الجمهورية الجزَائِرية الديمقرَاطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزَارة التعليم العَالي و البحث العلمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جَامعة عمَار ثليجي بِالأَغوَاط
UNIVERSITE AMAR TELIDJI LAGHOUAT

FACULTE DES SCIENCES
DEPARTEMENT D' INFORMATIQUE

Mémoire de Master

Domaine : Mathématiques et Informatique
Filière : Informatique
Option : Réseaux, Systèmes, et Applications Réparties

Présenté par : KHENE Yacine

Thème

Offloading pour les Mobile Edge Computing à l'aide de l'apprentissage par renforcement

Soutenu publiquement devant le jury composé de

Mr.	ALLAOUI Tahar	MCB	Président	UATL
Mr.	OULADDJEDID Lakhdar	MCA	Examinateur	UATL
Mr.	BENARFA Abdelmajid	MCB	Examinateur	UATL
Mr.	BENDOUMA Tahar	MCA	Encadrant	UATL
Mr.	REMACHE Idris	Doctorant	Co-Encadrant	UATL

Année universitaire 2023/2024

الجمهورية الجزَائِرية الديمقرَاطية الشعبية
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزَارة التعليم العَالي و البحث العلمي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جَامعة عمَار ثليجي بِالأَغوَاط
UNIVERSITY OF LAGHOUAT

FACULTY OF SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Master Thesis

Domain : Mathematics and Computer Science
Field : Computer Science
Option : Networks, Systems, and Distributed Applications

Submitted by : KHENE Yacine

THEME

Offloading in Mobile Edge Computing using Reinforcement Learning

Examination Committee :

Mr.	ALLAOUI Tahar	MCB	President	UATL
Mr.	OULADDJEDID Lakhdar	MCA	Reviewer	UATL
Mr.	BENARFA Abdelmajid	MCB	Reviewer	UATL
Mr.	BENDOUMA Tahar	MCA	Supervisor	UATL
Mr.	REMACHE Idris	PhD student	Co-Supervisor	UATL

Academic Year 2023/2024


بسم الله الرحمن الرحيم

Remerciements

Tout d'abord, je remercie mon Dieu, de m'avoir donné la force, la volonté et le courage afin d'accomplir ce modeste travail.

Je tiens à exprimer ma gratitude à mon encadreur pour ses conseils et ses orientations tout au long de ce projet.

Je remercie également toutes les personnes qui ont contribué, de près ou de loin, à l'aboutissement de ce travail.

Finalement, je souhaite exprimer ma profonde gratitude à ma famille, qui m'a toujours soutenu durant mon cursus universitaire, ainsi qu'à l'ensemble des enseignants qui ont contribué à ma formation.

Dédicaces

Je dédie ce modeste travail :

A ma chère mère et à mon cher père qui se sont sacrifiés pour mettre à ma disposition tous les moyens nécessaires à fin de réussir dans mes études, que Dieu nous les garde.

A ma famille, particulièrement mes chers frères.

A tous mes chers(es) amis(es).

Merci Infiniment.

KHENE Yacine

Résumé

Avec l'avancement rapide des technologies mobiles et des applications gourmandes en ressources, le Mobile Edge Computing (MEC) s'est imposé comme une solution prometteuse pour soulager les dispositifs mobiles aux ressources limitées des tâches intensives en calcul, permettant aux appareils de offloading des charges de travail vers des serveurs MEC voisins. Pour minimiser le coût de calcul moyen en termes de consommation d'énergie et de délai dans mémoire tampon, ce travail considère un système multi-utilisateurs activé par MEC afin d'étudier une stratégie de offloading de calcul basée sur l'apprentissage par renforcement pour construire un système évolutif et améliorer ses performances. Cette stratégie, fondée sur le Deep Deterministic Policy Gradient (DDPG) dans un espace d'actions continu, est adoptée pour apprendre des politiques de offloading de calcul pour tous les utilisateurs. Les résultats numériques indiquent que la stratégie DDPG permet à chaque utilisateur d'apprendre une politique de offloading efficace et démontre sa supériorité aux approches basées sur des actions discrètes, comme le Deep Q-Network (DQN).

Mots clés : Mobile Edge Computing (MEC), Offloading, Apprentissage par renforcement, Deep Deterministic Policy Gradient (DDPG), Système multi-utilisateurs, Deep Q-Network (DQN).

ملخص

مع التقدم السريع في التكنولوجيا المحمولة والتطبيقات التي تتطلب موارد كبيرة، برزت الحوسبة على حافة الشبكة كحل واعد لتخفيف الأعباء على الأجهزة المتنقلة المحدودة في الموارد الكثيرة في الحساب مما يتيح للأجهزة تحميل أعباء العمل إلى خوادم الحوسبة المجاورة. لتقليل التكلفة المتوسطة للحوسبة من خلال استهلاك للطاقة ومدة التخزين المؤقتة، ينظر هذا العمل في نظام متعدد المستخدمين مُفعل بواسطة الحوسبة على حافة الشبكة، لدراسة استراتيجية تحميل الحوسبة المعتمدة على التعلم المعزز لبناء نظام قابل للتطوير وتحسين أدائه. تعتمد هذه الإستراتيجية القائمة على خوارزمية تدرج السياسة الحتمية العميقة في فضاء عمل مستمر، على تعلم سياسات التفريغ الحسابية لجميع المستخدمين.

تشير النتائج المتوصل إليها الى أن سياسة التفريغ فعالة وتظهر تفوقها مقارنة بالأساليب المعتمدة على الأعمال المنفصلة.

الكلمات المفتاحية الحوسبة على حافة الشبكة، التفريغ، التعلم المعزز، خوارزمية تدرج السياسة الحتمية العميقة.

Abstract

With advancement of mobile technologies and resource-intensive applications, Mobile Edge Computing (MEC) has emerged as a promising solution to alleviate resource-constrained mobile devices from computation-intensive tasks, allowing devices to offload workloads to nearby MEC servers. To minimize the average computation cost in terms of energy consumption and buffering delay, this work considers a multi-user MEC-enabled system to study a computation offloading strategy based on reinforcement learning, aimed at building a scalable system and improving its performance. This strategy, based on the Deep Deterministic Policy Gradient (DDPG) in a continuous action space, is adopted to learn computation offloading policies for all users. Numerical results indicate that the DDPG strategy enables each user to learn an efficient offloading policy and demonstrates its superiority over approaches based on discrete actions, such as the Deep Q-Network (DQN).

Keywords : Mobile Edge Computing (MEC), Offloading, Reinforcement Learning, Deep Deterministic Policy Gradient (DDPG), Multi-user System, Deep Q-Network (DQN).

