Université Sultan Moulay Slimane
Faculté des Sciences et Techniques
Département Informatique
- Béni Mellal -

Master : **Informatique Décisionelle**

*****

Projet de fin d'études

Optimisation d'Allocation des Ressources dans le Mobile Edge Computing par l'Apprentissage par Renforcement Multi-Agents (MARL)

Réalisé par :
Youssef ELMIR

Encadré par :
Pr : Driss AIT OMAR

Soutenue le 21/06/2023 **devant le jury composé de**

Mr Mohamed FAKIR	**Prof. à faculté des Sciences et Techniques, Beni Mellal**
Mr Mohamed BASLAM	**Prof. à faculté des Sciences et Techniques, Beni Mellal**
Mr Rachid EL AYACHI	**Prof. à faculté des Sciences et Techniques, Beni Mellal**
Mr Hamid GARMANI	**Prof. à faculté des Sciences et Techniques, Beni Mellal**

Année Universitaire : **2022-2023**

DÉDICACES

J'offre ce modeste travail :

A mes chers parents,
Mais aucune dédicace ne serait témoin de mon amour profond, de mon immense gratitude et de mon plus grand respect, car je ne pourrai jamais oublier la tendresse et l'amour dévoué par lesquels, ils m'ont toujours entouré depuis mon enfance. Ils ont toujours été là pour les bons conseils. Leur affection et leur soutien ont été d'une grande aide tout au long de ma vie professionnelle et personnelle. Puissent-ils trouver dans ce modeste travail ma gratitude pour tous leurs efforts.

A toute ma famille.

A tous mes amis, et à tous ceux que j'aime et à toutes les personnes qui m'ont encouragé et qui ont pris la peine de me soutenir durant ces deux années de formation.

A mes chers Professeurs du master Informatique Décisionelle

A *toute la promotion Master* MID 2021-2023

A *tous les membres de l'administration* de la faculté des Sciences et Techniques de béni mellal.

Et à tous les *chers lecteurs.*

REMERCIEMENTS

Je tiens à exprimer ma sincère gratitude et ma profonde reconnaissance envers toutes les personnes qui ont contribué à la réalisation de mon projet de fin d'étude. C'est avec une immense satisfaction que j'écris ces lignes pour exprimer ma reconnaissance envers ceux qui m'ont apporté leur aide précieuse.

Avant tout, je souhaite exprimer ma reconnaissance à **Allah**, dont l'aide et la guidance m'ont permis de persévérer et de trouver le courage nécessaire tout au long de ces années d'études. Je suis également extrêmement reconnaissant envers mes parents pour leur soutien inconditionnel et leurs encouragements constants.

Je tiens à exprimer ma profonde gratitude envers mon encadrant, **M. Driss Ait Omar**, pour son précieux accompagnement tout au long de mon projet. Sa disponibilité, ses conseils éclairés et sa contribution en termes de concepts et de documentation ont été d'une importance capitale pour la réussite de mon projet.

Je tiens également à remercier les membres du jury pour l'intérêt qu'ils ont porté à ce travail et pour les remarques constructives qu'ils ont faites sur notre projet, et à tous mes professeurs pour leurs efforts et leur support toute au long de ces années d'études. Enfin, merci à tous ceux qui nous ont soutenus et encouragés de près comme de loin tout au long de la réalisation de ce projet de fin d'études.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES ABBREVIATIONS

MEC	Mobile Edge Computing
MARL	Multi-Agent Reinforcement Learning
NOMA	Non-Orthogonal Multiple Access
MBSs	Macro Base Stations
LTE	Long Term Evolution
M2M	Mobile-to-Mobile
D2D	Device-to-Device
CNN	Convolution Neural Networks
IA	Intelligence Artificielle
PDM	Processus de Decision de Markov
PD	Programmation Dynamique
ECS	Edge Computing Server
UD	User Device

RÉSUMÉ

Résumé -
L'informatique mobile en périphérie (MEC) et L'accès multiple non orthogonal sont considérés comme des technologies prometteuses pour répondre aux exigences strictes des réseaux émergents de cinquième génération (5G).Ce travail étudie le problème de l'allocation des ressources dans un système MEC à accès multiple non orthogonal pour plusieurs utilisateurs, en optimisant conjointement la puissance et les ressources de calcul afin d'améliorer le cout totale du système. En raison de la grave non convexité du problème, un schéma décentralisé d'apprentissage par renforcement multi-agents (MARL) est proposé, où chaque lien utilisateur-station de base (U2B) agit comme un agent et interagit collectivement avec l'environnement du réseau, afin de minimiser la fonction objective sous des contraintes limitées de puissance et de ressources de calcul. Les résultats de la simulation démontrent que le schéma MARL proposé entraîne une amélioration considérable du cout totale du système, qui est comparable aux résultats optimaux dérivés de la recherche optimale exhaustive, avec un surcoût significativement faible.

Mots-clés :
Informatique Mobile en Périphérie (MEC), Apprentissage par Renforcement Multi-Agents (MARL).

ABSTRACT

Abstract -
Non-orthogonal multiple access and mobile edge computing (MEC) have gained recognition as promising technologies to address the demanding requirements of future 5G networks. This study focuses on exploring the resource allocation problem in a NOMA-based MEC system with multiple users, aiming to maximize power and computing resources in order to minimize the overall system cost. Due to the highly non-convex nature of the problem, a decentralized multi-agent reinforcement learning (MARL) scheme is introduced. In this scheme, each user-base station link (U2B) serves as an agent and collaboratively interacts with the network environment to minimize the objective function, considering constraints on power and computational resources. Simulation results demonstrate that the proposed MARL scheme significantly improves the overall system cost, comparable to the optimal results obtained through exhaustive optimal search, but with significantly reduced overhead.

Key words :
Mobile Edge Computing (MEC), Multi-Agent Reinforcement Learning (MARL).

INTRODUCTION GÉNÉRALE

La croissance sans précédent des appareils intelligents et l'afflux croissant de nouvelles applications ont entraîné un changement de paradigme dans les réseaux de communication sans fil. Cette croissance explosive va également de pair avec les progrès de la technologie mobile, qui ont permis à plusieurs utilisateurs d'effectuer des tâches intensives à distance sur des appareils mobiles. Cependant, les appareils mobiles ont une puissance de calcul et une capacité de stockage limitées, qui peuvent être jugées inadéquates pour répondre aux exigences de certaines des applications émergentes à forte intensité de calcul [1]. Les services hétérogènes, tels que la réalité augmentée (AR), la réalité virtuelle (VR), le streaming vidéo à haut débit et les jeux interactifs en temps réel, nécessitent une efficacité spectrale et énergétique élevée, ainsi qu'une faible latence et une meilleure équité, etc. Ces services nécessitent des calculs très complexes et énergivores, ce qui limite leur utilisation sur les appareils mobiles en raison de la capacité limitée de la batterie et de la capacité de calcul.

La technologie en nuage est une tendance récente qui peut aider à surmonter ce dilemme [2]. Les appareils mobiles peuvent décharger leurs tâches à forte intensité de calcul en les confiant à des services en nuage. Toutefois, ces services en nuage sont centralisés, ce qui entraîne des retards importants et une congestion du trafic au niveau des serveurs. Pour relever ce défi, les tendances récentes consistent à décharger ces tâches à forte intensité de calcul sur les bords du réseau [3]. Le Mobile Edge Computing (MEC) peut amplifier la capacité de calcul à la périphérie du réseau en déployant des serveurs très performants, réduisant ainsi les délais et améliorant la qualité de service (QoS). Le MEC a trouvé son utilité dans un certain nombre d'applications telles que le délestage de calcul économe en énergie (EECO) [4], pour les réseaux d'accès sans fil à fibre optique (Fi-Wi) [5], ce qui fait du MEC l'un des éléments clés des réseaux 5G [6].

Toutefois, l'augmentation prévue des équipements des utilisateurs et des appareils de l'internet des objets (IdO) à la suite du déploiement prospectif des réseaux de la prochaine génération entraînera un problème inhérent à l'accès multiple dans un scénario de spectre déjà épuisé. En raison de son amélioration considérable de l'efficacité spectrale, l'accès multiple non orthogonal (NOMA) a également été considéré comme une technologie émergente pour les futurs réseaux sans fil. L'accès multiple non orthogonal dessert

simultanément plusieurs utilisateurs dans un bloc de ressources donné, en incorporant un nouveau degré de liberté dans le domaine de la puissance. Afin d'accomplir cela, nous adoptons une approche de codage en superposition (CS) au niveau de l'émetteur, suivi d'une annulation progressive des interférences au niveau du récepteur, ce qui assure un équilibre optimal entre le débit du système et l'équité pour l'utilisateur [7]. Les utilisateurs dont les conditions de canal sont moins bonnes se voient attribuer des puissances plus élevées, tandis que les utilisateurs dont les conditions de canal sont meilleures se voient attribuer des puissances plus faibles afin de garantir l'équité. La capacité de NOMA à s'adapter à d'autres technologies émergentes, telles que MEC, a motivé d'importantes contributions de recherche aux réseaux NOMA-MEC.

Le travail présenté dans ce manuscrit est composé de quatre chapitres soigneusement présentés ci-dessous.

Chapitre 1 : Ce chapitre présente l'architecture hiérarchique de mobile edge computing et les défis de l'informatique mobile de périphérie, ainsi que la recherche sur le déchargement informatique et les défis et orientations futures.

Chapitre 2 : Ce chapitre présente les éléments théoriques et mathématiques nécessaires à la formulation de défis basés sur l'apprentissage par renforcement.

Chapitre 3 : Ce chapitre présente les différents modèles du système, ainsi que le modèle de calcul local et de déchargement. et donne un formulation du problème sous la forme d'un problème d'optimisation.

Chapitre 4 : Ce dernier chapitre clôture notre projet en exposant les résultats définitifs, les décisions prises et les outils utilisés tout au long de notre travail.

Ce travail est finalisé par une **Conclusion générale et perspectives** récapitulant l'ensemble des principaux résultats et les différentes perspectives envisagées.

CHAPITRE 1

MOBILE EDGE COMPUTING

Sommaire

Introduction

L'informatique mobile de périphérie est un concept intéressant qui fournit des ressources informatiques aux utilisateurs mobiles à la périphérie du réseau, ce qui permet aux applications à forte intensité de calcul et sensibles au facteur temps d'être exécutées rapidement par des serveurs situés à la périphérie afin de répondre aux besoins des utilisateurs mobiles. Dans ce chapitre, on commence par présenter une architecture hiérarchique d'informatique mobile en périphérie composée d'un plan de nuage, d'un plan de périphérie et d'un plan d'utilisateur. Ensuite, trois décisions courantes de déchargement de calcul sont présentées. Enfin, nous abordons les recherches les plus récentes sur le déchargement de calcul et présentons une étude de cas sur le déchargement de calcul collaboratif.

1.1 Architecture de Mobile Edge Computing

1.1.1 Architecture hiérarchique de MEC

Pour mieux comprendre la logique interne de la MEC, nous présentons d'abord une architecture hiérarchique qui divise verticalement le système informatique périphérique en trois couches : la couche utilisateur, la couche périphérique et la couche nuage, comme le montre la figure 1.1. La couche utilisateur se distingue par le mode de communication sans fil entre les appareils mobiles et les infrastructures sans fil. Les couches edge et cloud se réfèrent principalement aux ressources informatiques des serveurs edge et cloud, respectivement.

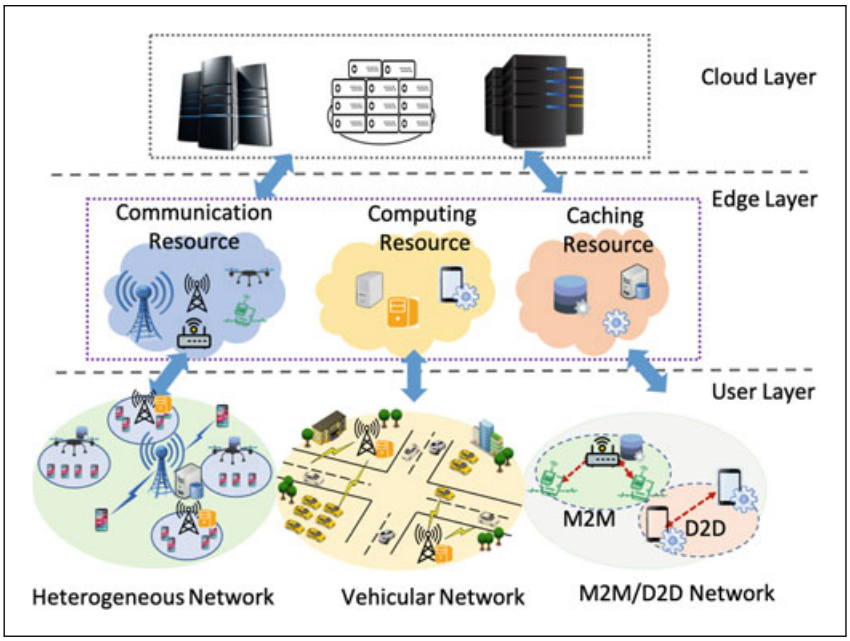

FIGURE 1.1 – Architecture hiérarchique de MEC

Les dispositifs de la couche utilisateur comprennent les capteurs, les smartphones, les véhicules, les compteurs intelligents et les dispositifs d'identification par radiofréquence. Ces appareils accèdent aux serveurs périphériques par le biais de la communication sans fil et déchargent ensuite les tâches à forte intensité de calcul vers les serveurs périphériques légers et distribués pour qu'ils les traitent. En fonction de la topologie du réseau sans fil et des modes de communication, la communication entre les appareils mobiles et une infrastructure sans fil peut être divisée en trois modes.

- Réseau hétérogène : Les réseaux sans fil de la prochaine génération utiliseront des applications qui nécessitent des débits de données élevés. La densification du réseau par le déploiement de petites cellules est une solution qui permet de réduire les exigences en matière de débit de données. Cette densification permet d'améliorer l'efficacité spectrale et de réduire la consommation d'énergie d'un appareil mobile en raison de sa communication avec les petites stations de base cellulaires voisines. Cette solution améliore considérablement la couverture du réseau. Le fonctionnement simultané de stations de base macro (MBS) et de stations de base micro, pico, femto et assistées par des véhicules aériens sans pilote est appelé réseau hétérogène. Dans les réseaux hétérogènes, toutes les stations de base sont équipées de ressources informatiques et de fonctions d'intelligence artificielle. Les appareils mobiles à ressources limitées peuvent décharger leurs tâches sur ces stations de base hétérogènes, qui peuvent alors utiliser une politique d'allocation des ressources informatiques à grain fin pour traiter les tâches déchargées.
- Réseau de véhicules : Les réseaux de véhicules sont indissociables de l'environnement d'une ville intelligente, en raison de plusieurs applications qui améliorent la qualité de vie, la sécurité et la sûreté. Un réseau véhiculaire est formé de véhicules en mouvement, d'unités routières et de piétons, qui peuvent être déployés dans des environnements ruraux, urbains et autoroutiers. La communication de véhicule à véhicule permet aux véhicules de communiquer avec d'autres véhicules et leur environnement via des liaisons sans fil. La communication de véhicule à tout a trois scénarios principaux : de véhicule à véhicule, de véhicule à infrastructure et de véhicule à piéton [8]. Les technologies couramment utilisées sont les communications dédiées à courte distance, l'IEEE 802.11p, la famille de normes IEEE 1609 et l'évolution à long terme (LTE).Avec les progrès des technologies de communication, un certain nombre d'applications prometteuses émergent pour les réseaux de véhicules. Celles-ci vont des applications de sécurité, telles que l'avertissement des angles morts et les infractions aux feux de circulation à des applications de divertissement, comme la diffusion de médias en continu, ou de commodité, comme l'identification d'une place de parking. Dans les réseaux de véhicules, des ressources périphériques omniprésentes peuvent être déployées sur des infrastructures proches afin d'offrir aux véhicules une qualité de service élevée. Par rapport aux n ?uds mobiles courants, les véhicules peuvent se déplacer à des vitesses assez élevées, ce qui entraîne des changements fréquents dans la topologie d'un réseau de véhicules. L'élaboration d'une politique détaillée doit tenir compte de ces topologies de réseau

dynamiques.

- Réseaux de mobile à mobile (M2M) et d'appareil à appareil (D2D) : M2M est une technologie de l'internet des objets, qui implique une connectivité autonome et des réseaux de la communication entre les appareils, qu'il s'agisse de capteurs et d'actionneurs intégrés ou d'appareils puissants à forte capacité de calcul, sans intervention humaine. Le D2D permet aux appareils de communiquer entre eux par une liaison sans fil directe sans passer par la station de base ou le réseau central. Avec les progrès technologiques des appareils intelligents, davantage de ressources de calcul et de mise en cache sont distribuées parmi les utilisateurs finaux. Les tâches informatiques peuvent donc être déchargées non seulement sur les serveurs périphériques, mais aussi sur les appareils des réseaux D2D et M2M.

