniteurs, ainsi que du placement efficace des moniteurs. Des réseaux neuronaux sont utilisés pour traiter l'inférence des métriques. Une approche d'apprentissage par transfert est également utilisée pour faire face aux changements qui peuvent se produire sur les slices surveillés ou sur la topologie physique sur laquelle elles sont placées. Des sondes cycliques sont envisagées pour le problème du placement des moniteurs. Le problème est formulé comme une variante du problème de couverture par ensembles. En raison de sa complexité, il est proposé d'introduire une solution autonome basée sur des réseaux neuronaux à base de graphes (GNN) et des algorithmes génétiques pour trouver un compromis entre la qualité du placement des moniteurs et le coût pour y parvenir.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Title: From services placement to services monitoring in 5G and post-5G networks

Keywords: 5G, Network Slicing, Deep Learning, Graph neural network, Network Tomography, Deep Reinforcement learning

Abstract: 5G networks and beyond are intended to serve a wide range of network services with widely disparate needs while using the same physical infrastructure. By splitting the physical infrastructure into a set of virtual networks, each operating a specific service, Network Slicing (NS) will allow these services to coexist. Despite its advantages, NS is technically complex since it is an NP-hard problem. The first section of the thesis explores the potential of deep reinforcement learning (DRL) based on graph neural networks to solve the problem of network slice placement and overcome the limitations of existing techniques. Two approaches are proposed: the first is to learn how to automatically solve the placement problem. Rather than being limited to the topology of the physical infrastructure or manually extracting features, the problem is formulated as a Markovian decision process that is solved using a convolutional neural network based of graphs to learn how to discover an optimal solution. Then, rather than training a DRL agent from scratch to identify the best solution, which could lead to a reliability issue, an agent is presented to reduce the optimality gap from existing heuristics. Once the slices are placed, monitoring the status of the network slices becomes a priority to ensure that SLAs are met. Thus, in the second part of the thesis, it is proposed to use machine learning techniques and network tomography (NT) to monitor network slices. There are two major issues to consider. First, slice metrics are inferred based on various end-to-end measurements between monitors, as well as the efficient placement of the monitors. Neural networks are used to process inference from metrics. A transfer learning approach is also used to deal with changes that may occur on the monitored slices or on the physical topology on which they are placed. Cyclic probes are considered for the problem of placement of monitors. The problem is formulated as a variant of the set covering problem. Due to its complexity, it is proposed to introduce an autonomous solution based on graph-based neural networks (GNN) and genetic algorithms to find a compromise between the quality of the placement of the monitors and the cost to achieve it.