Table des matières

Introduction Générale **x**

1 Cloud et Edge Computing : Concepts et Applications **2**
1.1 Introduction . . . 3
1.2 Cloud Computing . . . 3
1.2.1 Définition . . . 3
1.2.2 Types de déploiements Cloud . . . 4
1.2.3 Services de Cloud Computing . . . 5
1.2.4 Utilisations du Cloud Computing . . . 7
1.2.5 Mobile Cloud Computing . . . 7
1.2.6 Cloudlet . . . 8
1.2.7 Comparaison entre le Cloud Computing et le Cloudlet . . . 9
1.3 Edge Computing / Fog Computing . . . 10
1.3.1 Définition et Principe de Edge Computing . . . 10
1.3.2 Architecture de Edge Computing . . . 11
1.3.3 Fog Computing . . . 13
1.3.4 Caractéristiques . . . 14
1.4 Mobile Edge Computing . . . 15
1.4.1 Définition . . . 15
1.4.2 Architecture hiérarchique du MEC . . . 15
1.4.3 Applications dans les réseaux MEC . . . 19
1.4.4 Computation Offloading dans les réseaux MEC . . . 21
1.4.4.1 Décision de Computation Offloading . . . 22
1.4.4.2 Processus d'exécution du déchargement de calcul pour les utilisateurs . . . 23
1.4.4.3 Défis soulignés . . . 24
1.4.4.4 Métriques d'évaluation de déchargement de calculs dans les réseaux MEC . . . 24
1.4.4.5 Classifications de déchargement dans les réseaux MEC . . . 26
1.4.5 Comparaison etre le MCC et MEC . . . 28
1.5 Conclusion . . . 29

2 Intelligence Artificielle et Apprentissage Automatique : Fondements et Techniques **30**
2.1 Introduction . . . 31
2.2 Intelligence Artificielle . . . 31
2.2.1 Définition . . . 31
2.2.2 Évolution historique de l'IA . . . 32

2.2.3 Compréhension du fonctionnement de l'IA . . . 34
2.2.4 Branches de l'IA . . . 34
2.3 Machine Learning . . . 35
2.3.1 Définition . . . 35
2.3.2 Types de Machine Learning . . . 36
2.3.2.1 Apprentissage Supervisé . . . 36
2.3.2.2 Apprentissage Non Supervisé . . . 38
2.3.2.3 Apprentissage Semi-Supervisé . . . 40
2.3.2.4 Apprentissage par Renforcement . . . 40
2.4 Apprentissage par Renforcement . . . 40
2.4.1 Processus de décision markovien . . . 41
2.4.2 Composants et Phases de RL . . . 41
2.4.3 Classification des algorithmes du RL . . . 43
2.4.3.1 Model-Free RL . . . 43
2.4.3.2 Model-Based RL . . . 45
2.4.4 Avantages et Inconvénients des Approches Model-Based et Model-Free en Apprentissage par Renforcement . . . 45
2.4.5 Domaines d'applications de RL . . . 46
2.5 Conclusion . . . 47

3 Simulation et Performance des Algorithmes RL dans les Systèmes MEC 48
3.1 Introduction . . . 49
3.2 Modèle de système et formulation du problème . . . 49
3.2.1 Modèle de calcul . . . 50
3.2.1.1 Traitement local . . . 50
3.2.1.2 Traitement parallèle des tâches dans les serveurs MEC . . . 51
3.2.1.3 Coût de calcul . . . 51
3.3 Méthode décentralisée de déchargement dynamique des calculs . . . 52
3.3.1 Espace d'état . . . 52
3.3.2 Espace d'action . . . 53
3.3.3 Fonction de récompense . . . 53
3.4 Entraînements et tests . . . 53
3.4.1 Outils et environnements utilisés . . . 54
3.5 Résultats et discussion . . . 56
3.5.1 Paramètres de simulation . . . 56
3.5.2 Stratégies de comparaison . . . 56
3.5.3 Scénarios de comparaison . . . 57
3.5.3.1 Scenario 1 : Un seul utilisateur . . . 57
3.5.3.2 Scénario 2 : Plusieurs utilisateurs . . . 62
3.6 Conclusion . . . 63

Conclusion Générale 64

Bibliographie 69

Table des figures

1.1 Approche de Cloud Computing.[12] 4
1.2 Les services de Cloud Computing. 6
1.3 Architecture de Mobile Cloud Computing.[33] 8
1.4 Approche de Edge Computing. 11
1.5 Architecture de Edge Computing. 12
1.6 Approche Fog Computing. 14
1.7 Architecture de Mobile Edge Computing. 18
1.8 Cas d'utilisation du offloading de calculs dans les réseaux MEC. 21
1.9 Déchargement binaire et déchargement partial. 23
1.10 Processus d'exécution du déchargement de calcul pour les utilisateurs. 24
1.11 Classification de offloading. 26

2.1 Cycle de perception de l'IA. 31
2.2 Composants de l'IA. 35
2.3 Aperçu des catégories de ML. 36
2.4 Structure d'apprentissage supervisé. 37
2.5 Types de classification. 38
2.6 Formes de régression. 38
2.7 Exemple d'apprentissage non supervisé. 39
2.8 Exemple d'apprentissage semi supervisé. 40
2.9 Interaction entre l'agent et l'environnement dans le cadre d'un MDP. 41
2.10 Hiérarchie des algorithmes de Reinforcement Learning. 43
2.11 Classifications des méthodes pour Model-free RL. 44
2.12 Applications du RL. 47

3.1 Modèle de système. 49
3.2 Concept d'observation locale et de décision décentralisée. 52
3.3 Environnement Anaconda. 54
3.4 Bibliothèque TensorFlow. 55
3.5 Récompense moyenne par épisode pour un seul utilisateur. 57
3.6 Comparaison de puissance moyenne pour différentes stratégies. 59
3.7 Comparaison de délai moyen dans la mémoire tampon pour différentes stratégies. 60
3.8 Analyse du Compromis Énergie-Délai. 61
3.9 Comparaison des résultats de simulation pour les trois utilisateurs mobiles. . . . 62

Liste des tableaux

1.1 Comparaison des modèles de service cloud 6
1.2 Comparaison entre le Cloud Computing et les Cloudlet 9
1.3 Caractéristiques du Edge, Fog et Cloud Computing. 15
1.4 Comparaison entre MCC et MEC.[47] 29

2.1 Évolution de l'IA. 33
2.2 Avantages et Inconvénients des algorithmes Model-Based et Model-Free. 46

3.1 Spécifications matérielles. 54
3.2 Paramètres de la simulation. 56

Liste des abréviations

API	Application Programming Interface
AR	Augmented Reality
BS	Base Station
CC	Cloud Computing
DDPG	Deep Deterministic Policy Gradient
DL	Deep Learning
DQN	Deep Q-Network
EC	Edge Computing
FC	Fog Computing
IA	Intelligence Artificielle
IoT	Internet of Things
MCC	Mobile Cloud Computing
MEC	Mobile Edge Computing
ML	Machine Learning
MV	Machine Virtuelle
RL	Reinforcement Learning
UAV	Unmanned Aerial Vehicle

Introduction générale

Avec l'émergence des technologies mobiles et le développement rapide des applications gourmandes en ressources, telles que les jeux en ligne, la réalité augmentée et les services d'Internet des objets, les systèmes informatiques traditionnels atteignent leurs limites. Les contraintes liées au délai de traitement, à la bande passante et à la capacité de traitement rendent difficile la gestion efficace des demandes croissantes en matière de calcul.

C'est dans ce contexte que le Mobile Edge Computing (MEC) s'impose comme une solution prometteuse. En rapprochant les capacités de calcul des utilisateurs finaux, le MEC permet de réduire considérablement les délais de traitement en plaçant des serveurs de calcul à la périphérie du réseau.