La couche périphérique est située au milieu de l'architecture hiérarchique et se compose de plusieurs serveurs périphériques distribués qui fournissent aux utilisateurs des services informatiques sans fil intelligents et distribués. Les serveurs périphériques peuvent être déployés dans l'infrastructure du réseau, comme les stations de base, les unités routières, les points d'accès sans fil et les passerelles, ou ils peuvent être des téléphones mobiles, des véhicules, des tablettes et d'autres dispositifs dotés de capacités de calcul et de stockage. En général, les serveurs périphériques sont largement répartis dans des points névralgiques tels que les cafés, les centres commerciaux, les gares routières, les rues et les parcs. Étant donné la proximité des serveurs périphériques avec les utilisateurs finaux, les tâches à forte intensité de calcul et sensibles aux délais peuvent être déchargées et accomplies avec une faible latence et une grande efficacité. Il existe trois types de ressources dans la couche périphérique : les ressources de communication, les ressources de mise en cache et les ressources informatiques. Les ressources de communication se réfèrent à la bande passante, au spectre et à la puissance de transmission. Les ressources informatiques se réfèrent principalement aux cycles de l'unité centrale. Les ressources de mise en cache sont liées à la capacité de mémoire des serveurs périphériques. Étant donné que les serveurs périphériques sont distribués de manière omniprésente, leurs capacités de calcul et de mise en mémoire cache sont généralement limitées. L'utilisation complète des ressources périphériques nécessite l'optimisation conjointe des ressources de communication, de mise en cache et de calcul.

La couche centrale du nuage est constituée de plusieurs serveurs dotés de fortes capacités de traitement, de mise en cache et de calcul. Avec une vision globale, cette couche peut exploiter des techniques avancées telles que l'exploration de données et le big data, pour passer d'un fonctionnement réactif à un fonctionnement proactif du réseau, en prédisant des événements ou en pré-attribuant des ressources. Grâce à leur capacité de calcul élevée et à des ressources de mise en cache suffisantes, les serveurs en nuage peuvent traiter des applications tolérant les retards et stocker des contenus plus volumineux ou moins populaires. En outre, la couche centrale du nuage peut gérer et contrôler efficacement plusieurs serveurs périphériques et leur fournir des connexions sécurisées.

1.1.2 L'accès multiple non orthogonal

L'accès multiple non orthogonal (NOMA) joue un rôle essentiel dans le contexte du Mobile Edge Computing (MEC). Le NOMA est une technique de multiplexage qui permet à plusieurs utilisateurs de partager la même ressource spectrale simultanément, en utilisant différentes combinaisons de puissance et de codage. Cette approche permet une utilisation plus efficace du spectre et améliore les performances du système MEC.

En exploitant le NOMA dans le cadre du MEC, il est possible d'optimiser l'allocation des ressources de calcul et de communication aux utilisateurs. Cette technique permet de partager de manière flexible les ressources disponibles entre les utilisateurs, en fonction de leurs besoins et de leurs exigences. Par exemple, des tâches intensives en calcul peuvent être exécutées sur le serveur MEC tandis que des tâches moins exigeantes peuvent être traitées localement sur les appareils des utilisateurs.

L'utilisation du NOMA dans le MEC présente plusieurs avantages. Tout d'abord, cela permet d'augmenter la capacité du réseau en permettant à plusieurs utilisateurs d'accéder simultanément à la même ressource spectrale. Ensuite, cela améliore l'efficacité énergétique en optimisant l'utilisation des ressources de calcul et de communication. Enfin, cela offre une flexibilité accrue dans l'allocation des ressources, ce qui permet d'adapter dynamiquement les ressources en fonction des besoins changeants des utilisateurs.

L'utilisation de l'accès multiple non orthogonal (NOMA) dans le contexte du Mobile Edge Computing (MEC) présente de nombreux avantages. Cette technique permet d'optimiser l'allocation des ressources, d'améliorer les performances du système et d'augmenter la capacité du réseau. L'intégration du NOMA dans le MEC offre de nouvelles opportunités pour la conception et l'optimisation des systèmes de communication et de calcul, en répondant aux demandes croissantes des applications mobiles et en offrant une expérience utilisateur améliorée.

1.2 les défis de mobile edge computing

Les systèmes MEC, comme toute technologie naissante, présentent un certain nombre de problèmes et défis non résolus. Les gains obtenus grâce à ce paradigme dépendent totalement de la résolution de ces problèmes. Plusieurs de ces difficultés sont énumérées ci-après [10].

1.2.1 délai de communication

Bien que l'informatique locale puisse facilement calculer des estimations déterministes des retards de service en fonction de la puissance de calcul de l'appareil local, une grande partie du retard subi par les appareils sur le réseau local est déterminée par l'environnement probabiliste du réseau. Par exemple, si l'état du canal de l'appareil vers son serveur MEC attribué n'est pas bon, ou si le niveau d'interférence auquel l'appareil est confronté est trop élevé, les retards de communication augmenteront et le processus ne pourra pas se terminer dans le temps requis. Une allocation précise des ressources de

la couche physique (telles que la puissance d'émission et l'allocation des sous-canaux) dans le réseau MEC est donc nécessaire et joue un rôle important dans la minimisation de ce retard.

1.2.2 Optimisation des décisions de déchargement

Le MEC aide les appareils à traiter les tâches en temps opportun, mais le transfert vers des serveurs périphériques n'est pas toujours la meilleure option. Par exemple, si la qualité du canal d'un utilisateur particulier est défavorable, le taux de téléchargement peut être faible, ou le temps requis pour soumettre une tâche au serveur MEC peut dépasser le temps requis pour le traitement local. Dans cette situation, il serait préférable que l'appareil exécute le travail localement ou trouve un meilleur serveur et externalise le travail, ce qui souligne que cela a un impact important sur la latence.

1.2.3 Allocation des ressources de calcul

Malgré le fait que l'appareil a déchargé la tâche dans un délai raisonnable, il doit attendre que le serveur de l'appareil traite la tâche. Les serveurs Edge ont une puissance de traitement limitée par rapport aux serveurs cloud, de sorte que le pourcentage de ressources de traitement allouées à la tâche de chaque utilisateur a un impact significatif sur la latence de traitement. Cela souligne l'importance cruciale d'allouer de manière optimale les ressources informatiques sur les serveurs de périphérie pour garantir une utilisation efficace des ressources et une haute qualité de service pour autant d'appareils disponibles.

1.3 Politique de déchargement

Le problème clé de l'informatique de périphérie est de prendre la décision de Déchargement. D'après la description précédente, les résultats de la décision de déchargement sont soit une exécution locale, soit un déchargement complet, soit un déchargement partiel. En combinant l'exécution locale et le déchargement complet, le problème peut être modélisé comme un problème de déchargement binaire zéro-un. Le Déchargement partiel peut être modélisé comme un problème de prise de décision de Déchargement continu. Tout d'abord, nous présentons la recherche sur le déchargement binaire dans la section suivante.

1.3.1 Déchargement binaire

Le déchargement binaire concerne principalement des tâches de calcul à petite échelle qui nécessitent des ressources informatiques importantes. Ces tâches seront entièrement transférées au serveur périphérique. Le déchargement informatique peut réduire efficacement le délai d'exécution des tâches et économiser la consommation d'énergie des appareils. Lorsque l'appareil ne choisit pas le Déchargement (c'est-à-dire l'exécution locale),

le délai d'achèvement de la tâche n'implique que le temps de calcul de la tâche locale. Lorsque l'appareil choisit le Déchargement, le délai d'achèvement de la tâche comprend trois parties : (1) le temps de transmission sans fil de la tâche de calcul de l'appareil au serveur périphérique, (2) le temps de calcul de la tâche sur le serveur périphérique, et (3) le temps de transmission sans fil du résultat du calcul entre le serveur périphérique et l'appareil. De même, lorsque l'appareil ne décharge pas la tâche, la consommation totale d'énergie nécessaire pour accomplir la tâche ne comprend que la consommation d'énergie pour le calcul de la tâche locale. Si l'appareil se décharge d'une partie de la tâche de calcul, la consommation totale d'énergie se compose de deux parties : la consommation d'énergie de la transmission sans fil de l'appareil au serveur périphérique et la consommation d'énergie du calcul sur le serveur périphérique.

Minimisation du délai d'exécution des tâches

Les auteurs de [11] ont proposé un algorithme de recherche unidimensionnel pour minimiser le délai d'exécution. L'algorithme proposé peut trouver une politique de décision de déchargement optimale basée sur l'état de la mémoire tampon, la puissance de traitement disponible et les informations sur le canal. La décision de déchargement détermine si l'application doit être traitée localement ou sur le serveur de la MEC. Une autre idée visant à minimiser le délai d'exécution a été introduite dans [13]. Par rapport à [11], ces auteurs ont pris en compte les utilisateurs appliquant une mise à l'échelle dynamique de la tension et de la fréquence et ont proposé un algorithme de déchargement de calcul dynamique basé sur l'optimisation de Lyapunov de faible complexité. Cet algorithme permet aux utilisateurs de prendre une décision de déchargement à chaque intervalle de temps et d'allouer simultanément les cycles de l'unité centrale et la puissance de transmission. La méthode proposée peut réduire les temps d'exécution jusqu'à 64 % en déchargeant la tâche de calcul sur le serveur périphérique. Contrairement aux deux travaux axés sur la conception d'algorithmes de déchargement de calcul, les auteurs de [12] ont proposé une architecture de déchargement assistée par la MEC qui permet de déployer une logique de programmation intelligente, à savoir un planificateur de périphérie mobile, au niveau de la MEC sans nécessiter de grandes ressources de calcul au niveau du matériel de l'eNodeB. L'ordonnanceur mobile introduit fonctionne sur l'eNodeB. Un processus d'ordonnancement en deux étapes a été proposé pour minimiser le retard des flux de trafic général dans la liaison descendante LTE via le serveur MEC déployé au niveau de l'eNodeB.

Minimisation de la consommation d'énergie

La décision de décharger les calculs pour minimiser la consommation d'énergie des appareils a été proposée dans [14]. Ces auteurs ont formulé le problème d'optimisation sous la forme d'un processus décisionnel de Markov contraint. Pour résoudre le problème d'optimisation, deux types de stratégies d'allocation des ressources tenant compte à la fois des ressources de calcul et des ressources radio ont été introduits. La première stratégie est basée sur l'apprentissage en ligne, où le réseau s'adapte dynamiquement en

fonction de l'application exécutée sur l'appareil. Des expériences numériques ont montré que la stratégie hors ligne précalculée peut être jusqu'à 50 % plus performante que la stratégie en ligne pour des taux d'arrivée (charges) faibles et moyens. La stratégie hors ligne proposée dans [14] ayant fait ses preuves, les auteurs de [15] ont proposé deux approches supplémentaires de programmation dynamique hors ligne pour minimiser la consommation d'énergie moyenne des appareils.

L'une des approches de programmation dynamique visant à trouver la politique optimale de déchargement de la programmation radio est déterministe, tandis que l'autre est aléatoire. Des expériences numériques ont montré que les deux politiques hors ligne peuvent réduire la consommation d'énergie par rapport aux stratégies de déchargement uniquement et de traitement statique. Les auteurs de [15] ont étendu les travaux de [16] du mono-utilisateur au multi-utilisateur en optimisant conjointement l'allocation des ressources et le déchargement des calculs pour garantir l'équité entre les utilisateurs, une faible consommation d'énergie et des contraintes moyennes de file d'attente/délai. Une autre stratégie de décision de déchargement multi-utilisateurs a été proposée dans [17] pour minimiser la consommation d'énergie du système. Cet article a déterminé trois types d'utilisateurs multiples en fonction du temps et du coût énergétique du processus de calcul de la tâche. Le premier type d'utilisateur peut calculer les tâches sur le serveur MEC. Le deuxième type d'utilisateur calcule la tâche sur l'équipement local. Le troisième type d'utilisateur peut décider de mettre en oeuvre les tâches localement ou de les décharger sur le serveur MEC. Sur la base de la classification des utilisateurs, un algorithme conjoint de déchargement des calculs et d'allocation des ressources radio a été proposé. L'algorithme proposé peut réduire la consommation d'énergie jusqu'à 15 % par rapport au calcul sans déchargement.

Compromis entre la consommation d'énergie et le délai d'exécution

Une décision de déchargement de calcul pour un scénario multi-tâches multi-utilisateurs a été proposée dans [18] afin de trouver un compromis entre la consommation d'énergie et le délai d'exécution. Ces auteurs ont examiné conjointement les décisions de déchargement pour toutes les tâches de chaque utilisateur et le partage des ressources de calcul et de communication entre tous les utilisateurs lorsqu'ils sont en concurrence pour décharger les tâches par le biais d'une liaison sans fil à capacité limitée. Le problème du déchargement des calculs est formulé sous la forme d'un programme quadratique non convexe à contraintes quadratiques. Pour résoudre ce problème, un algorithme efficace en trois étapes a été conçu, qui implique une relaxation semi-définie, une optimisation alternée et un réglage séquentiel. Les résultats numériques ont montré que l'algorithme proposé était plus performant que le traitement purement local, le traitement purement dans le nuage et le traitement hybride local-nuage sans serveur périphérique. Un autre algorithme pour la décision de déchargement de calcul afin de trouver un compromis entre la consommation d'énergie et le délai d'exécution a été proposé dans [19], les auteurs ont proposé une décision de déchargement de calcul pour minimiser à la fois la latence

totale d'exécution des tâches et la consommation totale d'énergie des dispositifs mobiles. Deux cas d'appareils mobiles ont été envisagés : les appareils avec une fréquence d'unité centrale fixe et ceux avec une fréquence d'unité centrale élastique. Dans le scénario de l'unité centrale fixe, un algorithme de programmation linéaire basé sur la relaxation a été proposé pour déterminer la décision optimale d'allocation des tâches. Dans le scénario de l'unité centrale élastique, les auteurs ont d'abord envisagé un algorithme basé sur la recherche exhaustive et ont ensuite utilisé un algorithme de relaxation semi-définie pour trouver la solution quasi optimale.

1.3.2 Déchargement partiel

La littérature citée ci-dessus s'est concentrée sur les stratégies de déchargement binaire. Dans un problème de déchargement binaire, la tâche informatique est considérée comme un tout. Cependant, dans les applications pratiques, les tâches informatiques sont souvent divisées en plusieurs parties [20]. En raison de la nature divisible des tâches informatiques, les appareils peuvent décharger une partie d'une tâche, plutôt que sa totalité, vers le serveur périphérique. Il existe donc deux types de tâches : (1) les tâches qui peuvent être divisées en plusieurs segments discrets qui peuvent tous être transférés au serveur MEC pour exécution et (2) les tâches qui peuvent être divisées en deux parties consécutives, non transférables et transférables, et seule la partie transférable peut être transférée. Nous présentons ensuite les travaux axés sur le déchargement partiel.

Minimisation du délai d'exécution des tâches

Les auteurs de [21] ont étudié un problème d'allocation de ressources visant à minimiser le temps de latence pour un système de déchargement multi-utilisateurs avec déchargement partiel. Ils ont proposé un déchargement par compression partielle en trois étapes. Tout d'abord, chaque appareil compresse une partie des données brutes localement, puis transmet les données compressées au serveur périphérique. Ensuite, l'appareil transmet la partie restante des données brutes au serveur périphérique, qui compresse les données. Enfin, le serveur périphérique combine les deux parties des données compressées dans le centre d'hébergement. Un problème de minimisation de la somme pondérée de la latence pour le déchargement partiel de la compression a été formulé et un algorithme d'allocation optimale des ressources basé sur le sous-gradient a été conçu. Des travaux plus généraux sur le déchargement partiel ont été réalisés dans [22]. Les auteurs ont examiné conjointement un schéma de déchargement partiel et d'allocation des ressources afin de minimiser la latence totale pour un système de déchargement multi-utilisateurs basé sur l'accès multiple par répartition orthogonale de la fréquence. Le schéma proposé détermine tout d'abord la fraction de déchargement optimale pour garantir que le délai de calcul à la périphérie est inférieur au délai d'exécution local. Ensuite, il détermine comment allouer les ressources de communication et de calcul. En outre, les utilisateurs peuvent utiliser pleinement les transmissions multicanaux pour réduire davantage le délai de transmission pour les tâches comportant une grande quantité de données. Les résultats

de la simulation montrent que le schéma proposé permet d'améliorer les performances de 17 % et de 25 % par rapport aux schémas de déchargement aléatoire et complet, respectivement.