L'un des défis majeurs dans les environnements MEC est la gestion efficace de l'offloading, c'est-à-dire déterminer s'il faut décharger et où décharger certaines tâches computationnelles des appareils mobiles vers les serveurs MEC. Cette décision repose sur plusieurs facteurs, notamment les conditions du réseau, les ressources disponibles sur les serveurs MEC, la consommation d'énergie des dispositifs, et les exigences spécifiques des utilisateurs. L'objectif est de trouver le bon équilibre entre traitement local et à distance, tout en garantissant une performance optimale et une faible latence.

La problématique centrale de ce mémoire est donc la suivante : comment concevoir et implémenter des stratégies d'offloading efficaces et dynamiques dans les environnements MEC, en utilisant des techniques d'apprentissage par renforcement pour optimiser les performances de système.

Pour répondre à cette question, ce mémoire explorera l'utilisation des techniques d'apprentissage par renforcement, telles que le Deep Deterministic Policy Gradient (DDPG) et le Deep Q-Network (DQN), afin de simuler et d'analyser les résultats comparatifs pour minimiser le coût moyen de calcul pour chaque utilisateur en termes de consommation d'énergie et de délai moyen dans tampon.

Pour la réalisation de ce projet, nous avons structuré l'ensemble de notre travail en trois chapitres :

Le premier chapitre « **Cloud et Edge Computing : Concepts et Applications** » Ce chapitre présente le Cloud Computing, l'Edge Computing et le Mobile Edge Computing, en définissant leurs formes de déploiement, services et applications. Nous analysons également les principes et défis du MEC.

Le deuxième chapitre « **Intelligence Artificielle et Apprentissage Automatique : Fondements et Techniques** » Ce chapitre introduit l'intelligence artificielle et le machine learning, en décrivant leur évolution et leurs branches principales. Nous approfondissons l'apprentissage par renforcement, en détaillant ses composants et ses applications.

Le troisième chapitre « **Simulation et Performance des Algorithmes RL dans les Systèmes MEC** » Ce chapitre explore la simulation des systèmes de calcul, en comparant les algorithmes d'apprentissage par renforcement dans le MEC. Nous formulons les problèmes de traitement et discutons d'une méthode de déchargement basée sur DDPG, avant de présenter les résultats des simulations.

Enfin, nous achevons ce travail par une conclusion générale et quelques perspectives.

Chapitre 1

Cloud et Edge Computing : Concepts et Applications

1.1 Introduction

Dans ce chapitre, nous aborderons les concepts fondamentaux du Cloud Computing, de l'Edge Computing et du Mobile Edge Computing (MEC). Nous commencerons par définir le Cloud Computing et examiner ses différentes formes de déploiement, services et applications. Ensuite, nous nous intéresserons aux approches plus récentes, telles que l'Edge Computing et le Fog Computing, en mettant en lumière leurs architectures et caractéristiques. Enfin, nous analyserons le MEC, ses principes, son architecture, ainsi que les défis liés au déchargement de calculs, tout en le comparant au Cloud Computing classique.

1.2 Cloud Computing

1.2.1 Définition

Il existe un certain nombre de définitions proposées pour l'informatique en nuage ou Cloud Computing (CC), bien que celle la plus largement acceptée semble être celle proposée par l'Institut National des Normes et Technologies (NIST). En effet, le NIST a défini le CC comme "Le Cloud Computing est un modèle offrant un accès réseau pratique et à la demande à un ensemble de ressources informatiques configurables, telles que des réseaux, des serveurs, du stockage, des applications et des services, qui peuvent être rapidement allouées et libérées avec peu d'effort de gestion ou d'interaction avec le fournisseur de services."[31]
CC est devenu une solution populaire pour offrir un accès abordable et facile aux organisations informatiques externalisées telles que les centres de recherche et les entreprises, qui bénéficient du CC pour héberger leurs applications. Aujourd'hui, à l'ère du Big Data, ces infrastructures sont utiles pour réaliser des calculs sur de grandes quantités de données, pour les agréger mais aussi pour les archiver.[12] Grâce à la virtualisation, le CC est en mesure de répondre, avec la même infrastructure physique, aux besoins informatiques variés d'une large clientèle.[4] [17]
En raison de leur architecture centralisée, les infrastructures de CC ne répondent pas aux besoins de l'Internet des objets. La distance des centres de données par rapport aux utilisateurs et aux objets connectés entraîne des latences élevées, rendant leur utilisation incompatible dans de nombreux cas d'utilisation. Certains chercheurs vont jusqu'à dire que la latence pour accéder à un serveur dans une infrastructure de CC est élevée et imprévisible,[46] dépassant les 100 ms. De plus, le nombre croissant d'objets connectés par centre de données augmente

la charge réseau et rend difficile la mise à l'échelle.
Ceci est illustré dans la Figure 1.1 , où l'infrastructure de CC est l'origine ou la destination de tous les transferts de données.

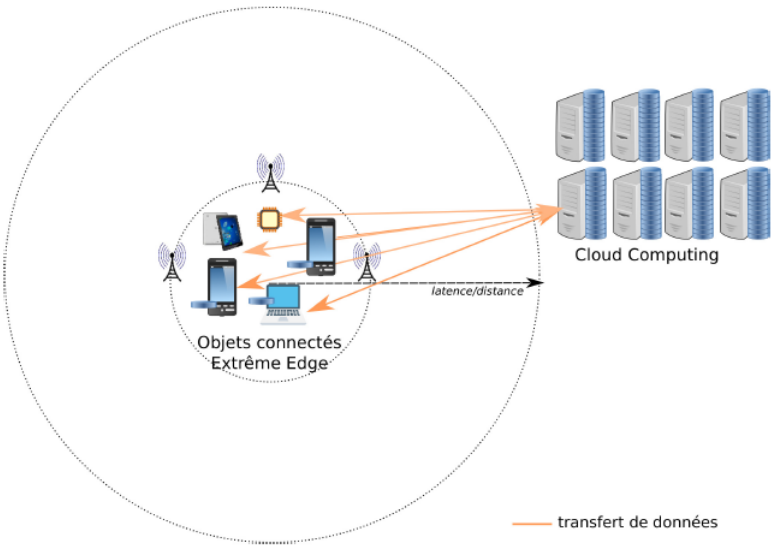

FIGURE 1.1 – Approche de Cloud Computing.[12]

1.2.2 Types de déploiements Cloud

Il existe trois principaux types de déploiements cloud ou d'architectures de CC sur lesquels les services cloud peuvent être mis en œuvre, chacun étant conçu pour répondre à des besoins spécifiques.

— **Cloud Public** Les clouds publics sont le type de déploiement de cloud computing le plus courant. Les ressources cloud (telles que les serveurs et le stockage) sont détenues et exploitées par un fournisseur de service cloud tiers, et livrées via Internet. Dans un cloud public, tout le matériel, tous les logiciels et toute l'infrastructure sont la propriété du fournisseur du cloud.

— **Cloud Privé** Un cloud privé consiste en l'ensemble des ressources de cloud computing qu'une entreprise ou une organisation utilise en exclusivité. Le cloud privé peut être situé physiquement dans un centre de données local de l'entreprise, ou être hébergé par un fournisseur de services tiers. Toutefois, dans un cloud privé, la maintenance des services et de l'infrastructure est toujours effectuée sur un réseau privé, le matériel et les logiciels sont exclusivement dédiés à l'entreprise.

— **Cloud Hybride** Un cloud hybride est un type de cloud computing qui associe une infrastructure locale (ou un cloud privé) à un cloud public. Les clouds hybrides permettent aux données et aux applications de se déplacer entre les deux environnements. [6]

1.2.3 Services de Cloud Computing

La plupart des services de CC peuvent être classés en trois grandes catégories : IaaS (infrastructure as a service), PaaS (platform as a service) et SaaS (software as a service). On les appelle parfois "pile" de Cloud Computing, car elles s'empilent les unes sur les autres.

— **Infrastructure as a Service (IaaS)** IaaS offre un accès à la demande aux services d'infrastructure informatique, dont le calcul, le stockage et la mise en réseau et la virtualisation.
Ce modèle offre le plus haut niveau de contrôle sur vos ressources informatiques et se rapproche le plus des ressources informatiques sur site traditionnel.

— **Platform as a Service (PaaS)** PaaS fournit toutes les ressources matérielles et logicielles nécessaires au développement d'applications cloud. Grâce aux PaaS, les entreprises peuvent se concentrer entièrement sur le développement d'applications, sans avoir à gérer ni à entretenir l'infrastructure sous-jacente.

— **Software as a Service (SaaS)** SaaS fournit une pile d'applications en tant que service complète, de l'infrastructure sous-jacente à la maintenance, en passant par les mises à jour du logiciel d'application.
Une solution SaaS est souvent une application d'utilisateur final dans laquelle le service et l'infrastructure sont gérés par le fournisseur de services cloud. [11]

la Figure 1.2 illustre les modèles de services Cloud disponibles.

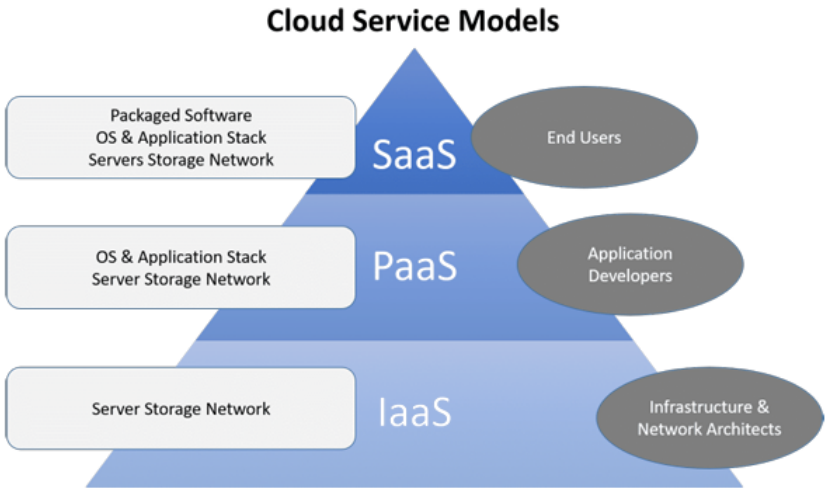

FIGURE 1.2 – Les services de Cloud Computing.

Voici le tableau comparatif 2.2 des avantages et des inconvénients des modèles IaaS, PaaS et SaaS : [3]

Modèles	**IaaS**	**PaaS**	**SaaS**
Avantages	- Contrôle total sur l'infrastructure. - Environnement hautement personnalisable et évolutif. - Coût efficace en termes d'achat de matériel basé sur la consommation. - L'utilisateur a un contrôle total sur toute l'infrastructure.	- Solution rentable et simple pour le développement et le déploiement d'applications. - Scalabilité illimitée pour la plateforme. - Automatisation des politiques commerciales.	- Solution rentable pour accéder aux applications sans besoin de mettre à jour le matériel. - Variété de solutions et services disponibles pour les utilisateurs . - l'utilisateur a toujours accès à la version la plus récente de l'application.
Inconvénients	- Préoccupations plus élevées en matière de sécurité par rapport aux autres modèles. - Nécessite généralement une formation afin d'apprendre à contrôler et gérer efficacement l'infrastructure.	- Risque de verrouillage des fournisseurs. - Problèmes potentiels d'intégration avec les applications locales.	- Manque de prise en charge de l'intégration avec d'autres applications locales. - Capacités de personnalisation minimales.

TABLE 1.1 – Comparaison des modèles de service cloud

1.2.4 Utilisations du Cloud Computing

Le Cloud Computing offre une large gamme d'applications possibles pour les organisations. Voici quelques cas d'utilisation courants :

- **Créer des applications natives Cloud** Créer, déployer et mettre à l'échelle rapidement des applications (web, mobiles et API).
- **Stocker, sauvegarder et récupérer des données** Protéger les données à moindre coût et à grande échelle en les transférant via internet vers un système de stockage en ligne hors site, accessible à partir de tout emplacement et appareil.
- **Tester et générer des applications** Réduire les coûts et délais de développement d'applications en utilisant des infrastructures cloud dont l'échelle peut être facilement adaptée.
- **Analyser des données** Unifier les données entre les équipes, les divisions et les emplacements dans le cloud.
  Utiliser ensuite des services cloud, par exemple de Machine Learning et d'intelligence artificielle, pour extraire des informations analytiques qui vous permettent de prendre des décisions éclairées. [6]

1.2.5 Mobile Cloud Computing

Le Mobile Cloud Computing (MCC) consiste à combiner les capacités des appareils mobiles tels que les véhicules, les UAVs (Unmanned Aerial Vehicle), etc avec les services de Cloud Computing. Son objectif est d'améliorer les fonctionnalités des appareils mobiles, notamment en termes de stockage et de calcul. Pour cela, il utilise des services web comme interface de programmation d'application (API) pour communiquer avec le cloud via une station de base proche. [47]
Ces infrastructures offrent aux clients mobiles, qui disposent de ressources limitées, la possibilité d'accéder aux ressources des infrastructures Cloud.
Deux architectures principales permettent cela.[16]La première approche implique le déploiement de serveurs, appelés "Cloudlets", à proximité des utilisateurs du réseau, une pratique similaire à celle du Edge Computing ou du Mobile Edge Computing. La seconde appelés "Extrême Edge Computing", consiste à mettre en commun les ressources des différents périphériques situés à proximité, pour une puissance de calcul adéquate.[20]

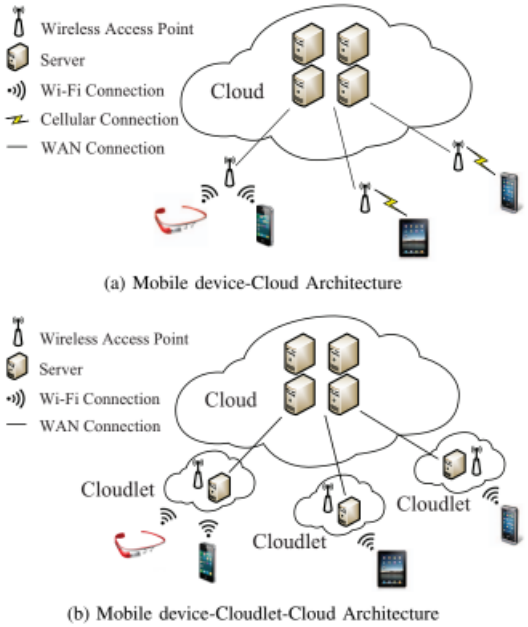

FIGURE 1.3 – Architecture de Mobile Cloud Computing.[33]