Minimisation de la consommation d'énergie

Dans [20], les auteurs ont étudié le déchargement partiel de calcul pour minimiser la consommation d'énergie des appareils en optimisant conjointement la fréquence du cycle du CPU, la puissance de transmission et le ratio de déchargement. Ils ont conçu un algorithme de déchargement partiel de calcul optimal sur le plan énergétique qui a transformé le problème non convexe de minimisation de la consommation d'énergie en un problème convexe basé sur la technique de substitution de variables et a obtenu une solution globalement optimale. Les auteurs ont également analysé les conditions dans lesquelles l'exécution locale est optimale. En analysant l'optimalité du déchargement total, les auteurs ont conclu que le déchargement total ne peut pas être optimal en cas de mise à l'échelle dynamique de la tension du dispositif. Les auteurs de [23] ont proposé un algorithme d'ordonnancement conjoint et de déchargement de calcul pour les applications à plusieurs composants en utilisant une approche de programmation en nombres entiers. La décision optimale de déchargement concerne les composants qui doivent être délestés, ainsi que leur ordre d'ordonnancement. L'algorithme proposé offre un plus grand degré de liberté dans la solution en s'éloignant d'un ordre d'ordonnancement prédéterminé par le compilateur pour les composants, au profit d'un ordre d'ordonnancement plus adapté à la technologie sans fil. Pour certaines structures de graphe de dépendance des composants, l'algorithme proposé peut raccourcir les temps d'exécution par le traitement parallèle des composants appropriés sur les appareils et dans le nuage. Pour minimiser la consommation d'énergie attendue de l'appareil mobile, une politique d'ordonnancement économe en énergie pour l'exécution collaborative de tâches entre l'appareil mobile et un clone dans le nuage a été proposée dans [24]. Les auteurs ont formulé le problème de l'ordonnancement énergétiquement efficace des tâches comme un problème stochastique contraint du plus court chemin sur un graphe acyclique dirigé. Ils ont également considéré trois modèles alternatifs de canaux sans fil stochastiques : le canal d'évanouissement par bloc, le canal stochastique indépendant et identiquement distribué et le canal stochastique markovien. Pour résoudre le problème formulé, les auteurs se sont appuyés sur une politique d'escalade unique et ont conçu un algorithme heuristique pour déterminer la décision d'exécution de la tâche.

Compromis entre la consommation d'énergie et le délai d'exécution

Une décision de déchargement partiel tenant compte d'un compromis entre la consommation d'énergie et le délai d'exécution a été décrite dans [25]. La décision de déchargement tient compte de quatre paramètres : (1) le nombre total de bits à traiter, (2) les cycles CPU du dispositif et du serveur MEC, (3) l'état du canal entre le dispositif et les points d'accès femtocellulaires de desserte, et (4) la consommation d'énergie du dispositif.

Le problème d'allocation conjointe des ressources de communication et de calcul a été formulé sous la forme d'un problème d'optimisation convexe. Les résultats de la simulation ont indiqué que le déchargement partiel pouvait réduire la consommation d'énergie des appareils par rapport au déchargement total, lorsque toutes les tâches de calcul doivent être exécutées sur l'appareil ou au point d'accès femtocellulaire. L'étude de [26] a fourni une analyse théorique plus approfondie du compromis entre la consommation d'énergie et la latence des applications déchargées traitées de manière préliminaire dans [25]. Pour réaliser un déchargement partiel, les auteurs ont considéré des applications orientées vers la partition des données et se sont concentrés sur trois paramètres d'une application : (1) la taille des données, (2) le délai d'achèvement et (3) la taille des données de sortie. Ils ont ensuite formulé un problème d'optimisation conjointe de la radio et des ressources informatiques et ont utilisé une technique d'optimisation numérique convexe unidimensionnelle simple pour le résoudre. Les auteurs ont en outre démontré que la probabilité de déchargement des calculs est plus élevée lorsque la qualité du canal est bonne. Les auteurs de [27] ont étudié le compromis entre la consommation d'énergie et le délai d'exécution pour un scénario multi-utilisateurs. Ils ont formulé un problème de minimisation de la consommation d'énergie avec une contrainte de stabilité de la mémoire tampon de l'application. Un algorithme en ligne basé sur l'optimisation de Lyapunov a été proposé pour décider de la fréquence optimale de l'unité centrale de l'appareil pour l'exécution locale et pour allouer la puissance de transmission et la bande passante lors du transfert de l'application vers un serveur périphérique. Les résultats numériques ont démontré que le déchargement de calcul peut réduire la consommation d'énergie jusqu'à environ 90 % et les délais d'exécution d'environ 98 %.

1.4 Challenges et orientations futures

Il existe un large éventail de défis et d'opportunités pour la recherche future sur le déchargement des calculs. Cependant, la recherche sur les CEM en est encore à ses débuts et de nombreux facteurs critiques ont été négligés par souci de simplicité. Dans cette section, nous soulignons plusieurs défis ouverts et mettons en lumière les orientations possibles de la recherche future.

- Programmation multi-serveurs : La collaboration de plusieurs serveurs MEC permet de gérer conjointement leurs ressources pour desservir simultanément un grand nombre d'appareils mobiles. La coopération des serveurs peut non seulement améliorer l'utilisation des ressources, mais aussi fournir aux utilisateurs mobiles davantage de ressources pour améliorer leur expérience. Toutefois, l'augmentation de la taille du réseau entrave la programmation pratique des serveurs MEC. Un trop grand nombre d'utilisateurs délestés provoquera de graves interférences dans les communications entre utilisateurs et le système devra prendre un grand nombre de décisions de déchargement. Des recherches plus approfondies sont nécessaires pour la programmation multi-serveurs.

- Optimisation multi-ressources : L'architecture des réseaux mobiles en périphérie implique diverses ressources : ressources de calcul, de cache et de communication. L'intégration efficace de ces ressources afin d'obtenir des performances optimales pour tous les utilisateurs et toutes les applications est un véritable défi. La gestion efficace des ressources nécessite la conception d'algorithmes distribués d'optimisation des ressources peu complexes, en tenant compte des contraintes des ressources radio et informatiques et des frais généraux de calcul.
- Mobilité des utilisateurs : La mobilité des utilisateurs est un défi majeur dans les réseaux périphériques mobiles. Étant donné que le mouvement et la trajectoire des utilisateurs fournissent des informations de localisation et de préférences personnelles aux serveurs périphériques, les temps de contact entre les utilisateurs et les serveurs MEC sont dynamiques, ce qui aura une incidence sur la stratégie de déchargement. Par conséquent, il convient de mettre en ?uvre des techniques de gestion de la mobilité à la fois horizontales et verticales pour permettre aux utilisateurs d'accéder en toute transparence aux serveurs périphériques.
- Sécurité : La sécurité est l'une des principales préoccupations des conseillers technologiques en ce qui concerne la sécurisation des déploiements MEC. Le déploiement de serveurs en nuage crée de nouveaux défis en matière de sécurité en raison de l'exploitation des informations relatives aux appareils mobiles. Le rythme d'évolution des solutions de sécurité ne peut pas suivre le rythme des nouveaux défis en matière de sécurité. De nombreux protocoles de sécurité existants supposent une connectivité totale, ce qui n'est pas réaliste dans les réseaux mobiles périphériques, car de nombreuses liaisons sont intermittentes par défaut. D'autre part, dans le cadre de la technologie MEC, Les données de l'utilisateur sont transférées à un serveur MEC qui contrôle l'accès des autres utilisateurs mobiles. Cela pose des problèmes, notamment en matière d'intégrité des données et d'autorisation. Par exemple, les données transférées peuvent être modifiées ou consultées par des utilisateurs malveillants. En outre, les propriétaires et les serveurs de données possèdent des identités et des intérêts commerciaux différents, ce qui rend le scénario plus vulnérable. Par conséquent, une étude scientifique complète est nécessaire pour éviter tout problème de sécurité susceptible d'endommager les systèmes MEC.

1.5 Conclusion

Dans ce chapitre, nous avons examiné en détail l'architecture hiérarchique de l'informatique mobile en périphérie, qui se compose du plan de nuage, du plan de périphérie et du plan d'utilisateur. Cette architecture offre une approche prometteuse pour répondre aux besoins croissants de traitement des données à faible latence et de capacité de calcul dans les environnements mobiles. Nous avons également abordé les défis auxquels est confrontée l'informatique mobile en périphérie.

En particulier, nous nous sommes intéressés à la décision de déchargement, qui est une considération clé dans les environnements de calcul en périphérie. Nous avons examiné les

recherches actuelles sur le déchargement de calcul, en mettant l'accent sur les problèmes liés au déchargement binaire et au déchargement partiel. Ces problèmes soulèvent des questions complexes sur la manière de prendre des décisions optimales pour transférer les tâches de calcul entre le périphérique mobile et les serveurs de périphérie.

En conclusion, ce chapitre a mis en lumière l'architecture hiérarchique de l'informatique mobile en périphérie, les défis associés à cette approche et les recherches actuelles sur la décision de déchargement. Cependant, il reste encore de nombreux défis ouverts à relever et des orientations futures à explorer dans ce domaine. L'optimisation de la décision de déchargement, l'amélioration des performances et de l'efficacité énergétique, ainsi que l'adaptation aux différentes conditions de réseau sont autant de domaines qui nécessitent des efforts de recherche supplémentaires pour faire progresser l'informatique mobile en périphérie.

CHAPITRE 2

APPRENTISSAGE PAR RENFORCEMENT

Sommaire

Introduction

L'apprentissage par renforcement est un domaine de l'intelligence artificielle ; il est apparu comme un outil efficace pour construire des systèmes artificiellement intelligents et résoudre des problèmes de prise de décision séquentielle. L'apprentissage par renforcement a réalisé de nombreuses percées impressionnantes ces dernières années et a pu dépasser le niveau humain dans de nombreux domaines ; il est capable de jouer et de gagner divers jeux. Historiquement, l'apprentissage par renforcement s'est avéré efficace pour résoudre certains problèmes de systèmes de contrôle. Aujourd'hui, ses applications sont de plus en plus nombreuses.

2.1 Apprentissage par renforcement

2.1.1 Apprentissage par renforcement dans l'intelligence artificielle

L'intelligence artificielle (IA) est devenue un sujet brûlant ces dernières années. De nombreux articles, livres et films ont été produits avec des questions telles que "les machines peuvent-elles penser ?", "l'IA peut-elle dépasser l'intelligence humaine ?", "les machines remplaceront-elles les humains ?", "quel est le danger de l'IA ?", "qu'est-ce qui distingue l'humain de l'IA ?".

L'essor de l'intelligence artificielle est associé aux réalisations de l'apprentissage profond (Deep Learning) au cours des dernières années. L'apprentissage profond est essentiellement un ensemble de couches multiples de réseaux neuronaux connectés les uns aux autres. Bien que les algorithmes d'apprentissage profond soient les mêmes que ceux utilisés à la fin des années 1980, les progrès de l'apprentissage profond sont dus au développement de la puissance de calcul et à l'augmentation considérable des données générées et collectées. Le passage du CPU (Central Processing Unit) au GPU (Graphics Processing Unit) et plus tard au TPU (Tensor Processing Unit) a accéléré la vitesse de traitement et ouvert la voie à davantage de succès. Toutefois, les capacités de calcul sont limitées par la loi de Moore [28], ce qui peut ralentir la construction de systèmes d'IA puissants.

L'apprentissage par renforcement consiste à apprendre en interagissant avec un environnement en prenant différentes mesures et en expérimentant de nombreux échecs et succès tout en essayant de maximiser les récompenses reçues. On ne dit pas à l'agent quelle action il doit entreprendre. L'apprentissage par renforcement est similaire aux processus d'apprentissage naturels où il n'y a pas d'enseignant ou de superviseur et où le processus d'apprentissage évolue par essais et erreurs, à la différence de l'apprentissage supervisé, dans lequel un agent doit être informé de la bonne action à adopter pour chaque situation qu'il rencontre.

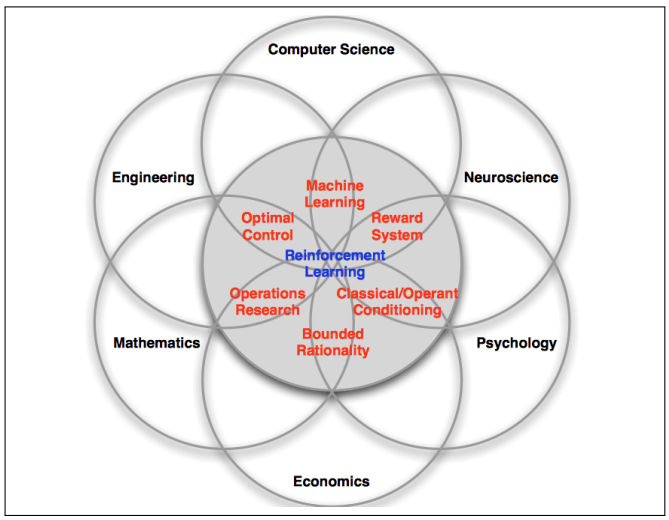

FIGURE 2.1 – facettes de l'apprentissage par renforcement

L'apprentissage par renforcement recoupe plusieurs domaines : l'informatique, l'ingénierie, les neurosciences, les mathématiques, la psychologie et l'économie. La figure 1 illustre ces recoupements [29]. L'apprentissage par renforcement est différent des autres branches de l'apprentissage automatique, qu'il s'agisse d'apprentissage supervisé ou non supervisé.

L'apprentissage supervisé est la branche de l'apprentissage automatique la plus étudiée et la plus recherchée. Dans l'apprentissage supervisé, la machine apprend à partir d'un ensemble de données étiquetées fournies par un enseignant ou un superviseur externe qui détermine les actions correctes que le système doit prendre pour chaque exemple. La tâche du système consiste à généraliser ses réponses pour agir correctement dans des cas qui ne figurent pas dans les exemples de formation. Les performances du système d'apprentissage supervisé s'améliorent en augmentant le nombre d'exemples de formation. Voici quelques exemples de problèmes d'apprentissage supervisé : la classification, la détection d'objets, le sous-titrage d'images, la régression et l'étiquetage. Bien que ce type d'apprentissage soit important, il n'est pas adapté aux environnements interactifs car il n'est pas réaliste d'obtenir des données étiquetées qui soient à la fois représentatives et correctes. Dans les environnements interactifs, l'apprentissage sera plus efficace si le système peut apprendre de sa propre expérience.

L'apprentissage non supervisé consiste à trouver une structure cachée dans un ensemble de données non étiquetées. Voici quelques exemples d'apprentissage non supervisé : Le regroupement, l'apprentissage des caractéristiques, la réduction de la dimensionnalité et l'estimation de la densité. Même si l'apprentissage par renforcement peut sembler être un type d'apprentissage non supervisé puisqu'il n'apprend pas à partir de données étiquetées, il est différent ; l'apprentissage par renforcement tente de maximiser les récompenses plutôt que de trouver une structure cachée.
L'apprentissage par renforcement est considéré comme un troisième paradigme de l'ap-

prentissage automatique, à côté de l'apprentissage non supervisé et de l'apprentissage supervisé.

2.1.2 Modèle standard d'apprentissage par renforcement

Les principaux composants d'un système d'apprentissage par renforcement sont : la politique, le signal de récompense, la fonction de valeur et le modèle.

- La politique (Π) est la manière dont l'agent (quelque chose qui perçoit et agit dans un environnement) se comportera dans certaines circonstances. Simplement, la politique traduit les états en actions. Il peut s'agir d'une table de recherche, d'une fonction ou d'un processus de recherche. La recherche de la politique optimale est l'objectif principal du processus d'apprentissage par renforcement.
- Le signal de récompense (R) indique dans quelle mesure un événement est bon ou mauvais et définit l'objectif du problème, l'objectif de l'agent étant de maximiser la récompense totale reçue. Par conséquent, la récompense est le principal facteur de mise à jour de la politique. La récompense peut être immédiate ou différée. Pour les signaux différés, l'agent doit déterminer quelles actions sont plus pertinentes pour une récompense différée.
- La fonction de valeur est une prédiction des récompenses totales futures, elle est utilisée pour évaluer les états et sélectionner les actions en conséquence.

La fonction état-valeur V(s) est le rendement attendu à partir d'un état s.

$$V(s) = E(G_t|S_t = s) \tag{2.1}$$

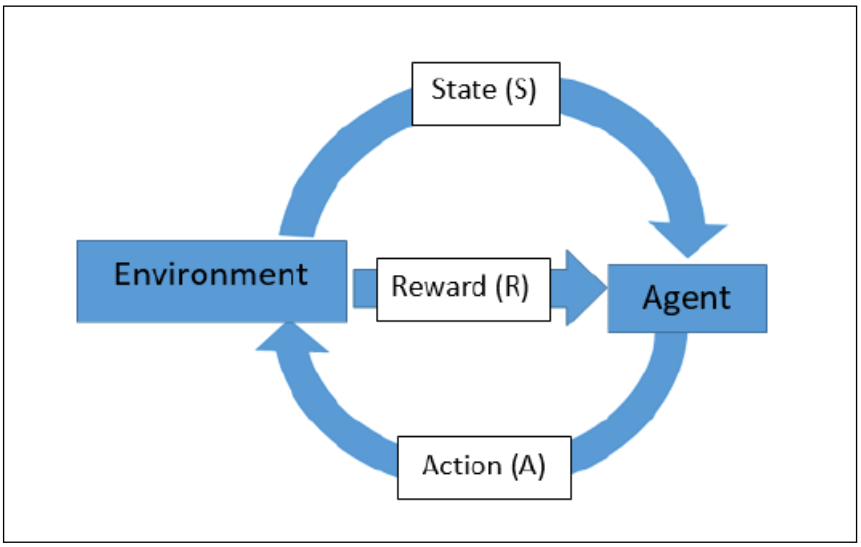

FIGURE 2.2 – Diagramme standard de l'apprentissage par renforcement

La performance G_t correspond à la somme de la récompense immédiate et de la récompense future actualisée R à partir de l'instant t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{n-1} R_{t+n} \tag{2.2}$$

L'escompte γ représente le facteur de dégradation des bénéfices futurs lorsqu'ils sont évalués à l'heure actuelle, γ étant compris entre 0 et 1. Toutefois, l'utilisation de l'escompte est parfois controversée.