1.2.6 Cloudlet

Selon Satyanarayanan, les Cloudlets représentent une infrastructure Internet décentralisée et étendue, dont les cycles de calcul et les ressources de stockage peuvent être utilisés par les appareils mobiles proches.[38]
Contrairement au Cloud, un Cloudlet est physiquement proche de l'appareil mobile, généralement à une distance d'un saut et accessible via une liaison sans fil haut débit telle que le Wi-Fi.[40]
Les cloudlets constituent un niveau intermédiaire dans une hiérarchie à trois niveaux : dispositif mobile - cloudlet - cloud, afin d'assurer des temps de réponse précis.[18]
Les cloudlets permettent une déployabilité rapide des machines virtuelles (MV), que les utilisateurs peuvent personnaliser selon leurs besoins pour créer une image ou une superposition de MV contenant l'application et toutes ses exigences nécessaires pour fonctionner correctement.[21]
Dans les deux types d'implementations, l'image ou la superposition de MV est crée au moment de l'execution par l'utilisateur, offrant ainsi une grande flexibilité pour décharger (offloading) la charge de travail sur le cloudlet.
Cependant, malgré cette souplesse, le processus de création d'une image ou d'une superposition de MV, ainsi que l'encapsulation de l'état de l'application, peuvent être chronophages(demande beaucoup de temps). En fin de compte, la décision d'utiliser ou non les cloudlets comme sources de ressources riches

dépend entièrement de la conception, des exigences et de l'environnement spécifique de l'application.

1.2.7 Comparaison entre le Cloud Computing et le Cloudlet

L'évolution des technologies informatiques a introduit de nouveaux modèles de traitement des données et de gestion des ressources, notamment le cloud computing et les cloudlets.
Le tableau ci-dessous 1.2 offre une comparaison concise entre ces deux approches, mettant en lumière leurs différences clés en termes de puissance de calcul, disponibilité, coût, latence, et plus encore.

Caractéristiques	**Cloud Computing**	**Cloudlet**
Puissance de calcul	Haute	Intermittente
Expérience utilisateur	Qualité de service satisfaisante	Qualité de service excellente
Disponibilité	Élevée	Élevée
Mobilité du client	Supporte une grande mobilité	Limitée
Coût	Élevé	Faible à gratuit
Latence	Élevée	Faible
Sécurité des données	Élevée	Intermittente
Gestion	Centralisée et complexe	Décentralisée et autonome
Disponibilité hors ligne	Non disponible	Disponible
Bande passante	Faible	Élevée
Partage de ressources réseau	Grand nombre d'utilisateurs (1000s)	Nombre limité d'utilisateurs (10s)
Environnement de déploiement	Grands centres de données avec environnement contrôlé	Peut être déployé virtuellement n'importe où
Impact des catastrophes	Catastrophique	Impact minimal

TABLE 1.2 – Comparaison entre le Cloud Computing et les Cloudlet

1.3 Edge Computing / Fog Computing

1.3.1 Définition et Principe de Edge Computing

Edge Computing (EC) ou "Informatique En Périphérie" est un nouveau mode de calcul informatique qui permet de réduire le temps de traitement des données.[27]
Son idée centrale est de rapprocher le traitement informatique de la source des données, c'est a dire sur des serveurs situées en bordure du réseau, au plus près des utilisateurs et des objets connectés. [8]
Zha et al. décrivent le EC comme suit "Le Edge computing est un nouveau modèle informatique qui unifie les ressources proches de l'utilisateur en termes de distance géographique ou de distance réseau pour fournir des services de calcul, de stockage et de réseau pour les applications."[8]
Dans un réseau cellulaire, les serveurs sont déployés près de chaque station de base (BS). Les utilisateurs utilisent alors les ressources fournies par les serveurs situés dans la cellule à laquelle ils sont connectés. Ces serveurs effectuent les calculs à faible latence localement et transférent les tâches plus lourdes vers le Cloud.
Cette architecture facilite également le stockage de données : les utilisateurs interrogent d'abord les serveurs en périphérie du réseau (serveurs de EC), puis les requêtes non satisfaites (pas de ressources demandées) sont transmises au Cloud. Les données renvoyées par le Cloud sont mises en cache afin de pouvoir répondre directement aux futures requêtes, comme illustre par la Figure 1.4.
Les serveurs en périphérie ont des ressources moins importantes que celles du Cloud, mais elles sont adéquates pour exécuter des calculs nécessitant une faible latence. [12]

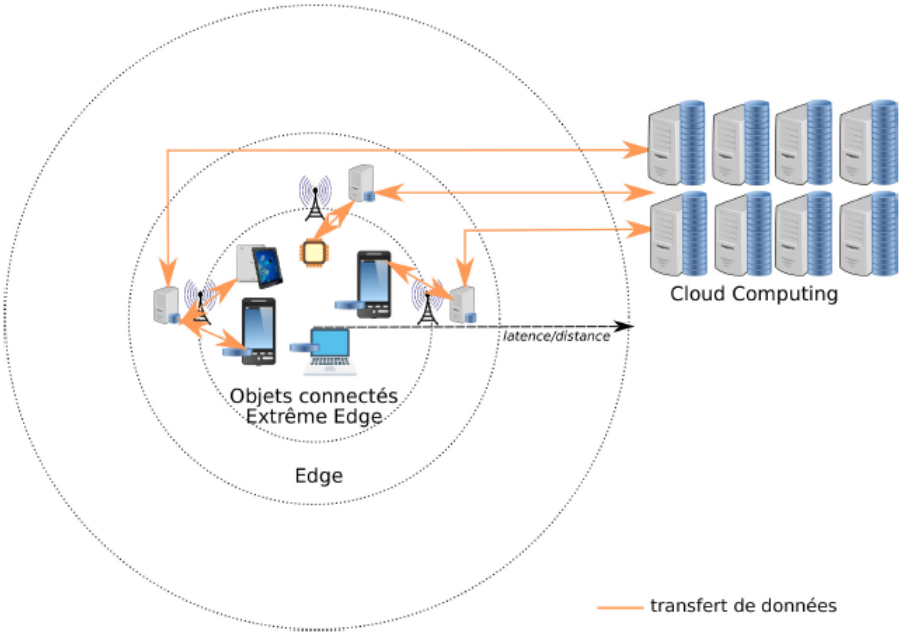

FIGURE 1.4 – Approche de Edge Computing.

1.3.2 Architecture de Edge Computing

La philosophie de EC repose principalement sur l'exécution de calculs au niveau de la périphérie, c'est-à-dire à proximité d'une source de données.
Les ressources informatiques en périphérie peuvent être un réseau ou une ressource informatique exécutée entre les utilisateurs finaux d'un côté, et les nœuds de brouillard (Fog) et les centres de données en CC de l'autre.
Les capteurs/appareils interconnectés au niveau de l'IoT génèrent et transfèrent des données entre eux en utilisant une infrastructure de réseau de communication moderne, puis ces données sont traitées à différents niveaux selon les besoins de l'application.
L'architecture de EC peut être expliquée en quatre couches : la couche IoT, la couche Edge, la couche Fog et la couche Cloud [2] comme la montre la Figure 1.5.