La fonction de valeur d'action q(s,a) est le rendement attendu lorsque l'on part d'un état s et que l'on entreprend une action a. L'équation 2.3 présente la formulation mathématique.

$$\begin{aligned} q(s,a) &= E(G_t|S_t = s, A_t = a) \\ &= E(R_{t+1} + \sum_{m=1}^{\infty} \gamma^m R_{t+m+1}|S_t = s, A_t = a) \end{aligned} \tag{2.3}$$

Le modèle de l'environnement permet de faire des prédictions sur le comportement de l'environnement. Cependant, le modèle est un élément facultatif de l'apprentissage par renforcement ; les méthodes qui utilisent des modèles et la planification sont appelées méthodes basées sur un modèle. D'un autre côté, il y a les méthodes sans modèle où l'agent n'a pas de modèle pour l'environnement. Les méthodes sans modèle sont explicitement des méthodes d'apprentissage par essais et erreurs.

2.2 Processus de Décision de Markov

2.2.1 Définition

Le problème du processus de décision de Markov consiste en des décisions séquentielles dans lesquelles une action (A) doit être entreprise dans chaque état (S) visité par l'agent. Le processus de décision de Markov peut être représenté comme une séquence d'états, d'actions et de récompenses, comme le montre la ligne suivante.

$$...S_t, A_t, R_t, S_{t+1}, A_{t+1}, R_{t+1}, S_{t+2}, A_{t+2}, R_{t+2}...$$

La principale propriété du processus de Markov est que le futur est indépendant du passé compte tenu du présent. L'équation 4 formule l'état de Markov comme suit : la probabilité de l'état suivant S_{t+1} ne dépend que de l'état actuel S_t , quels que soient les états passés $S_{t-1}, S_{t-2}, S_2, S_1$.

$$Prob(S_{t+1}|S_t) = Prob(S_{t+1}|S_t, S_{t-1}, ..., S_2, S_1) \tag{2.4}$$

2.2.2 Équation d'optimalité de Bellman

Les PDM sont bien étudiés dans la théorie du contrôle, et leurs bases remontent aux travaux pionniers de Richard Bellman. La principale contribution de Bellman a été de montrer que la résolution des PDM à l'aide de la programmation dynamique (PD) réduit efficacement les charges de calcul.
L'objectif d'un agent est de prendre des mesures qui maximisent les récompenses totales. Il s'agit essentiellement d'un problème d'optimalité dans lequel l'agent d'apprentissage par renforcement tente de prendre des mesures qui conduisent à des récompenses maximales, qui peuvent être représentées par la fonction de valeur d'action q(s, a).
la fonction de valeur d'action optimale $q_*(s, a)$ est la fonction de valeur d'action maximale pour toutes les politiques

$$q_*(s, a) = max_\pi(q_\pi(s, a)) \tag{2.5}$$

L'équation d'optimalité de Bellman (équation 2.7) tire parti de la forme récursive de la fonction de valeur d'action q(s, a) qui peut être réécrite sous la forme récursive suivante (équation 2.6)

$$q(s, a) = R(s, a) + \gamma q(s', a') \tag{2.6}$$

Équation d'optimalité de Bellman :

$$q_*(s, a) = R(s, a) + \gamma \sum_{s'} T(s, a, s') max_{a'} q_\pi(s', a') \tag{2.7}$$

Où :

R(s,a) est la récompense immédiate compte tenu de l'état actuel s et de l'action actuelle a

$$R(s, a) = E(R_{t+1}|S_t = s, A_t = a) \tag{2.8}$$

$q_*(s', a')$ est la fonction de valeur de l'action étant donné l'état s' (prochain état possible) et l'action a' (prochaine action possible).

$T(s, a, s')$ est la probabilité d'une transition vers l'état s' étant donné l'état actuel s et l'action actuelle a

$$T(s, a, s') = E(S_{t+1} = s'|S_t = s, A_t = a) \tag{2.9}$$

L'équation d'optimalité de Bellman n'est pas linéaire et, en général, il n'existe pas de solution sous forme fermée. Toutefois, de nombreuses méthodes de résolution itératives ont été rapportées, telles que l'itération de valeur, l'itération de politique, Q-learning et SARSA.

Les méthodes classiques de PD, telles que l'itération de la politique et l'itération de la valeur, peuvent s'effondrer lorsqu'il s'agit de PDM complexes et à grande échelle ; deux

obstacles peuvent entraver l'évolutivité des PDM : 1) la malédiction de la modélisation, qui est la difficulté de calculer les probabilités de transition, et 2) la malédiction de la dimensionnalité, qui survient lorsque la manipulation des éléments du PDM devient un défi. Toutefois, les approximations des fonctions adaptatives et les méthodes basées sur l'apprentissage permettent de trouver une solution quasi optimale pour les PDM à grande échelle.

2.2.3 Dilemme de l'exploration et l'exploitation

Un des dilemmes centraux dans l'apprentissage par renforcement est celui de l'exploration et de l'exploitation. Ce dilemme se pose lorsqu'un agent doit décider s'il explore de nouvelles actions pour découvrir des récompenses potentielles ou s'il exploite les actions connues qui ont déjà donné des résultats positifs.

Lors de la phase d'exploration, l'agent prend des actions aléatoires ou non éprouvées afin de découvrir de nouvelles informations sur l'environnement et de déterminer les récompenses associées à ces actions. L'exploration est cruciale pour découvrir des stratégies optimales inconnues, mais elle comporte le risque de gaspiller des ressources ou de subir des récompenses faibles.

D'un autre côté, lors de la phase d'exploitation, l'agent choisit les actions qui ont été identifiées comme étant les plus récompensantes selon les connaissances acquises jusqu'à présent. L'exploitation permet à l'agent de maximiser les récompenses à court terme en se basant sur les connaissances actuelles, mais il existe le risque de rester bloqué dans des comportements sous-optimaux ou de ne pas découvrir de meilleures stratégies.

Trouver le bon équilibre entre exploration et exploitation est essentiel pour optimiser les performances de l'agent. Une stratégie courante est d'utiliser des méthodes d'exploration au début de l'apprentissage, puis de passer progressivement à l'exploitation à mesure que l'agent acquiert des connaissances plus fiables sur l'environnement.

Cependant, il n'existe pas de solution universelle à ce dilemme, car cela dépend du contexte spécifique de l'application et des contraintes imposées. Certains problèmes peuvent nécessiter une exploration continue pour éviter de se retrouver dans des optimums locaux, tandis que d'autres peuvent exiger une exploitation plus précoce pour maximiser les performances globales.

En résumé, le dilemme de l'exploration et de l'exploitation est un défi clé dans l'apprentissage par renforcement. Trouver le bon équilibre entre explorer de nouvelles actions et exploiter les actions connues est crucial pour maximiser les performances de l'agent. Cela nécessite une stratégie adaptative en fonction de l'évolution des connaissances et des contraintes spécifiques du problème.

2.3 Algorithmes d'Apprentissage par Renforcement

La politique optimale pour l'apprentissage par renforcement peut être trouvée à l'aide de méthodes de solutions exactes ou de méthodes de solutions approximatives (approxi-

mation de la fonction).

2.3.1 Méthodes Tabulaires

Q-learning est un algorithme simple d'apprentissage par renforcement qui apprend le comportement optimal à long terme sans aucun modèle de l'environnement. L'idée qui sous-tend l'algorithme Q-learning est que la valeur actuelle de notre estimation de Q peut améliorer notre solution estimée (bootstrapping).

$$\begin{aligned} \Delta Q(S_t, A_t) &= \alpha\delta \\ &= \alpha(R_{t+1} + \gamma \max_a (Q(S_t, a) - Q(S_t, A_t)) \end{aligned} \tag{2.10}$$

Où α est le taux d'apprentissage et δ est l'erreur de différence temporelle (TD).

Q-learning est hors politique ; il évalue une politique (politique cible) tout en suivant une autre politique (politique de comportement). Cependant, la recherche d'une fonction de valeur d'action optimale q* dans le cadre d'une politique de comportement arbitraire peut être réalisée à l'aide de l'itération de la politique. Q converge vers la fonction de valeur d'action optimale $q_*(s, a)$, et sa politique gourmande converge vers une politique optimale sous réserve d'un choix approprié du taux d'apprentissage au fil du temps.

L'itération de la politique nécessite l'amélioration et l'évaluation de la politique. Cette dernière améliore l'estimation de la fonction de valeur de l'action en minimisant les erreurs de différence temporelle (TD) dans les chemins expérimentés en suivant la politique. Au fur et à mesure que l'estimation se développe, la politique peut normalement être améliorée en sélectionnant des actions gourmandes en fonction de la dernière fonction de valeur Q. Au lieu d'effectuer les étapes précédentes séparément (comme dans l'itération traditionnelle de la politique), le processus peut être accéléré en permettant des étapes entrelacées comme dans l'itération généralisée de la politique [30].

2.3.2 Méthodes de solutions approximatives

Les méthodes tabulaires représentent les fonctions de valeur exactes et les politiques exactes dans des tableaux. À mesure que l'échelle et la complexité de l'environnement augmentent, la puissance de calcul requise s'accroît considérablement. Cependant, les approximations sont un concept puissant qui absorbe le problème des états cachés et le problème de l'extensibilité.

Une fonction de valeur d'action est représentée par un approximateur de fonction paramétrée $\widehat{q}(s, a, \theta)$ avec le paramètre θ. L'approximateur peut être une pondération linéaire des caractéristiques ou un réseau neuronal profond avec des poids comme paramètre θ.

Le paramètre de la fonction approximative θ peut être mis à jour à l'aide de la mise à jour SARSA semi-gradient présentée dans l'équation 2.11

$$\Delta\theta_t = \alpha(R_{t+1} + \gamma\widehat{q}(S_{t+1}, A_{t+1}, \theta_t) - \widehat{q}(S_t, A_t, \theta_t))\frac{\delta\bar{q}(S_{t+1}, A_{t+1}, \theta_t)}{\delta\theta} \tag{2.11}$$

SARSA est une méthode "on-policy" dans laquelle la politique de comportement est la même que la politique cible ; elle évalue et suit une seule politique. L'algorithme SARSA approxime $\widehat{q}(S_t, A_t$ et par conséquent, la politique ne doit pas être totalement gourmande ; elle doit être proche de la gourmandise $(\epsilon - greedy)$ lorsque la politique agit de manière gourmande la plupart du temps mais qu'il existe une faible probabilité ϵ de choisir des actions aléatoires.

En général, les méthodes "on policy" sont plus performantes que les méthodes "off policy", mais elles trouvent de moins bonnes politiques. Une méthode hors politique trouve une meilleure politique (politique cible) mais ne la suit pas, ce qui se traduit par de moins bonnes performances.

En ce qui concerne les méthodes on-policy avec approximateur de fonction linéaire, de nombreux travaux publiés font état d'une convergence garantie pour les problèmes de prédiction et d'une non-divergence garantie pour les problèmes de contrôle.

L'approximation d'une fonction peut fonctionner correctement avec des méthodes hors politique telles que Q-learning ; une mise à jour de Q-learning avec semi gradient pour les paramètres de la fonction approximative est présentée dans l'équation 2.12.

$$\Delta\theta_t = \alpha(R_{t+1} + \gamma \max_a \widehat{q}(S_{t+1}, a, \theta_t) - \widehat{q}(S_t, A_t, \theta_t))\frac{\delta\bar{q}(S_{t+1}, A_{t+1}, \theta_t)}{\delta\theta_t} \tag{2.12}$$

Cependant, Q-learning avec approximation de fonction peut rencontrer le problème de l'instabilité ; la cause principale de l'instabilité n'est pas l'apprentissage ou l'échantillonnage, puisque la programmation dynamique souffre de divergence avec l'approximation de fonction ; ni l'avidité, ni l'exploration, ni le contrôle ne sont la cause principale, puisque l'évaluation de la politique peut à elle seule produire de l'instabilité ; en outre, la complexité de l'approximation de fonction n'est pas la cause principale, puisque l'approximation de fonction linéaire peut diverger.

La combinaison de l'approximation de fonction, du bootstrapping et de l'apprentissage hors politique présente un risque d'instabilité et de divergence. Cependant, il est difficile d'en choisir deux sur trois : l'approximation de la fonction est importante pour la généralisation et l'évolutivité, le bootstrapping est important pour l'efficacité du calcul et des données, et l'apprentissage hors politique aide à trouver une meilleure politique en libérant la politique comportementale de la politique cible.

Les tentatives rapportées pour atteindre la stabilité et survivre à la triade mortelle

(en combinant l'approximation de fonction, le bootstrapping et l'absence de politique) suggèrent que l'Experience Replay [31] et des cibles plus stables comme le Double Q-learning [32]] peuvent aider. L'utilisation de méthodes des moindres carrés telles que la méthode LSTD(λ) hors politique permet de survivre facilement à la triade. Cependant, leurs coûts de calcul augmentent avec le carré du nombre de paramètres. Les méthodes de vrai gradient telles que Gradient-TD et proximal-gradient-TD s'appuient sur des propriétés de convergence robustes de la descente de gradient stochastique.

2.3.3 Méthode d'apprentissage par Différence Temporelle et Méthode de Monte Carlo

La méthode d'apprentissage par différence temporelle et la méthode de monte carlo ne s'appuient sur aucun modèle et n'ont aucune connaissance des transitions ou des récompenses des PDM. Elles apprennent directement à partir d'épisodes d'expérience, c'est-à-dire que le rendement du MC peut être estimé en faisant la moyenne du rendement de plusieurs déploiements. Contrairement à la méthode de Monte Carlo qui apprend à partir d'épisodes complets par échantillonnage, la méthode de la différence temporelle apprend à partir d'épisodes incomplets par échantillonnage et bootstrap. Toutefois, il est possible d'obtenir le meilleur de la méthode Monte Carlo et de l'apprentissage TD, comme dans l'algorithme TD(λ) où λ interpole entre la différence temporelle ($\lambda = 0$) et le MC ($\lambda = 1$).

2.3.4 Apprentissage par Renforcement basé sur une politique

La méthode d'apprentissage par renforcement basée sur les politiques recherche directement une politique optimale sans fonction de valeur ; elle est donc efficace dans les espaces continus et de grande dimension et présente de meilleures propriétés de convergence, mais elle converge généralement vers un optimum local plutôt que vers un optimum global. Bien qu'elle puisse apprendre des politiques stochastiques, l'évaluation des politiques présente une variance élevée.

La méthode basée sur la politique est un problème d'optimisation dans lequel une politique paramétrée est mise à jour pour maximiser le rendement à l'aide de techniques d'optimisation sans gradient ou basées sur le gradient.

Les méthodes de critique des acteurs mettent en oeuvre une itération généralisée des politiques, alternant entre l'amélioration et l'évaluation des politiques. Une politique est appelée acteur parce qu'elle sélectionne des actions, et une fonction de valeur estimée est appelée critique parce qu'elle introduit des critiques dans les actions de l'acteur. Les méthodes de critique d'acteur compensent la réduction de la variance des gradients de politique par l'introduction d'un biais provenant des méthodes de fonction de valeur [33].

2.4 Q-learning

Q-learning est un algorithme sans modèle, basé sur la valeur et hors politique, qui trouve la meilleure série d'actions en fonction de l'état actuel de l'agent. Le "Q" signifie qualité. La qualité représente la valeur de l'action pour maximiser les récompenses futures.

Les algorithmes basés sur un modèle utilisent des fonctions de transition et de récompense pour estimer la politique optimale et créer le modèle. En revanche, les algorithmes sans modèle apprennent les conséquences de leurs actions par l'expérience, sans transition ni fonction de récompense.
La méthode basée sur la valeur entraîne la fonction de valeur pour apprendre quel état a le plus de valeur et prendre des mesures. D'autre part, les méthodes basées sur la politique entraînent directement la politique pour apprendre quelle action entreprendre dans un état donné.
Dans le cas hors politique, l'algorithme évalue et met à jour une politique qui diffère de celle utilisée pour prendre une mesure. Inversement, l'algorithme "on-policy" évalue et améliore la même politique que celle utilisée pour entreprendre une action.

2.4.1 Q-Table

L'agent utilisera une Q-Table pour prendre la meilleure action possible en fonction de la récompense attendue pour chaque état de l'environnement. En d'autres termes, une Q-Table est une structure de données composée d'ensembles d'actions et d'états, et nous utilisons l'algorithme d'apprentissage Q pour mettre à jour les valeurs de la table.