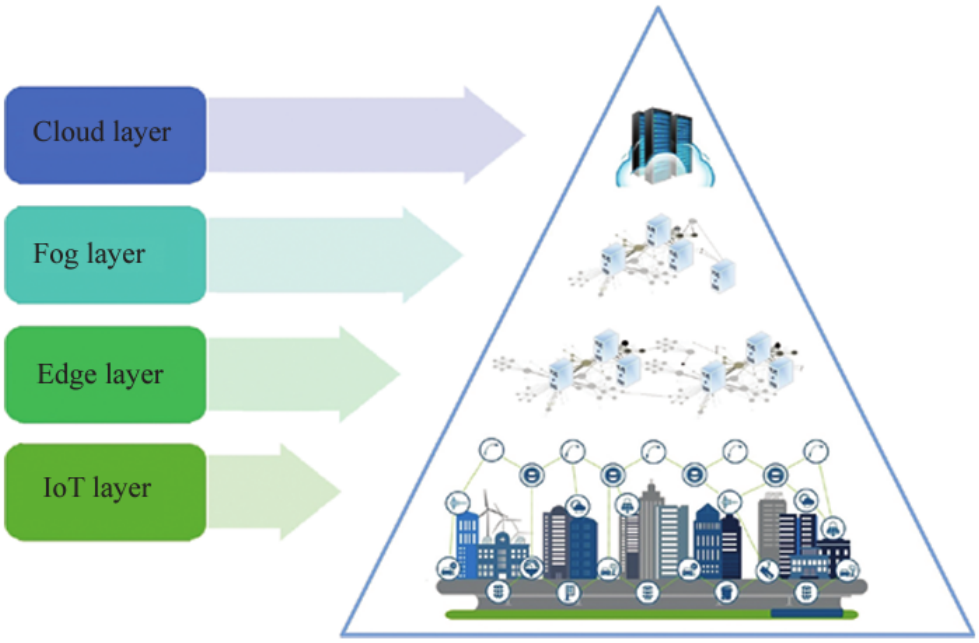

FIGURE 1.5 – Architecture de Edge Computing.

Dans cette architecture, la couche IoT est constituée de millions de dispositifs/capteurs qui génèrent en continu des données, échangent des informations importantes entre eux via une infrastructure de réseau de communication moderne, et surveillent et contrôlent les infrastructures critiques du monde intelligent. Ces dispositifs/capteurs IOT représentent les utilisateurs finaux de EC.
L'IoT et EC évoluent rapidement et de manière indépendante. En général, les dispositifs/capteurs IoT peuvent bénéficier de la capacité de calcul élevée et du grand stockage des couches EC, de Fog et de Cloud. Cependant, l'EC présente des avantages supplémentaires par rapport à FC et Cloud pour l'IoT, même si sa capacité de calcul et de stockage est plus limitée.
L'IoT nécessite spécifiquement un temps de réponse rapide plutôt qu'une capacité de calcul élevée et un grand stockage.
L'EC offre une capacité de calcul tolérable, suffisamment d'espace de stockage et un temps de réponse rapide pour satisfaire aux exigences des applications IoT. De plus, l'EC peut également bénéficier de la couche de Fog, car les capacités des ressources sur la couche de Fog sont relativement plus importantes que celles sur la couche Edge.
La couche de CC est composée de plusieurs serveurs haute performance et de dispositifs de stockage. Parmi toutes les couches, la couche de Cloud possède la puissance de calcul et la capacité de stockage les plus élevées.[2]

1.3.3 Fog Computing

Les infrastructures informatiques "en nuage bas", "en brouillard" ou encore appelées infrastructures de Fog Computing (FC) initialement proposées par Cisco en 2012 et sont aujourd'hui supportées par de nombreux industriels.[7]
La nouvelle architecture suggère l'installation de petits centres de données distribués (décentralisée) [23] dans différents sites en périphérie du réseau, au lieu de serveurs isolés près des utilisateurs. Chaque centre de données, comprenant généralement une dizaine de serveurs, offre des ressources de calcul et de stockage aux clients en périphérie du réseau. En raison de leur taille plus importante par rapport à l'Edge Computing, les sites de Fog supportent une meilleure mobilité des utilisateurs tout en améliorant les temps de réponse par rapport à une infrastructure de type Cloud.
Pour faciliter le déploiement, une proposition a été faite : installer les serveurs dans les points de présence de l'Internet.
Dans la Figure 1.6, chaque équipement se connecte à un serveur dans son environnement immédiat. Ce serveur effectue des calculs localement et délègue les tâches plus exigeantes à un autre serveur situé légèrement plus loin de l'utilisateur final. Ainsi, les calculs sont répartis dans toute l'infrastructure. Les serveurs à proximité des utilisateurs gèrent les opérations moins coûteuses mais nécessitant une réponse rapide, tandis que le Cloud Computing prend en charge les calculs très gourmands en ressources mais moins sensibles au temps de réponse.
Une telle architecture présente d'autres bénéfices, notamment en termes de confidentialité et de sécurité.[42] Confier ses données à un serveur situé proche de soi permet de limiter le nombre d'attaquants potentiels que nous pourrions rencontrer sur le chemin.

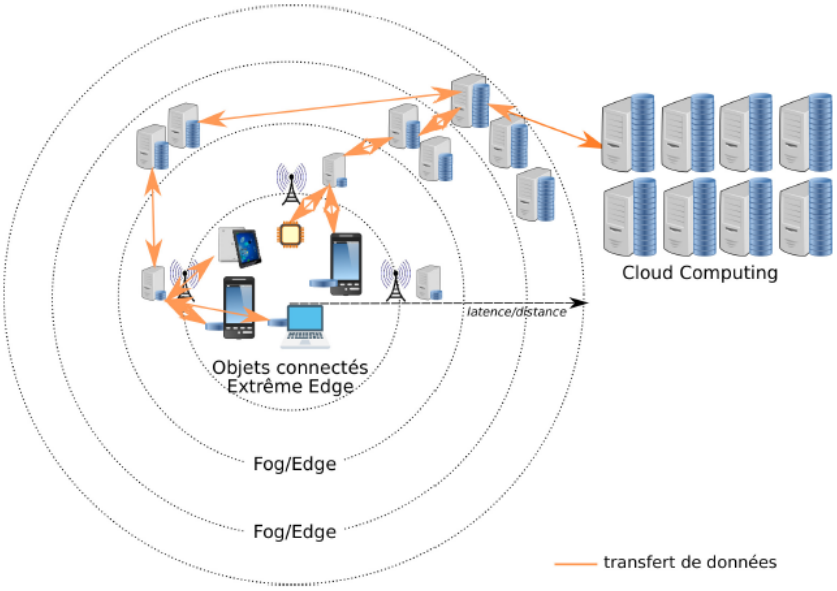

FIGURE 1.6 – Approche Fog Computing.