2.4.2 Q-Function

La Q-fonction utilise l'équation de Bellman et prend l'état(s) et l'action(a) en entrée. L'équation simplifie le calcul des valeurs d'état et d'action.

$$Q^{\pi}(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{n-1} R_{t+n}][s_t, a_t] \tag{2.13}$$

2.4.3 Les étapes de l'Algorithme Q-learning

- **Initialisation de Q-table :**

Nous allons d'abord initialiser la table Q. Nous construirons le tableau avec des colonnes basées sur le nombre d'actions et des lignes basées sur le nombre d'états.

- **Choisir une action :**

La deuxième étape est assez simple. Au départ, l'agent choisira une action aléatoire, et lors de la deuxième exécution, il utilisera une table Q mise à jour pour sélectionner l'action.

- **Effectuer une action :**

Le choix d'une action et son exécution se répètent plusieurs fois jusqu'à ce que la boucle d'apprentissage s'arrête. La première action et le premier état sont sélectionnés à l'aide de la table Q.
Ensuite, l'agent met à jour la table Q à l'aide de l'équation de Bellman. À chaque déplacement, nous mettons à jour les valeurs de la table Q et nous l'utilisons également pour déterminer le meilleur plan d'action.
Au départ, l'agent est en mode exploration et choisit une action aléatoire pour explorer l'environnement. La stratégie cupide d'Epsilon est une méthode simple pour équilibrer l'exploration et l'exploitation. L'epsilon représente la probabilité de choisir d'explorer et d'exploiter lorsque les chances d'explorer sont moindres.
Au début, le taux d'epsilon est plus élevé, ce qui signifie que l'agent est en mode exploration. Pendant l'exploration de l'environnement, l'epsilon diminue et les agents commencent à exploiter l'environnement. Au cours de l'exploration, à chaque itération, l'agent devient plus confiant dans l'estimation des valeurs Q.

- **Mesurer les récompenses :**

Après l'action, nous mesurons le résultat et la récompense.

- **Mise à jour de la table Q :**

Nous mettrons à jour la fonction $Q(S_t, A_t)$ à l'aide de l'équation. Elle utilise les valeurs Q estimées de l'épisode précédent, le taux d'apprentissage et l'erreur de différence temporelle. L'erreur de différence temporelle est calculée à l'aide de la récompense immédiate, de la récompense future maximale escomptée actualisée et de la valeur Q estimée précédente.

Le processus est répété plusieurs fois jusqu'à ce que la table Q soit mise à jour et que la fonction de valeur Q soit maximisée.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{2.14}$$

Au début, l'agent explore l'environnement pour mettre à jour la table Q. Lorsque la table Q est prête, l'agent commence à explorer et à prendre de meilleures décisions. Lorsque la table Q est prête, l'agent commence à exploiter l'environnement et à prendre de meilleures décisions.

2.5 l'apprentissage par renforcement profond

2.5.1 Pourquoi l'apprentissage par renforcement profond

Le Q-learning est un algorithme simple mais très puissant qui permet de créer une antisèche pour notre agent. Cela permet à l'agent de savoir exactement quelle action effectuer.

Mais que se passe-t-il si cette antisèche est trop longue ? Imaginons un environnement comportant 10 000 états et 1 000 actions par état. Cela créerait un tableau de 10 millions de cellules. Les choses deviendraient rapidement incontrôlables !

Il est évident que nous ne pouvons pas déduire la valeur Q de nouveaux états à partir d'états déjà explorés. Cela pose deux problèmes :
-Premièrement, la quantité de mémoire nécessaire pour sauvegarder et mettre à jour ce tableau augmenterait avec le nombre d'états.
-Deuxièmement, le temps nécessaire à l'exploration de chaque état pour créer la table Q requise serait irréaliste.

Voici une idée : et si nous faisions une approximation de ces valeurs Q avec des modèles d'apprentissage automatique tels qu'un réseau neuronal ? C'est l'idée à l'origine de l'algorithme de DeepMind, qui a été racheté par Google, qui combine les réseaux neuronaux à convolution (CNN) et Q-Learning de base. Q-Learning utilise une fonction d'approximation à l'aide d'un CNN lorsqu'il devient difficile d'exprimer la fonction de valeur pour chaque état. Q-Learning profond combine deux approches en plus de l'approximation de la valeur à l'aide d'un CNN [34]. L'une d'entre elles est le rejeu d'expérience et l'autre est la technique Q cible. L'approximation de la valeur à l'aide d'un réseau neuronal est très instable et le retour d'expérience la stabilise.

2.5.2 Réseaux de neurones

Un réseau de neurones est un modèle informatique puissant inspiré du fonctionnement du cerveau humain. Il est composé de nombreux neurones interconnectés qui traitent les informations en passant des signaux à travers des connexions pondérées. Grâce à un processus d'apprentissage basé sur des données, le réseau de neurones peut ajuster les poids de ces connexions pour trouver des motifs, des associations et des relations complexes dans les données d'entrée. Cette capacité à extraire des informations significatives à partir de données non structurées en fait un outil précieux dans de nombreux domaines tels que la vision par ordinateur, le traitement du langage naturel et la prédiction de séries temporelles.

L'apprentissage profond, une branche de l'intelligence artificielle, a grandement bénéficié des réseaux de neurones. Les réseaux de neurones profonds, également connus sous le nom de réseaux de neurones à plusieurs couches, sont capables d'apprendre des représentations hiérarchiques de données complexes en utilisant plusieurs couches de neurones. Cela permet de capturer des caractéristiques de plus en plus abstraites et de réaliser des

tâches sophistiquées telles que la reconnaissance d'objets, la génération de contenu et la prise de décision. Grâce à leur capacité à apprendre à partir de grandes quantités de données et à généraliser à de nouvelles situations, les réseaux de neurones sont devenus un outil essentiel dans le domaine de l'intelligence artificielle et continuent de repousser les limites de ce qui est possible dans le traitement et l'analyse des données.

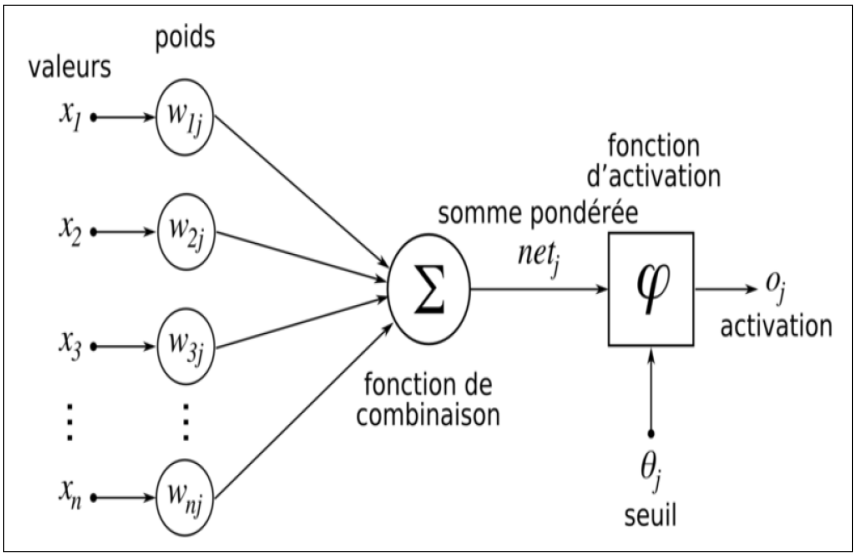

FIGURE 2.3 – Neurone artificiel

Les réseaux neuronaux sont constitués de nombreux noeuds de traitement simples qui sont interconnectés et s'inspirent vaguement du fonctionnement du cerveau humain. Ces noeuds sont généralement disposés en couches et des poids sont attribués aux connexions entre eux. L'objectif est d'apprendre ces poids par le biais de plusieurs itérations de propagation ascendante et descendante des données d'apprentissage à travers le réseau.

Un réseau neuronal comprend des noeuds de traitement disposés en couches. À partir de quelques noeuds et couches, un réseau peut compter des millions de noeuds disposés en milliers de couches.

Nous construisons généralement ces réseaux pour résoudre des problèmes sophistiqués et les classons dans la catégorie de l'apprentissage profond. Lorsque nous appliquons l'apprentissage profond dans le contexte de l'apprentissage par renforcement profond, nous parlons souvent d'apprentissage par renforcement profond.

2.5.3 Deep Q-learning

Dans l'apprentissage profond de la valeur Q, nous utilisons un réseau neuronal pour approximer la fonction de valeur Q. L'état est donné en entrée et la valeur Q de toutes les actions possibles est générée en sortie. La comparaison entre Q-learning et Q-learning approfondi est illustrée ci-dessous :

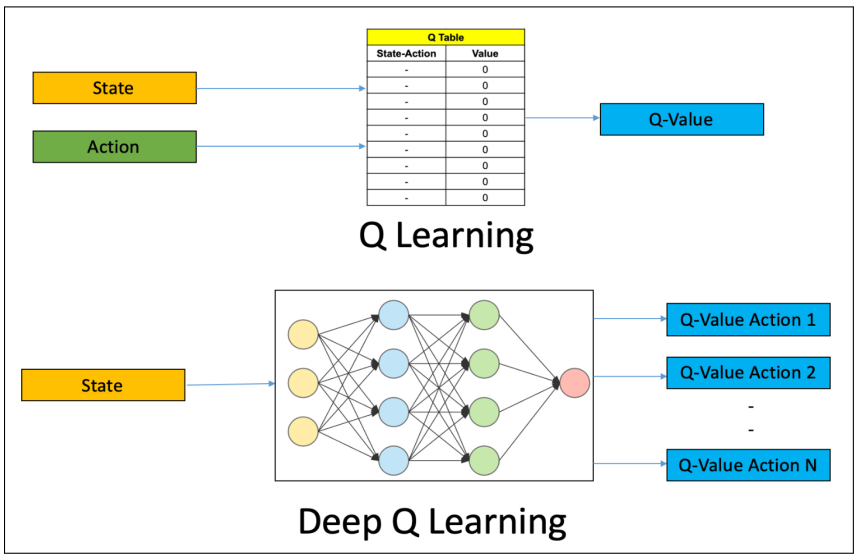

FIGURE 2.4 – modèle de Q-learning et Deep Q-learning

Les étapes de l'apprentissage par renforcement à l'aide deep Q-learning networks (DQN) :

-Toutes les expériences passées sont stockées en mémoire par l'utilisateur.

-La sélection de l'action suivante est basée sur la sortie maximale du réseau Q. La fonction de perte utilisée est l'erreur quadratique moyenne entre les valeurs Q prédites par le réseau et les valeurs cibles.

-La fonction de perte est ici l'erreur quadratique moyenne entre la valeur Q prédite et la valeur Q cible - Q*. Il s'agit essentiellement d'un problème de régression. Cependant, nous ne connaissons pas la valeur cible ou réelle, car il s'agit d'un problème d'apprentissage par renforcement.

2.5.4 Double Deep Q-learning

L'algorithme Double Deep Q-Learning (DDQN) est une extension du Q-Learning traditionnel qui vise à améliorer la stabilité et la convergence de l'apprentissage par renforcement profond. Il a été proposé pour surmonter les problèmes de surestimation des valeurs d'action rencontrés dans le Q-Learning classique. les étapes sont de l'algorithme Double Deep Q-Learning,

-Initialisation du réseau neuronal : Un réseau neuronal profond est initialisé avec des poids aléatoires. Ce réseau est souvent appelé le réseau principal ou le réseau Q.

-Initialisation de la mémoire de relecture : Une mémoire de relecture (replay memory) est utilisée pour stocker les transitions d'expérience passées. Chaque transition est composée d'un état, d'une action, d'une récompense, d'un nouvel état et d'un indicateur de terminaison. Cela permet de créer un ensemble de données pour l'apprentissage hors politique.

-Choix d'une action : L'agent choisit une action à prendre dans l'état actuel en utilisant une stratégie d'exploration/exploitation. Par exemple, on peut utiliser l'epsilon-greedy pour sélectionner une action avec la probabilité epsilon ou choisir l'action optimale en utilisant le réseau Q.

-Exécution de l'action et observation du nouvel état et de la récompense : L'agent exécute l'action choisie et observe le nouvel état résultant de cette action, ainsi que la récompense associée à cette transition d'état.

-Stockage de la transition : La transition (état, action, récompense, nouvel état, indicateur de terminaison) est stockée dans la mémoire de relecture.

-Échantillonnage de mini-lots d'expérience : À partir de la mémoire de relecture, un mini-lot d'expériences est échantillonné de manière aléatoire. Cela permet de décorréler les transitions et d'éviter les problèmes de corrélation temporelle.

-Calcul de la cible Q : Pour chaque transition de l'échantillon, deux étapes distinctes sont effectuées :
a. Estimation de l'action optimale : Le réseau Q principal est utilisé pour sélectionner l'action optimale pour le nouvel état en utilisant l'étape de maximisation de Q. Cependant, dans DDQN, on utilise également un autre réseau appelé le réseau Q cible pour évaluer la valeur Q de l'action choisie par le réseau Q principal.
b. Calcul de la cible Q : En utilisant le réseau Q cible, on calcule la valeur cible Q en utilisant la formule :
Qtarget = récompense + facteur d'escompte * Qcible(nouvel état, argmax(Qprincipal(nouvel état)))
Cela permet de réduire la surestimation des valeurs d'action dans l'estimation de Q.

-Entraînement du réseau principal : Le réseau Q principal est entraîné en minimisant la différence quadratique moyenne (MSE) entre la valeur Q prédite pour chaque transition et la cible Q calculée à l'étape précédente.

-Mise à jour du réseau cible : Périodiquement, les poids du réseau Q cible sont mis à jour en copiant les poids du réseau Q principal.

2.5.5 Replay Memory

La mémoire de relecture, également connue sous le nom de "replay memory" en anglais, est un concept clé dans l'apprentissage par renforcement lorsqu'il est combiné avec des réseaux de neurones. Il s'agit d'une technique utilisée pour stocker et réutiliser les expériences passées de l'agent, afin d'améliorer l'apprentissage et la stabilité des algorithmes d'apprentissage par renforcement basés sur des réseaux de neurones.

La mémoire de relecture permet à l'agent de mémoriser un ensemble d'expériences passées, qui sont généralement composées de l'état, de l'action, de la récompense et de l'état suivant. Ces expériences sont enregistrées dans une mémoire tampon, également appelée "buffer", sous forme de tuples.

Lors de l'apprentissage, l'agent peut alors échantillonner aléatoirement des mini-lots d'expériences de la mémoire de relecture pour entraîner le réseau de neurones. Cette technique présente plusieurs avantages. Tout d'abord, elle permet de briser la corrélation

temporelle entre les expériences, ce qui rend l'apprentissage plus stable. De plus, elle permet de réutiliser efficacement les données d'apprentissage, en évitant le gaspillage de l'information contenue dans les expériences passées.

L'utilisation de la mémoire de relecture avec des réseaux de neurones est particulièrement pertinente dans les algorithmes d'apprentissage par renforcement profond, tels que le Deep Q-Network (DQN) et Double Deep Q-Network (DDQN). Dans ces algorithmes, les réseaux de neurones sont utilisés pour estimer les valeurs d'action ou les politiques d'apprentissage de l'agent.

La mémoire de relecture est une technique utilisée dans l'apprentissage par renforcement pour stocker et réutiliser les expériences passées de l'agent. Elle contribue à l'amélioration de l'apprentissage et de la stabilité des algorithmes basés sur des réseaux de neurones. En échantillonnant aléatoirement des mini-lots d'expériences de la mémoire de relecture, l'agent peut entraîner efficacement le réseau de neurones et améliorer ses performances.

2.6 Conclusion

Dans ce chapitre, nous avons abordé les éléments théoriques et mathématiques essentiels à la compréhension et à la formulation des défis basés sur l'apprentissage par renforcement. Ces éléments constituent une base solide et fondamentale pour la réalisation de notre projet.

Nous avons examiné les principes fondamentaux de l'apprentissage par renforcement, et nous avons exploré différentes approches et algorithmes utilisés dans l'apprentissage par renforcement. Ces méthodes offrent des outils et des techniques pour modéliser et résoudre des problèmes complexes basés sur l'apprentissage par renforcement.

En résumé, ce chapitre a jeté les bases théoriques et mathématiques nécessaires à la formulation des défis d'apprentissage par renforcement. La compréhension de ces éléments est essentielle pour la mise en ?uvre réussie de notre projet, car ils nous fournissent les outils conceptuels et les méthodes nécessaires pour développer et entraîner des agents capables d'apprendre et d'optimiser leur comportement en fonction des récompenses attendues.

CHAPITRE 3

MINIMIZATION DE DÉLAI ET ÉNERGIE DANS UN SYSTÉME MULTI-UTILISATEURS EN MEC

Sommaire

Introduction

Dans un réseau IoT basé sur l'informatique périphérique, le déchargement des tâches est un moyen important de résoudre les limites des ressources informatiques, de stockage et d'énergie des dispositifs périphériques. L'appareil périphérique peut décharger une partie ou la totalité des tâches de calcul vers le serveur informatique périphérique, ce qui accélère la vitesse de traitement des tâches, économise l'énergie de l'appareil et réduit le temps de réponse. Le calcul de la décharge entraînera des frais généraux de communication supplémentaires, tels que le délai de transmission et la consommation d'énergie due à la communication. De nombreuses recherches sont consacrées à la stratégie optimale de déchargement pour différents scénarios et différents objectifs d'optimisation.

3.1 Description du problème et modélisation

3.1.1 Modèle de système

Dans un scénario multi-utilisateurs à cellule unique, plusieurs utilisateurs N=1,2,3,...,N communiquent sans fil par l'intermédiaire d'une seule macro-station de base Long Term Evolution (LTE). La station de base peut être divisée en plusieurs sous-canaux pour différents utilisateurs. Parallèlement, la station de base est connectée à plusieurs serveurs périphériques M=1,2,3,...,M afin de fournir aux utilisateurs des ressources informatiques pour les aider à effectuer des tâches informatiques, comme le montre la figure 3.1 [35].

Chaque utilisateur peut être un téléphone mobile, un ordinateur, un dispositif intelligent portable, etc. Dans le même temps, la puissance de calcul et la charge de chaque serveur périphérique sont également différentes. En outre, les tâches informatiques générées par plusieurs équipements d'utilisateurs peuvent également être en concurrence pour des ressources informatiques limitées. Chaque utilisateur étant égoïste, il souhaite décharger ses tâches sur le serveur le plus performant pour le calcul. Cependant, lorsqu'un grand nombre de tâches d'utilisateurs simultanées atteignent un certain serveur périphérique à haute performance, le serveur ne sera pas en mesure d'allouer les ressources informatiques nécessaires à toutes les tâches d'utilisateurs, ce qui affectera l'expérience de l'utilisateur.

Bien que certains serveurs soient peu performants, ils sont relativement inactifs et peuvent fournir des ressources informatiques relativement supérieures, mais ils ne sont pas utilisés, ce qui entraîne un gaspillage de ressources. Par conséquent, lorsque la répartition des tâches est relativement équilibrée, non seulement le taux d'utilisation des ressources de chaque serveur peut être amélioré, mais la qualité globale du service à l'utilisateur peut également être améliorée. Sous les contraintes de ressources limitées, pour des équipements d'utilisateurs hétérogènes et des tâches d'utilisateurs différentes, il est urgent de résoudre le problème de l'allocation raisonnable des ressources informatiques du serveur périphérique à chaque tâche, et de garantir l'expérience de l'utilisateur en équilibrant la charge de chaque serveur périphérique.

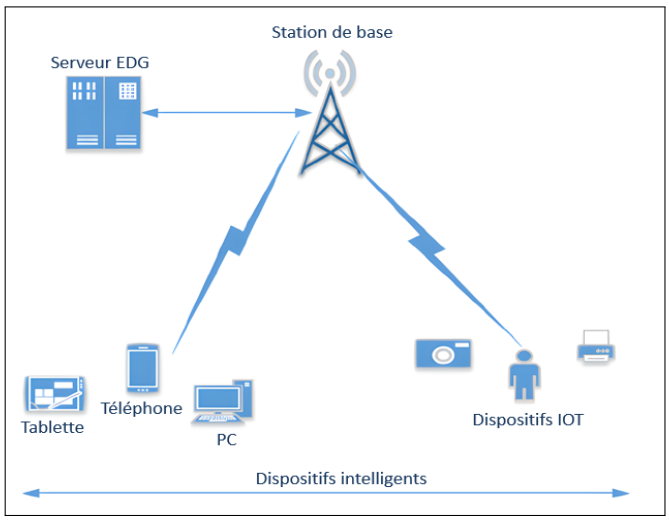

FIGURE 3.1 – Scénario multi-utilisateurs à cellule unique dans mobile edge computing

Nous supposons qu'il y a n=1,2,3,...,N dispositifs d'utilisateur UD (User Device) et une station de base évoluée (eNodeB, eNB) dans une cellule totale. Elle est directement connectée à un serveur informatique de périphérie ECS. On suppose qu'à chaque instant, chaque équipement d'utilisateur génère une tâche à forte intensité de calcul et peut choisir de calculer localement ou de décharger au serveur périphérique directement connecté à la station de base par l'intermédiaire de la station de base.

3.1.1.1 Modèle de communication

Dans le scénario actuel, plusieurs utilisateurs partagent un eNB, de sorte que l'interférence d'intervalle peut être ignorée. En supposant que N utilisateurs choisissent de décharger les tâches de calcul générées en même temps, la largeur de bande du canal sans fil sera distribuée uniformément à l'UD. Le débit de téléchargement disponible pour le nième UD est le suivant

$$r_n = \frac{W}{K} log_2(1 + \frac{P_n h_n}{\frac{W}{K} N_0}) \tag{3.1}$$

Où W est la bande passante du canal sans fil, K est le nombre d'UE de déchargement, P_n est la puissance de téléchargement de l'équipement d'utilisateur n, h_n est le gain du canal sans fil attribué à l'équipement d'utilisateur n, et N_0 est la variance du canal de bruit blanc gaussien complexe.

3.1.1.2 Modèle de tâche

Chaque UE est supposée avoir une tâche à forte intensité de calcul $R_n(B_n, D_n, \tau_n)$, qui peut être exécutée soit localement sur l'unité centrale de l'UE, soit sur le serveur MEC grâce à la décharge de calcul. La taille du calcul, incluant les codes de programme et les paramètres d'entrée nécessaires à R_n, est représentée par B_n. Le nombre total de cycles de l'unité centrale nécessaires pour effectuer la tâche de calcul R_n est représenté par D_n. Il existe une relation directe entre D_n et la taille de B_n. Dn est indicatif de la quantité

Notation	Definition
r_n	le débit de téléchargement
K	le nombre d'UE de déchargement
P_n	la puissance de transmission de l'UE pour le téléchargement des données
h_n	le gain de canal de l'UE dans le canal sans fil
N_0	la variance du bruit gaussien blanc complexe du canal
W	la bande passante du canal sans fil
R_n	la tâche de calcul
f_n^l	la capacité de calcul de l'UE n
B_n	la taille du calcul de la tâche R_n
D_n	la quantité de ressources informatiques nécessaires pour terminer la tâche R_n
τ_n	le retard maximum tolérable de la tâche R_n
α_n	comme la décision de déchargement de calcul de l'UE
f_n	la ressource allouée par le serveur MEC pour accomplir la tâche R_n
F	la ressource totale du serveur MEC
T_n^l	Le délai d'exécution local de la tâche R_n.
E_n^l	La consommation d'énergie associée à la tâche R_n en local
z_n	La consommation d'énergie nécessaire pour accomplir la tâche R_n
T_n^o	Le délai d'exécution de la tâche R_n en déchargement
E_n^o	La consommation d'énergie associée à la tâche R_n en déchargement
I_n^t	la pondération du temps pour le cout total
I_n^e	la pondération énergétique pour le cout total

TABLE 3.1 – Table de notations

de capacité de calcul requise pour accomplir la tâche R_n. Nous faisons l'hypothèse que, que ce soit en exécutant la tâche localement sur l'UE ou sur le serveur MEC, la valeur de D_n reste constante.

τ représente la limite maximale acceptable du retard pour la tâche Rn, ce qui implique que chaque utilisateur doit éviter de dépasser cette limite. Cette contrainte joue un rôle crucial dans notre problème d'optimisation. Les paramètres B_n, D_n et τ_n sont tous liés aux caractéristiques spécifiques des applications et peuvent être estimés à l'aide de profils de tâches. Ces paramètres peuvent varier considérablement selon les différents types d'applications. Nous supposons que la tâche ne peut pas être divisée en sous-parties pour être traitée sur différents dispositifs, ce qui signifie que chaque utilisateur doit exécuter sa tâche localement ou la décharger complètement.

Nous designons $\alpha_n \in \{0, 1\}$ pour représenter le choix de déchargement de l'UE. Le vecteur de décision de déchargement A est défini comme suit : $A = [\alpha_1, \alpha_2, ..., \alpha_n]$. Si $\alpha_n = 0$, cela signifie que l'UE exécute sa tâche localement, tandis que si $\alpha_n = 1$, cela indique que l'UE opte pour le déchargement de sa tâche.

3.1.1.3 Modèle d'exécution local

Si l'UE décide d'exécuter sa tâche R_n localement, nous définissons T_n^l comme le délai d'exécution local de l'UE, qui englobe uniquement le temps de traitement sur les unités centrales locales. Nous désignons également f_n^l comme la capacité de calcul de l'UE n, mesurée en cycles de CPU par seconde. Il est important de noter que les capacités de calcul peuvent varier d'une UE à l'autre. Le délai d'exécution local T_n^l de la tâche R_n est exprimé de la manière suivante,

$$T_n^l = \frac{D_n}{f_n^l} \tag{3.2}$$

Et nous définissons E comme la consommation d'énergie associée à la tâche R_n, exprimée par,

$$E_n^l = z_n D_n \tag{3.3}$$

Où z_n représente la consommation d'énergie par cycle de CPU nécessaire pour accomplir la tâche R_n. Nous fixons $z_n = 10^{-27}(f_n^l)^2$, basé sur des mesures pratiques rapportées dans [36]. En combinant le coût en termes de temps (2) et en termes d'énergie (3), le coût total du déchargement local peut être exprimé de la manière suivante,

$$C_n^l = I_n^t T_n^l + I_n^e E_n^l \tag{3.4}$$

I_n^t et I_n^e représentent les pondérations du temps et du coût énergétique de la tâche R_n. Les pondérations satisfont $0 \leq \mathrm{I} \leq 1$, $I_n^t + I_n^e = 1$ comme les trois notations mentionnées dans le modèle de tâche, les pondérations peuvent varier en fonction du type de tâche. Afin de simplifier le processus de déchargement des calculs, nous considérons que les poids de la tâche R_n ne varient pas pendant toute la durée de déchargement.

3.1.1.4 Modèle de déchargement

Si l'UE opte pour l'exécution de la tâche R_n en déchargeant le calcul, le processus se déroulera en trois étapes distinctes. Dans un premier temps, l'UE doit transférer suffisamment de données d'entrée, y compris les codes de programme et les paramètres, vers l'eNB via le réseau d'accès sans fil. Ensuite, l'eNB transmet ces données au serveur MEC qui, à son tour, alloue une partie de ses ressources informatiques pour exécuter la tâche de calcul pour l'UE. Enfin, le serveur MEC renvoie les résultats de l'exécution de la tâche à l'UE.

D'après les étapes mentionnées précédemment, le temps requis pour la première étape de la décharge du calcul est le temps de transmission des données.

$$T_{n,t}^o = \frac{B_n}{r_n} \tag{3.5}$$

r_n représente le débit de téléchargement réalisable de l'UE dans le canal sans fil mentionné dans le modèle de réseau. la consommation d'énergie correspondante de la

première étape est,

$$E^o_{n,t} = P_n T^o_{n,t} = \frac{P_n B_n}{r_n} \tag{3.6}$$

Pour la deuxième étape du décharnement de l'informatique, le temps requis nécessaire est le délai de traitement du serveur MEC est :

$$T^o_{n,p} = \frac{D_n}{f_n} \tag{3.7}$$

f_n la ressource allouée (les cycles de l' unite centrale par seconde) par le serveur MEC pour accomplir la tâche R_n. et F est la ressource totale du serveur MEC. tq $\sum_{n=1}^{N} \alpha_n f_n \leq F$.

On suppose que l'UE n reste inactif et nous définissons la consommation d'énergie de l'état d'inactivité comme suit as P^i_n, la consommation d'énergie correspondante est,

$$E^o_{n,p} = P^i_n T^o_{n,p} = \frac{P^i_n D_n}{f_n} \tag{3.8}$$

Pour la dernière étape du calcul de déchargement, le temps nécessaire est le délai de téléchargement du résultat traité est,

$$T^o_{n,b} = \frac{B_b}{f_b} \tag{3.9}$$

B_b la taille du résultat traité. et r_b le débit de téléchargement de données de l'UE n. [4], le débit de données lors de cette étape est généralement très élevé, et la taille des données résultantes est souvent considérablement plus petite que celle des données d'entrée. En conséquence, les retards et la consommation d'énergie associés à cette étape sont considérés comme négligeables dans le contexte de cette étude.

d'après les equations (3.5)(3.6), (3.7), (3.8) le délai d'exécution de l'UE n par l'approche de déchargement est,

$$T^o_n = \frac{B_n}{R_n} + \frac{D_n}{f_n} \tag{3.10}$$

Et la consommation dénergie est,

$$E^o_n = \frac{P_n B_n}{R_n} + \frac{P^i_n D_n}{f_n} \tag{3.11}$$

Pour obtenir le coût total du déchargement du calcul, En combinant le coût en temps (3.10) et en énergie (3.11),

$$C^o_n = I^t_n T^o_n + I^e_n E^o_n \tag{3.12}$$

et le coût total de tous les utilisateurs dans le système de déchargement MEC est,

$$C_{all} = \sum_{n=1}^{N} (1 - \alpha_n) C^l_n + \alpha_n C^o_n \tag{3.13}$$

$\alpha_n \in \{0, 1\}$ la décision de déchargement de l'UE. si UE n exécute sa tâche par calcul local $\alpha_n = 0$, sinon $\alpha_n = 1$

3.2 Formulation du problème

le travail consiste à optimiser l'utilisation d'un système de calcul en périphérie de réseau (MEC) en minimisant à la fois le délai et la consommation d'énergie pour tous les utilisateurs du système. Cela doit être fait en respectant la contrainte du délai maximal tolérable et de la capacité de calcul disponible. En d'autres termes, il s'agit d'un problème d'optimisation multi-objectifs où il faut trouver un compromis entre la rapidité de traitement des requêtes et la consommation d'énergie du système. La contrainte de temps maximal tolérable doit également être prise en compte pour garantir que les résultats sont livrés dans les délais requis. Il sera également important de prendre en compte les contraintes du système, telles que la capacité de calcul disponible et les délais maximaux tolérables pour chaque utilisateur. le problème est formulé comme suit,

$$\begin{aligned} &\min_{A,f} \sum_{n=1}^{N} (1-\alpha_n) C_n^l + \alpha_n C_n^o \\ &s.t \\ &C1 : \alpha_n \in 0, 1, \forall n \in N* \\ &C2 : (1-\alpha_n) T_n^l + \alpha_n T_n^o \leq \tau_n, \forall n \in N* \\ &C3 : 0 \leq f_n \leq \alpha_n F, \forall n \in N* \\ &C4 : \sum_{n=1}^{N} \alpha_n f_n \leq F, \forall n \in N* \end{aligned} \tag{3.14}$$

$A = [\alpha_1, \alpha_2, ..., \alpha_n]$: le vecteur de décision de déchargement. et $f = [f_1, f_2, ..., f_n]$: l'allocation des ressources de calcul. en général l'objectif du problème est l'optimisation et le minimisation du coût total de l'ensemble du système, et sous les contraintes,

Contrainte 1 : chaque UE a le choix d'exécuter sa tâche de calcul par le calcul local ou le calcul décharge. $\alpha_n = 0$ si UE n exécute sa tâche par calcul local, sinon $\alpha_n = 1$.

Contrainte 2 : le coût en temps ne doit pas dépasser le délai maximum tolérable, que ce soit au niveau local ou au niveau du serveur.

Contrainte 3 : la ressource de calcul à allouer à l'UE n ne peut dépasser F la ressource totale du serveur MEC.

Contrainte 4 : la somme des ressources informatiques allouées aux UE de déchargement ne peut pas dépasser la ressource totale F du serveur MEC.

Pour résoudre le problème (3.14), il est nécessaire de trouver les valeurs optimales du

vecteur de décision de déchargement A et de l'allocation des ressources informatiques f. Cependant, en raison du caractère binaire de la variable A, l'ensemble réalisable et la fonction objective du problème (3.14) ne sont pas convexes. De plus, la taille du problème peut augmenter très rapidement si le nombre d'utilisateurs augmente, ce qui rend difficile la résolution du problème non convexe étendu à tous les utilisateurs. Ainsi, plutôt que de résoudre le problème (3.14) avec des méthodes d'optimisation conventionnelles, nous proposons d'utiliser des méthodes d'apprentissage par renforcement pour trouver A et f.

3.3 Conclusion

Dans ce chapitre, nous avons exploré les différentes facettes du système en présentant différents modèles, notamment le modèle de système local et le modèle de déchargement. En comprenant ces modèles, nous avons pu appréhender les interactions complexes entre les composants du système et les décisions de déchargement de calcul.

En formulant le problème sous la forme d'un problème d'optimisation, nous avons établi une approche méthodique pour trouver les solutions les plus efficaces et les plus optimales. L'optimisation nous permet de déterminer les décisions de déchargement de calcul qui maximisent les performances du système en termes de temps de calcul, d'utilisation des ressources et d'efficacité énergétique.

En résumé, ce chapitre a présenté les différents modèles du système et a formulé le problème sous la forme d'un problème d'optimisation. Cette approche nous permet d'analyser et de résoudre les défis complexes liés aux décisions de déchargement de calcul de manière systématique et efficace. En développant des stratégies et des algorithmes d'optimisation, nous sommes en mesure de trouver les solutions les plus adaptées pour améliorer les performances et l'efficacité globale du système.