1.3.4 Caractéristiques

L'EC partage de nombreuses caractéristiques communes avec le FC avec des objectives similaires. Les deux modèles permettent la computation et le stockage à proximité des utilisateurs finaux pour réduire la latence du service et économiser la bande passante du réseau pour les applications sensibles au délai.[22] Ils ont des architectures distribuées, hiérarchiques et décentralisées, différentes de l'architecture centralisée de CC. Leurs emplacements de service sont également à proximité des utilisateurs finaux.
L'EC est située sur les appareils en périphérie, tandis que le FC se déroule dans des appareils en bord de réseau situés à un ou quelques sauts de réseau de edge. En comparaison avec les ressources de cloud computing, celles des modèles de edge computing et de fog computing (telles que les ressources de calcul, de communication et de stockage) sont relativement limitées. Dans certains travaux, les modèles Edge Computing et Fog Computing sont considérés comme étant une seule couche.
Cependant, il existe toujours quelques différences fondamentales entre les deux, même s'ils partagent les mêmes objectifs.[19] Les appareils Edge sont incapables de mettre en œuvre plusieurs applications dans la couche de EC, car les ressources limitées entraîneraient des conflits et augmenteraient également la latence de traitement. En revanche, l'FG peut surmonter avec succès ces limitations et éviter les conflits de ressources.
Le tableau 1.3 montre également une comparaison entre ces couches.

Caractéristiques	**Edge**	**Fog**	**Cloud**
Déploiement	Distribué	Distribué	Centralisé
Composantes	Noeuds Edge	Noeuds Fog	Ressources virtuelles
Computation	Limitée	Limitée	Illimitée
Stockage	Limitée	Limitée	Illimitée
Source de données	Le plus proche	Proche	Loin
Temps de réponse	Le plus rapide	Rapide	Lente
Nombre de nœuds	Très grand	Grand	Petit

TABLE 1.3 – Caractéristiques du Edge, Fog et Cloud Computing.

1.4 Mobile Edge Computing

1.4.1 Définition

Le Mobile Edge Computing (MEC), également connu sous le nom de Multi-access Edge Computing, est un paradigme émergeant qui vise à fournir des ressources de calcul aux utilisateurs mobiles directement au niveau du réseau, permettant ainsi le traitement rapide des applications intensives en calcul et sensibles à la latence.[47]
En rapprochant les ressources de calcul et de stockage en proximité physique des dispositifs sans fil finaux, le MEC répond aux besoins des utilisateurs mobiles en offrant des services à faible latence nécessitant des calculs intensifs ou de grands volumes de données.[30]

1.4.2 Architecture hiérarchique du MEC

Pour mieux comprendre la logique interne du MEC, nous commençons par présenter une architecture hiérarchique qui divise verticalement le système de Edge Computing en trois niveaux : la couche utilisateur, la couche edge et la couche cloud.
La couche utilisateur est caractérisée par le mode de communication sans fil entre les appareils mobiles et les infrastructures sans fil. Les couches edge et cloud font principalement référence aux ressources de calcul des serveurs edge et cloud, respectivement.[47]

1. **Les appareils de la couche utilisateur** : Les appareils de la couche utilisateur comprennent des capteurs, des smartphones, des véhicules, des compteurs intelligents et des dispositifs d'identification par radiofréquence.

Ces appareils accèdent aux serveurs edge via une communication sans fil, puis déchargent des tâches intensives en calcul vers les serveurs edge légers et distribués pour les traiter. Selon la topologie du réseau sans fil et les modes de communication, la communication entre les appareils mobiles et une infrastructure sans fil peut être divisée en trois modes suivants.[47]

— **Réseau hétérogène** : Les réseaux sans fil de nouvelle génération exécuteront des applications nécessitant une forte demande en débits de données élevés. Une solution pour aider à réduire l'exigence de débit de données est la densification du réseau par le déploiement de petites cellules. Une telle densification entraîne une efficacité spectrale plus élevée et peut réduire la consommation d'énergie d'un appareil mobile en raison de sa communication avec de petites stations de base cellulaires voisines. Cette solution améliore significativement la couverture du réseau. Le fonctionnement concurrent des macro stations de base (MBS) et des micro stations de base, et les stations de base assistées par des véhicules aériens sans pilote (UAV) sont désignées comme faisant partie d'un réseau hétérogène. Dans les réseaux hétérogènes, toutes les stations de base sont équipées de ressources de calcul et de fonctions d'intelligence artificielle. Les appareils mobiles limités en ressources peuvent décharger(offload) leurs tâches vers ces stations de base hétérogènes, qui peuvent ensuite utiliser une politique d'allocation de ressources de calcul fine pour traiter les tâches déchargées.[47]

— **Réseau véhiculaire** : Les réseaux véhiculaires sont indispensables à un environnement de ville intelligente en raison de plusieurs applications qui améliorent la qualité de vie, la sécurité et la sûreté. Un réseau véhiculaire se forme entre les véhicules en mouvement, les unités de bord de route et les piétons, et peut être déployé dans des environnements ruraux, urbains et autoroutiers.

La communication véhicule-tout (Vehicle-to-everything) permet aux véhicules de communiquer avec d'autres véhicules et leur environnement via des liaisons sans fil. Cette communication comprend trois principaux scénarios : véhicule à véhicule, véhicule à infrastructure et véhicule à piéton. Les technologies couramment utilisées sont les communications dédiées à courte portée, les normes de la famille IEEE 802.11p et IEEE 1609, ainsi que la technologie Long Term Evolution (LTE). Avec les progrès des technologies de communication, un certain nombre d'applications innovantes se développent pour les réseaux véhiculaires. Celles-ci vont des applications de sécurité, telles que les avertissements de points morts et les infractions aux feux de circula-

tion, aux divertissements, comme le streaming multimédia, ou encore aux commodités, comme l'identification des places de parking. Dans les réseaux véhiculaires, des ressources d'edge sont disponibles sur les infrastructures environnantes pour garantir une haute qualité de service aux véhicules. La mobilité rapide des véhicules entraîne des changements fréquents dans la topologie du réseau, nécessitant une conception de politique détaillée pour s'adapter à cette dynamique.[47]

— **Réseaux mobiles à mobile (M2M)/dispositif à dispositif (D2D)** :
  — M2M (Machine-to-Machine) est une technologie clé pour l'Internet des objets (IOT), impliquant une connectivité et une communication autonomes entre les dispositifs, des capteurs, sans intervention humaine.
  — D2D (Device-to-Device) permet aux appareils de communiquer entre eux via une liaison sans fil directe sans passer par la station de base ou le réseau central.

  Avec l'avancement technologique des appareils intelligents, Les utilisateurs finaux ont maintenant accès à plus de ressources de calcul et de stockage.Ainsi, les tâches de calcul peuvent être déchargées (offloading) non seulement vers les serveurs de edge, mais aussi vers les appareils dans les réseaux D2D et M2M.[47]

2. **La couche edge** : La couche edge se situe au milieu de l'architecture hiérarchique et est composée de plusieurs serveurs edges distribués pour fournir un calcul sans fil intelligent et distribué aux utilisateurs. Les serveurs edges peuvent être déployés dans l'infrastructure réseau, tels que les stations de base, les unités de bord de route, les points d'accès sans fil et les passerelles, ou ils peuvent être des téléphones mobiles, des véhicules, des tablettes et d'autres appareils avec des capacités de calcul et de stockage.

   En raison de la proximité des serveurs edges avec les utilisateurs finaux, les tâches intensives en calcul et sensibles aux retards peuvent être déchargées (offloading) et accomplies avec une latence faible et une grande efficacité. Il existe trois types de ressources dans la couche edge : les ressources de communication, les ressources de stockage (caching resources) et les ressources de calcul.