CHAPITRE 4

APPRENTISSAGE PAR RENFORCEMENT POUR ALLOCATION DE RESSOURCES EN MEC

Sommaire

Introduction

Dans cette section, nous présentons les méthodes d'apprentissage par renforcement proposées pour minimiser le délai d'exécution des tâches et la consommation d'énergie du système en déterminant dynamiquement la méthode d'exécution des tâches, soit localement ou par le déchargement au serveur.

4.1 Cadre d'apprentissage par renforcement multi-agents

nous adoptons une approche d'apprentissage par renforcement multi-agents pour parvenir à une solution réalisable pour notre problème d'optimisation. Dans cette section, nous modélisons le problème d'optimisation formulé comme un processus de décision de Markov (PDM), et l'objectif de la sélection des actions est de maximiser la fonction de récompense.

4.1.1 L'espace d'état

L'espace d'état du système est composé de deux éléments $s = (tc, ac)$ [37]. Nous définissons tc comme le coût total de l'ensemble du système $tc = C_{all}$, et ac est la capacité de calcul disponible du serveur MEC, et peut être calculé comme suit $ac = F - \sum_{n=0}^{N} f_n$.

4.1.2 L'espace d'action

Pour L'espace d'action, l'action se compose de deux parties, respectivement la décision de décharger n UE $A = [\alpha_1, \alpha_2, ..., \alpha_n]$ et l'allocation des ressources $f = [f_1, f_2, ..., f_N]$. Le vecteur d'action peut alors être donné comme $[\alpha_1, \alpha_2, ..., \alpha_n, f_1, f_2, ..., f_N]$ avec la combinaison de certaines valeurs possibles de A et de f.

4.1.3 Fonction de récompense

Pour chaque étape, l'agent obtiendra une récompense R(s, a) dans un certain état s après avoir effectué chaque action possible. Par conséquent, l'objectif de notre problème d'optimisation est d'obtenir la somme minimale des coûts et l'objectif de RL est d'obtenir la récompense maximale. La récompense doit être négativement corrélée à l'importance du coût total.

4.1.4 Agents

Dans l'environnement du réseau assisté, chaque liaison utilisateur-station de base (U2B) agit comme un agent et interagit collectivement avec l'environnement du réseau, afin de maximiser la fonction objective sous des contraintes de puissance et de ressources de calcul limitées.

4.2 Apprentissage par renforcement pour la minimisation de l'énergie et délai en MEC

4.2.1 l'algorithme Q-learning

Le Q-learning est un algorithme d'apprentissage par renforcement qui permet à un agent d'apprendre à prendre des décisions dans un environnement pour maximiser les récompenses à long terme. L'agent maintient une table appelée la table Q, qui contient les valeurs de récompense attendues pour chaque paire (état, action). L'algorithme utilise des interactions avec l'environnement pour mettre à jour ces valeurs de manière itérative. L'agent choisit une action en fonction de l'état actuel de l'environnement et de la valeur Q associée à chaque action. Il exécute ensuite cette action, observe le nouvel état et la récompense associée, puis met à jour la valeur Q correspondante en utilisant une formule d'apprentissage. L'algorithme répète ces étapes pour de nombreuses itérations jusqu'à ce que les valeurs Q convergent vers des estimations précises des récompenses à long terme. et $Q(s, a)$ est donné comme suit,

$$Q(s, a) = R(s, a) + \gamma \max Q(s', a') \tag{4.1}$$

De cette manière, nous pouvons obtenir les valeurs optimales A et f pour notre problème d'optimisation.

Algorithm 1 Q-learning

1: Initialiser la table des valeurs Q avec des valeurs arbitraires ou à zéro.
2: Définir le taux d'apprentissage α et le facteur d'actualisation γ.
3: Initialiser l'état initial de l'environnement.
4: **while** non convergence **do**
5: Sélectionner une action a à partir de l'état actuel, en utilisant une politique d'exploration/exploitation.
6: Exécuter l'action a dans l'environnement et observer la récompense r et le nouvel état s'.
7: Mettre à jour la valeur Q de l'état actuel :
8: $Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_a Q(s', a))$
9: Aller à l'état suivant : $s \leftarrow s'$
10: **end while**

4.2.2 l'algorithme Deep Q-learning

L'algorithme Deep Q-learning (DQL) est une méthode d'apprentissage par renforcement qui combine le Q-learning avec des réseaux de neurones profonds. Son objectif est d'apprendre une politique optimale pour prendre des décisions dans des environnements complexes.

Dans le DQL, un réseau de neurones est utilisé pour estimer les valeurs Q, qui représentent la qualité d'une action donnée dans un état spécifique. Le réseau de neurones prend l'état en entrée et renvoie les valeurs Q pour toutes les actions possibles. Au fur et à mesure de l'apprentissage, le réseau de neurones est ajusté pour minimiser la différence entre les valeurs Q prédites et les valeurs Q réelles.

Pour entraîner le réseau de neurones, le DQL utilise la technique de mémoire de relecture. La mémoire de relecture stocke les transitions passées, composées de l'état, de l'action, de la récompense et du nouvel état observés. Lors de l'apprentissage, des échantillons de transitions sont prélevés de la mémoire de relecture pour mettre à jour le réseau de neurones. Cela permet d'apprendre à partir d'expériences passées et d'éviter la corrélation séquentielle des données.

L'algorithme DQL utilise également une politique d'exploration/exploitation pour équilibrer l'exploration de nouvelles actions et l'exploitation des actions connues pour maximiser les récompenses à long terme. Par exemple, une stratégie courante est l'epsilon-greedy, qui choisit une action aléatoire avec une probabilité ? et la meilleure action selon les valeurs Q avec une probabilité $(1 - \epsilon)$.

En résumé, le DQL est un algorithme d'apprentissage par renforcement qui utilise des réseaux de neurones profonds pour estimer les valeurs Q. Il utilise une mémoire de relecture pour entraîner le réseau de neurones à partir d'expériences passées et une politique d'exploration/exploitation pour prendre des décisions dans des environnements complexes.

Algorithm 2 Deep Q-Learning

Initialiser la mémoire de relecture D avec une capacité de N
Initialiser les paramètres du réseau θ
for *épisode* $= 1$ à M **do**
  Réinitialiser l'environnement à l'état initial s
  Initialiser la récompense totale de l'épisode $R = 0$
  while non terminé **do**
    Sélectionner l'action a en utilisant la politique $\pi(s, \theta)$ avec exploration
    Exécuter l'action a, observer la récompense r et l'état suivant s'
    Stocker la transition (s, a, r, s') dans D
    Échantillonner un mini-lot de transitions $(s_j, a_j, r_j, s'j)$ aléatoirement à partir de D
    Calculer les valeurs cibles Q :
      $Q\text{cible}(s_j, a_j) = r_j + \gamma \cdot \max_{a'} Q(s'j, a', \theta)$
    Mettre à jour le réseau en utilisant la descente de gradient :
      $\theta = \theta - \alpha \cdot \nabla\theta \left(Q(s_j, a_j, \theta) - Q_{\text{cible}}(s_j, a_j)\right)^2$
    Mettre à jour l'état $s = s'$
    Mettre à jour la récompense totale de l'épisode $R = R + r$
  end while
end for

Dans cet algorithme, θ représente les paramètres du réseau neuronal, N est la capacité

de la mémoire de relecture, M est le nombre total d'épisodes, ϵ est le taux d'exploration, γ est le facteur de réduction, α est le taux d'apprentissage.

4.2.3 l'algorithme Double Deep Q-learning

L'algorithme Double Deep Q-learning est une extension du Deep Q-learning (DQL) qui vise à améliorer la stabilité de l'apprentissage en utilisant deux réseaux de neurones plutôt qu'un seul. Dans le DDQL, nous avons deux réseaux de neurones, appelés Q et Q'. Le réseau Q est utilisé pour estimer les valeurs Q actuelles, tandis que le réseau Q' est utilisé pour estimer les valeurs Q cibles utilisées pour la mise à jour du réseau Q.

L'idée principale du DDQL est d'utiliser le réseau Q pour sélectionner l'action à prendre dans l'environnement, et ensuite d'utiliser le réseau Q' pour évaluer la valeur de cette action. Cette approche permet de séparer les estimations des valeurs Q de la sélection des actions, évitant ainsi les biais d'estimation qui peuvent se produire lorsque le même réseau est utilisé pour estimer et sélectionner les valeurs Q.

Algorithm 3 Double Deep Q-Learning

Initialiser la mémoire de relecture D avec une capacité de N
Initialiser les paramètres du réseau principal θ
Initialiser les paramètres du réseau cible $\theta' = \theta$
Initialiser le taux d'exploration ϵ
for $épisode = 1$ à M **do**
  Réinitialiser l'environnement à l'état initial s
  Initialiser la récompense totale de l'épisode $R = 0$
  while non terminé **do**
    Sélectionner l'action a en utilisant la politique $\pi(s, \theta)$ avec exploration
    Exécuter l'action a, observer la récompense r et l'état suivant s'
    Stocker la transition (s, a, r, s') dans D
    Échantillonner un mini-lot de transitions $(s_j, a_j, r_j, s'j)$ aléatoirement à partir de D
    Calculer les valeurs cibles Q :
      $Q\text{cible}(s_j, a_j) = r_j + \gamma \cdot Q(s'j, \arg\max a' Q(s'j, a', \theta), \theta')$
    Mettre à jour le réseau principal en utilisant la descente de gradient :
      $\theta = \theta - \alpha \cdot \nabla\theta \left(Q(s_j, a_j, \theta) - Q_{\text{cible}}(s_j, a_j)\right)^2$
    Mettre à jour le réseau cible tous les C pas : $\theta' = \theta$
    Mettre à jour l'état $s = s'$
    Mettre à jour la récompense totale de l'épisode $R = R + r$
  end while
end for

θ représente les paramètres du réseau neuronal, N est la capacité de la mémoire de relecture, M est le nombre total d'épisodes, ϵ est le taux d'exploration, γ est le facteur de réduction, α est le taux d'apprentissage et C est la fréquence de mise à jour du réseau cible.

4.3 les outils utilisés

4.3.1 Python

Python est un langage de programmation interprété, polyvalent et facile à apprendre. Il a été créé par Guido van Rossum et sa première version a été publiée en 1991. Python se distingue par sa syntaxe claire et lisible, ce qui en fait un excellent choix pour les débutants en programmation.

Il est largement utilisé dans différents domaines, tels que le développement Web, l'analyse de données, l'intelligence artificielle et l'automatisation des tâches. Python dispose d'une vaste bibliothèque standard qui facilite le développement d'applications et permet de réaliser une variété de tâches sans avoir à réinventer la roue à chaque fois.

En résumé, Python est un langage de programmation puissant et polyvalent, apprécié pour sa simplicité et sa communauté active. Sa popularité est en constante croissance en raison de sa facilité d'utilisation et de son large éventail d'applications.

4.3.2 OpenAI GYM

La bibliothèque GYM en Python est un framework open-source développé par OpenAI. Elle fournit une interface standardisée et facile à utiliser pour créer, tester et comparer des algorithmes d'apprentissage par renforcement (RL - ReinforMECent Learning).

La bibliothèque GYM propose une large gamme d'environnements, allant des jeux classiques tels que Pong et CartPole à des problèmes plus complexes comme les simulations de robotique. Chaque environnement fournit une interface cohérente avec des méthodes pour effectuer des actions, obtenir des observations de l'environnement et recevoir des récompenses.

De plus, GYM permet aux utilisateurs de développer leurs propres environnements personnalisés, ce qui facilite l'adaptation de la bibliothèque à des problèmes spécifiques. GYM prend également en charge l'évaluation et la comparaison des performances des algorithmes d'apprentissage par renforcement grâce à des métriques standardisées.

En résumé, la bibliothèque GYM en Python fournit une infrastructure robuste et conviviale pour le développement et l'évaluation d'algorithmes d'apprentissage par renforent. Elle facilite l'expérimentation et la comparaison des performances des agents logiciels dans une variété d'environnements.

4.3.3 TensorFlow

La bibliothèque TensorFlow en Python est un framework open-source développé par Google. Il est largement utilisé pour le développement et le déploiement de modèles d'apprentissage automatique (machine learning) et d'apprentissage profond (deep learning). TensorFlow offre une architecture flexible et extensible pour la création de réseaux neuronaux, le traitement de gros volumes de données et l'exécution d'opérations mathématiques complexes.

L'une des caractéristiques clés de TensorFlow est sa représentation des modèles d'apprentissage automatique sous la forme de graphes de flux de données. Les n ?uds de ces graphes représentent des opérations mathématiques et les arêtes indiquent le flux des données entre ces opérations. Cette structure permet une exécution efficace des calculs et facilite la distribution des tâches sur différentes ressources matérielles.

TensorFlow propose également une API riche et flexible pour la création de modèles d'apprentissage profond. Il offre une grande variété de couches (layers) préconstruites pour la création de réseaux de neurones, ainsi que des fonctionnalités avancées telles que le transfert d'apprentissage (transfer learning) et la génération automatique de code optimisé.

En résumé, TensorFlow est une bibliothèque puissante et populaire en Python pour le développement de modèles d'apprentissage automatique et d'apprentissage profond. Elle offre une grande flexibilité, des fonctionnalités avancées et des performances élevées, ce qui en fait un choix privilégié pour de nombreux projets d'intelligence artificielle.

4.3.4 PyCharm

PyCharm est un environnement de développement intégré (IDE) spécialement conçu pour le langage de programmation Python. Développé par JetBrains, PyCharm offre un large éventail de fonctionnalités pour faciliter le développement d'applications Python.

En tant qu'IDE, PyCharm propose un éditeur de code avancé avec des fonctionnalités telles que la coloration syntaxique, l'autocomplétion intelligente, la navigation facilitée et la refonte automatique du code. Il dispose également d'un débogueur intégré qui permet de traquer et corriger les erreurs plus facilement.

PyCharm facilite également la gestion des projets Python en offrant des fonctionnalités telles que la création de projets, l'organisation des fichiers source, l'intégration avec les systèmes de contrôle de version et la gestion des dépendances via l'utilisation de l'outil de gestion de packages pip.

En résumé, PyCharm est un environnement de développement intégré (IDE) complet et puissant pour Python. Il offre une multitude de fonctionnalités pour faciliter le développement d'applications Python, allant de l'édition du code à la gestion des projets, en passant par le débogage, l'analyse du code et l'intégration avec des frameworks populaires.

4.3.5 Matplotlib

Matplotlib est une bibliothèque de visualisation de données en Python. Elle offre une large gamme d'outils permettant de créer des graphiques de haute qualité, des diagrammes, des histogrammes, des nuages de points, des courbes, etc. Matplotlib est très flexible et permet un contrôle précis sur l'apparence des graphiques. Il fournit également des fonctionnalités avancées telles que la création de sous-graphiques, l'ajout de légendes, l'étiquetage des axes, la personnalisation des couleurs et des styles, ainsi que l'exportation des graphiques dans différents formats de fichiers. Matplotlib est largement utilisé

dans le domaine de la science des données, la visualisation de données, la recherche et l'analyse statistique.

4.4 Résultats et interprétations

Cette section présente les résultats de la simulation qui permettent d'évaluer les performances du thème suggéré. La simulation est basée sur le scénario suivant,

Nous prenons en considération une configuration où une cellule unique est utilisée avec une largeur de bande de $W = 10MHz$. Un eNB est déployé avec un serveur MEC (Mobile Edge Computing) situé au centre. Les utilisateurs finaux (UE) sont dispersés de manière aléatoire dans un rayon de 200 mètres autour de l'eNB. La capacité de calcul du serveur MEC est de $F = 6GHz/sec$, tandis que la fréquence du CPU de chaque UE est de $f_n^l = 1GHz/sec$. Les paramètres de puissance de transmission et de puissance d'inactivité de l'UE sont fixés à $P_n = 500mW$ et $P_i^n = 100mW$ [38].

Nous faisons l'hypothèse que la taille des données de la charge de calcul, notée Bn (en kbits), suit une distribution uniforme entre 300 et 500, tandis que le nombre de cycles CPU requis noté Dn, suit une distribution uniforme entre 100 et 1000. Dans un souci de simplicité, le poids de décision de chaque utilisateur final est fixé à $I_n^t = I_n^e = 0,5$.

Nous évaluons les algorithmes proposés en les comparant à deux autres méthodes, en fonction des paramètres suivants, l'option "Full Local" implique que tous les utilisateurs effectuent leurs tâches en utilisant leurs propres ressources de calcul locales. L'option "Full Offload" signifie que tous les utilisateurs délèguent leurs tâches au serveur MEC, et que l'ensemble des ressources informatiques F sont réparties de manière équitable entre tous les utilisateurs.

Nous commençons par présenter le coût total du système MEC en fonction de l'augmentation du nombre total d'utilisateurs finaux dans le réseau, avec une capacité de calcul du serveur MEC fixée à $F = 6GHz/sec$.

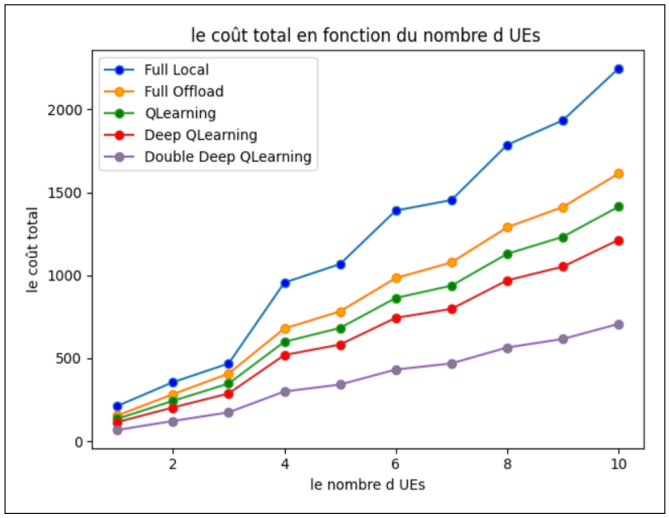

FIGURE 4.1 – le coût total du système MEC en fonction du nombre d'UE

Globalement, à mesure que le nombre d'UE augmente, le coût total des cinq méthodes augmente naturellement. Dans la figure 4.1, la méthode DDQN proposée peut obtenir le meilleur résultat, puis DQN et Q-learning suit avec un petit écart, les performances de ces trois méthodes sont relativement stables. La courbe de Full Offload est un peu plus élevée que celle du DQN et de Q-learning au point 3 UE, mais elle augmente beaucoup plus rapidement lorsqu'il y a plus d'UE, et on remarque que DDQN donne le meilleur résultat. En effet, lorsque le nombre d'utilisateurs est élevé, la capacité du serveur MEC n'est pas suffisante pour répondre aux besoins des utilisateurs. La capacité du serveur MEC n'est pas suffisante pour offrir à tous les utilisateurs une décharge de calcul. Un serveur MEC à capacité limitée ne doit pas servir un trop grand nombre d'utilisateurs. La manière de choisir les utilisateurs déchargés devient donc très importante dans ce cas.

La figure 4.2 illustre comment le coût total du système MEC évolue en fonction de deux variables : la capacité de calcul du serveur MEC et la taille des données (B_n) de la tâche de téléchargement, avec un nombre d'UE fixe à 10.

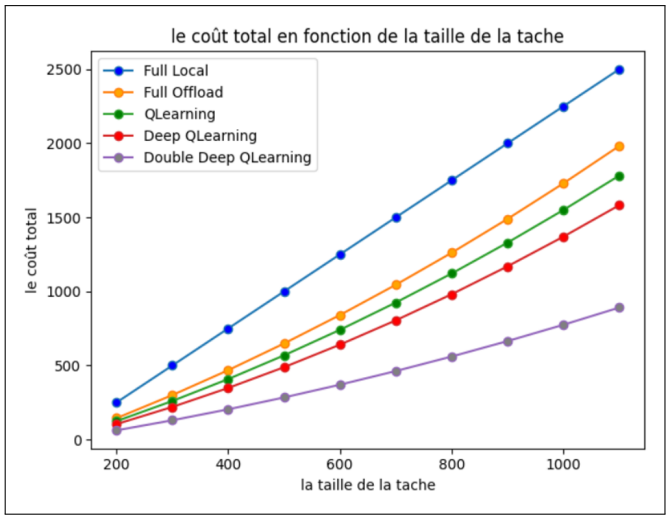

FIGURE 4.2 – le coût total du système MEC en fonction de la taille des tâches

La croissance de la taille des données de déchargement entraîne une augmentation des coûts totaux de toutes les méthodes, étant donné que cela engendre une consommation accrue de temps et d'énergie lors du processus de déchargement. Cette augmentation de la consommation de temps et d'énergie se traduit par une hausse du coût total du système de MEC. La méthode DDQN proposée se distingue en obtenant les meilleurs résultats, principalement grâce à sa tendance à augmenter de manière plus lente par rapport aux autres méthodes. En revanche, la courbe de la méthode Full Local affiche une augmentation beaucoup plus rapide que les quatre autres méthodes lorsque la taille des données augmente. Cela démontre que plus la taille des données de la tâche de déchargement est grande, plus le calcul du déchargement permet de réduire le délai et la consommation d'énergie. Il est également important de noter que l'augmentation de la taille des données peut entraîner une hausse significative du coût total de la MEC,

principalement en raison des facteurs D_n et B_n. En effet, D_n et B_n sont positivement corrélés, ce qui conduit à une croissance simultanée de notre fonction objectif.

La figure 4.3 illustre comment le coût total du système MEC évolue en fonction de F la capacité de calcul du serveur MEC, avec un nombre d'UE fixe à 10.

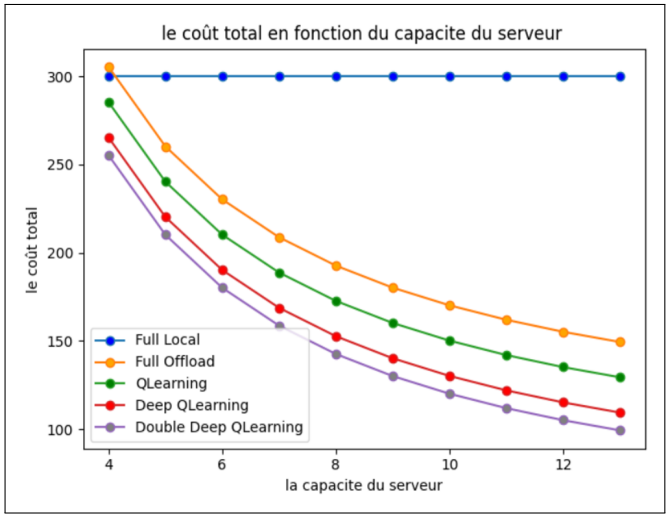

FIGURE 4.3 – le coût total du système MEC en fonction de la capacité de calcul du serveur MEC

La figure 4.3 met en évidence que la méthode DDQN propose les meilleurs résultats, tandis que les méthodes DQN et Q-learning présentent une légère divergence par rapport à la méthode DDQN. Dans cette figure, la courbe Full Local reste inchangée avec l'augmentation de la capacité du serveur MEC. Cela peut être expliqué par le fait que le calcul local n'utilise pas les ressources des serveurs MEC. Les autres courbes diminuent à mesure que la capacité de calcul du serveur MEC augmente, car cela réduit le temps d'exécution.

4.5 Conclusion

Dans ce chapitre, nous avons utilisé une simulation pour minimiser le coût total du système Mobile Edge Computing (MEC) en exploitant les algorithmes d'apprentissage par renforcement, notamment le Q-learning, le DQN et le DDQN. Nous avons évalué ces algorithmes en les comparant à deux autres méthodes, à savoir "Full Local" et "Full Offload", en fonction de certains paramètres.

Les résultats obtenus ont démontré que la méthode DDQN proposait les meilleurs résultats en termes de minimisation du coût total du système MEC. Cette approche a réussi à optimiser efficacement l'allocation des tâches et des ressources de calcul entre les utilisateurs et le serveur MEC. Elle a permis de réduire les coûts en tirant parti de la combinaison optimale de ressources locales et de délégation vers le serveur MEC.

D'autre part, les méthodes DQN et Q-learning ont également montré des performances intéressantes, bien qu'elles présentent une légère divergence par rapport à la méthode DDQN. Cela souligne l'importance de l'algorithme utilisé dans l'apprentissage par renforcement pour obtenir des résultats optimaux dans un contexte MEC.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

En résumé, ce projet de fin d'études a été une occasion pour nous d'acquérir de nouvelles compétences et de découvrir le domaine de l'apprentissage par renforcement. Nous avons pu relever divers défis liés à ce domaine à travers nos différentes missions, notamment la modélisation de problèmes d'optimisation qui nous a particulièrement enthousiasmés. Les performances des schémas que nous avons proposés ont été évaluées et comparées à quelques solutions de base. Les résultats de simulation démontrent que nos schémas peuvent surpasser les autres solutions de base dans différentes configurations du système. Nous avons rencontré des difficultés majeures lors de l'établissement et de la conception des problèmes de décision pour la charge de calcul et l'allocation des ressources de calcul dans le contexte de la MEC, ainsi que dans la dérivation de solutions basées sur l'apprentissage par renforcement pour ces problèmes. En conclusion, notre étude a démontré que l'utilisation des algorithmes d'apprentissage par renforcement, en particulier le DDQN, peut contribuer à la minimisation du coût total du système MEC. Ces résultats suggèrent des pistes prometteuses pour la conception et l'optimisation des systèmes MEC, en offrant des perspectives d'amélioration des performances, de l'efficacité et de l'allocation des ressources. Cependant, il convient de poursuivre les recherches pour explorer les avantages et les limites de ces méthodes et pour aborder les défis spécifiques liés à l'application pratique de ces approches dans des scénarios réels de MEC.

BIBLIOGRAPHIE

[1] G. A. Akpakwu, B. J. Silva, G. P. Hancke, A. M. Abu-Mahfouz, A survey on 5g networks for the internet of things : Communication technologies and challenges, IEEE Access 6 (2018) 3619 ?3647. doi :10.1109/ ACCESS.2017.2779844.

[2] A. u. R. Khan, M. Othman, S. A. Madani, S. U. Khan, A survey of mobile cloud computing application models, IEEE Communications Surveys Tutorials 16 (1) (2014) 393 ?413. doi :10.1109/SURV.2013.062613. 00160.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young, Mobile edge computing ?a key technology towards 5G, ETSI white paper 11 (11) (2015) 1 ?16.

[4] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, Y. Zhang, Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks, IEEE Access 4 (2016) 5896 ?5907. doi :10. 1109/ACCESS.2016.2597169.

[5] S. Nunna, A. Kousaridas, M. Ibrahim, M. Dillinger, C. Thuemmler, H. Feussner, A. Schneider, Enabling real-time context-aware collaboration through 5g and mobile edge computing, in : 2015 12th International Conference on Information Technology - New Generations, 2015, pp. 601 ?605. doi :10.1109/ITNG.2015.155.

[6] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le,W.-J. Hwang, Z. Ding, A survey of multi-access edge computing in 5g and beyond : Fundamentals, technology integration, and state-of-the-art, IEEE Access 8 (2020) 116974 ?117017. doi :10.1109/ACCESS.2020.3001277.

[7] S. Qureshi, S. Hassan, D. N. Jayakody, Divide-and-Allocate : An Uplink Successive Bandwidth Division NOMA System, Transactions on Emerging Telecommunications Technologies 29. doi :10.1002/ett.3216.

[8] S. Zhou, Y. Sun, Z. Jiang, Z. Niu, Exploiting moving intelligence : delay-optimized computation offloading in vehicular fog networks. IEEE Commun. Mag. 57(5), 49 ?55 (2019). https ://doi.org/10.1109/MCOM.2019.1800230

[9] P. Mach, Z. Becvar, Mobile edge computing :a survey on architecture and computation offloading, in IEEE Communications Surveys & Tutorials, vol. 19, no. 3, (2017), pp. 1628-1656

[10] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. De Foy, and Y. Zhang, Mobile edge cloud system : Architectures, challenges, and approaches, IEEE Systems Journal, vol. 12, no. 3, pp. 2495-2508, 2017.

[11] J. Liu,Y. Mao, J. Zhang, K.B. Letaief, Delay-optimal computation task scheduling for mobileedge computing systems, inIEEE International Symposium on Information Theory,Barcelona, (2016), pp. 1451-1455

[12] J. O. Fajardo, I. Taboada, F. Liberal, Radio-aware service-level scheduling tominimize downlink traffic delay through mobile edge computing, in International Conference on Mobile Networks and Management, Santander, 16-18 Sep 2015 (Springer, 2015), pp. 121-134

[13] Y. Mao, J. Zhang, K.B. Letaief, Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J. Sel. Areas Commun. 34(12), 3590-3605 (2016)

[14] M. Kamoun,W. Labidi,M. Sarkiss, Joint resource allocation and offloading strategies in cloud enabled cellular networks, in IEEE International Conference on Communications, London, 8-12 June 2015 (IEEE, 2015), pp. 5529-5534

[15] W. Labidi, M. Sarkiss, M. Kamoun, Energy-optimal resource scheduling and computation offloading in small cell networks, in IEEE 22nd International Conference on Telecommunications, Sydney, 27-29 Apr 2015 (IEEE, 2015), pp. 313-318

[16] W. Labidi, M. Sarkiss, M. Kamoun, Joint multi-user resource scheduling and computation offloading in small cell networks, in IEEE International Conference on Wireless and Mobile Computing, Network, and Communications, Abu Dhabi, 19-21 Oct 2015 (IEEE, 2015), pp. 794-801

[17] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, Y. Zhang, Energyefficient offloading for mobile edge computing in 5G heterogeneous networks. IEEE Access 4, 5896-5907 (2016)

[18] M. Chen, B. Liang, M. Dong. Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point, in 2017 IEEE International Conference on Computer Communications, Atlanta, Georgia, 1-4 May 2017 (IEEE, 2017), pp. 1-6

[19] T.Q. Dinh, J. Tang, Q.D. La, T.Q.S. Quek, Offloading in mobile edge computing : task allocation and computational frequency scaling. IEEE Trans. Commun. 65(8), 3571-3584 (2017)

[20] Y. Wang, M. Sheng, X. Wang, L. Wang, J. Li, Mobile-edge computing : partial computation offloading using dynamic voltage scaling. IEEE Trans. Commun. 64(10), 4268-4282 (2016)

[21] J. Ren, G. Yu, Y. Cai, Y. He, F. Qu, Partial offloading for latency minimization in mobile-edge computing, in 2017 IEEE Global Communications Conference, Singapore, 4-8 Dec 2017 (IEEE, 2017), pp. 1-6

[22] U. Saleem,Y. Liu, S. Jangsher, andY. Li, Performance guaranteed partial offloading formobile edge computing, in 2018 IEEE Global Communications Conference, Abu Dhabi, 9-13 Dec 2018 (IEEE, 2018), pp. 1-6

[23] S.E.Mahmoodi,R.N.Uma,K.P. Subbalakshmi, Optimal joint scheduling and cloud offloading for mobile applications. IEEE Trans. Cloud Comput. 7(2), 301-313 (2019). https ://doi.org/10.1109/TCC.2016.2560808

[24] W. Zhang, Y. Wen, D.O. Wu, Collaborative task execution in mobile cloud computing under a stochastic wireless channel. IEEE Trans. Wirel. Commun. 14(1), 81-93 (2015)

[25] O. Munoz,A. Pascual-Iserte, J.Vidal, Joint Allocation of Radio and Computational Resources in Wireless Application Offloading (Future Network & Mobile Summit, Lisbon, 2013), pp. 1-10

[26] O. Munoz, A. Pascual-Iserte, J. Vidal, Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading. IEEE Trans. Veh. Technol. 64(10), 4738-4755 (2015)

[27] Y. Mao, J. Zhang, S.H. Song, K.B. Letaief, Power-delay trade-off in multi-user mobile-edge computing systems, in IEEE Global Communications Conference,Washington, DC, 4-8 Dec 2016 (IEEE, 2016), pp. 1-6

[28] R. R. Schaller, Moore's law : past, present and future, IEEE Spectr., vol. 34, no. 6, pp. 52-59, 1997.

[29] D. Silver, Deep reinforcement learning, in International Conference on Machine Learning (ICML), 2016.

[30] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, A Brief Survey of Deep Reinforcement Learning, IEEE Signal Process. Mag. Spec. Issue Deep Learn. Image Underst., pp. 1-14, 2017.

[31] L. Lin, Reinforcement Learning for Robots Using Neural Networks, Report, C., pp. 1-155, 1993.

[32] H. Van Hasselt, A. C. Group, and C. Wiskunde, Double Q-learning, Nips, pp. 1-9, 2010.

[33] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, Policy Gradient Methods for Reinforcement Learning with Function Approximation, Adv. Neural Inf. Process. Syst. 12, pp. 1057-1063, 1999.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al., Playing atari with deep reinforcement learning, arXiv :1312.5602, Dec. 2013.

[35] Zhenjiang Zhang(corresponding author), Chen Li, ShengLung Peng, Xintong Pei, A New Task Offloading Algorithm in Edge Computing, School of Electronic and Information Engineering, Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, Beijing Jiaotong University, China 2National Taipei University of Business, China

[36] M. E. Khoda, M. Razzaque, A. Almogren, M. M. Hassan, A. Alamri, A. Alelaiwi, et al., Efficient computation offloading decision in mobile cloud computing over 5g network, Mobile Networks and Applications, vol. 21, no. 5, pp. 777-792, 2016.

[37] J. Li, H. Gao, T. Lv, and Y. Lu, Deep reinforcement learning based computation offloading and resource allocation for mec, in 2018 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1-6, IEEE, 2018.

[38] A. Kaur and R. Kaur, An efficient framework for improved task offloading in edge computing, in International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, pp. 94-101, Springer, 2018.