   — **Les ressources de communication** concernent la bande passante, le spectre et la puissance de transmission.
   — **Les ressources de calcul** se réfèrent principalement aux cycles CPU. Son objectif est décharger le travail de l'utilisateur vers le serveur EC

qui fournit le traitement et tramsmettre le resultat final.

— **Les ressources de mise en cache (caching resources)** sont liées à la capacité de mémoire des serveurs edges.

Le principe est d'envoyer des données à stocker vers les serveurs EC. Comme les serveurs edges sont largement répartis, leurs capacités de calcul et de stockage sont généralement limitées.[47]

3. **La couche centrale du cloud** : La couche centrale du cloud comprend plusieurs serveurs haute performance avec de fortes capacités de traitement, stockage et de calcul.[8]
   Cette couche peut exploiter des techniques avancées telles que l'exploration de données et le big data.
   Grâce à leur grande capacité de calcul et à leurs ressources de stockage suffisantes, les serveurs cloud peuvent traiter des applications tolérantes aux retards et stocker des contenus plus volumineux ou moins populaires. De plus, la couche centrale du cloud peut gérer et contrôler efficacement plusieurs serveurs edges et leur fournir des connexions sécurisées.[47]

la Figure 1.7 représente l'architecture hiérarchique qui divise verticalement le système MEC en trois couches.

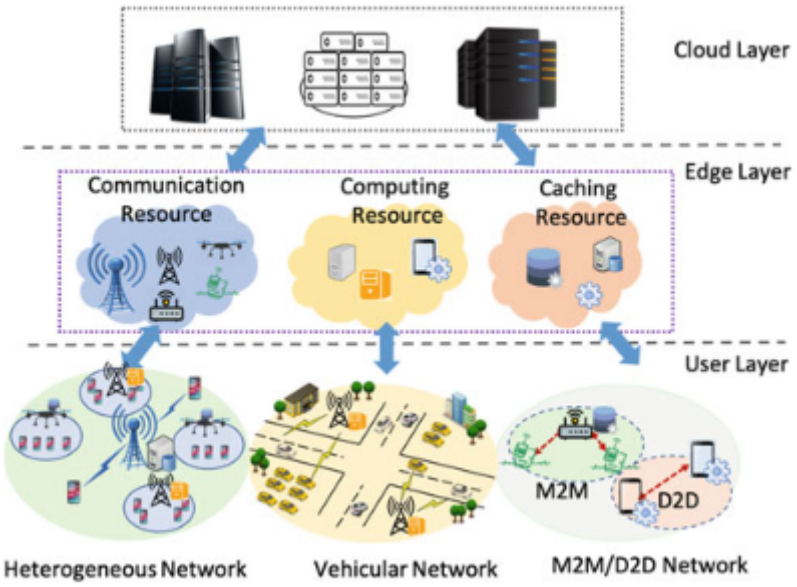

FIGURE 1.7 – Architecture de Mobile Edge Computing.

1.4.3 Applications dans les réseaux MEC

Ces dernières années, la demande de déchargement de tâches informatiques dans les réseaux MEC a considérablement augmenté en raison des progrès dans la communication et la technologie. Voici quelques cas d'utilisation importants qui nécessitent une énorme demande pour décharger (offloading) des tâches gourmandes en calcul dans les réseaux MEC.

1. **Unmanned Aerial Vehicle (UAV)** : La flexibilité des UAV les rend idéaux pour un déploiement auxiliaire dans les réseaux terrestres afin de gérer le trafic Internet mondial à grande échelle.
   Les réseaux MEC pris en charge par des UAV peuvent servir de plates-formes aériennes fournissant des services de calcul, améliorant ainsi la QoS du réseau. Les UAV peuvent également être déployés pour fournir des services de calcul stables pour les véhicules industriels dans des sites distants, compensant le manque d'infrastructure et les obstacles de signal.
2. **Autonomous Vehicle (AV)** : Les progrès rapides dans la technologie des véhicules autonomes ont entraîné une augmentation des besoins en calcul. Leur puissance de calcul limitée est souvent insuffisante pour gérer ces tâches intensives en calcul. Un des défis vitaux dans les réseaux de véhicules est d'assurer un service fiable pour les applications nécessitant une faible latence. Pour la plupart des applications de véhicules, en particulier celles liées au contrôle du trafic et à l'amélioration de la sécurité, la réactivité en temps réel est essentielle dans l'environnement en évolution rapide des réseaux de véhicules. À ce but, les AV peuvent accéder aux ressources des serveurs MEC via des liaisons véhicule-infrastructure (V2I) et décharger des tâches pour une réponse rapide. De plus, une stratégie de déchargement de données limitée à k sauts est conçue pour les véhicules hors de portée d'une Unité en bord de route (RSU). Cette stratégie utilise des chemins Véhicule à Véhicule (V2V) à plusieurs sauts, dans le but d'améliorer l'efficacité du déchargement de données.[44]
3. **Industry Internet of Things (IIoT)** : L'IIoT, ou l'Internet Industriel des Objets, connaît une croissance rapide avec l'émergence de l'industrie 4.0. Les services IIoT, tels que les robots de secours en cas de catastrophe et les robots cloud dans les environnements de fabrication intelligents, exigent des ressources de calculs importantes. MEC offre des ressources de calcul distribuées en périphérie, permettant l'exécution intelligente en temps réel des services IoT industriels. Cependant, le nombre limité de serveurs MEC peut entraîner des rejets de déchargement (Offloading) pendant les périodes de forte affluence, nécessitant le développement

de mécanismes de déchargement efficaces pour réduire la latence dans les réseaux IIoT-MEC.[44]

4. **Augmented Reality/Mobile Augmented Reality (AR/MAR)** : La technologie AR peut combiner des environnements réels et virtuels et impose des défis importants tels que des exigences de faible latence et une consommation d'énergie élevée sur les systèmes de communication actuels. Pour résoudre ce problème, l'intégration de la MEC avec l'AR offre une solution, permettant le déchargement des tâches intensives en calcul vers le MEC pour une exécution à faible latence et une meilleure efficacité énergétique.[44]
5. **Villes intelligentes** : Les villes intelligentes utilisent les systèmes IoT pour améliorer les services urbains et les infrastructures, mais la gestion efficace des ressources dans de vastes réseaux reste un défi. Les réseaux MEC peuvent aider en rapprochant les ressources des appareils, tout en traitant efficacement les tâches structurées pour garantir une exécution rapide et fiable. Par exemple, dans les voyages, les tâches telles que la réservation de vols et d'hôtels peuvent être exécutées en parallèle grâce à l'EC. [44]
6. **Soins de santé** : La technologie IoT a révolutionné les soins médicaux en permettant le suivi de la santé à distance via des dispositifs portables intelligents. L'intégration de MEC dans ces plateformes médicales offre une solution pour répondre aux besoins de portabilité et de faible latence des dispositifs portables. De plus, en offloading les tâches de calcul vers les serveurs MEC, on peut prolonger la durée de vie des batteries des dispositifs portables, tout en permettant aux patients de surveiller leur santé sans contact physique avec les médecins.[44]

Voici d'autres applications illustrées dans la Figure 1.8 qui démontrent le offloading de calcul dans les réseaux MEC.

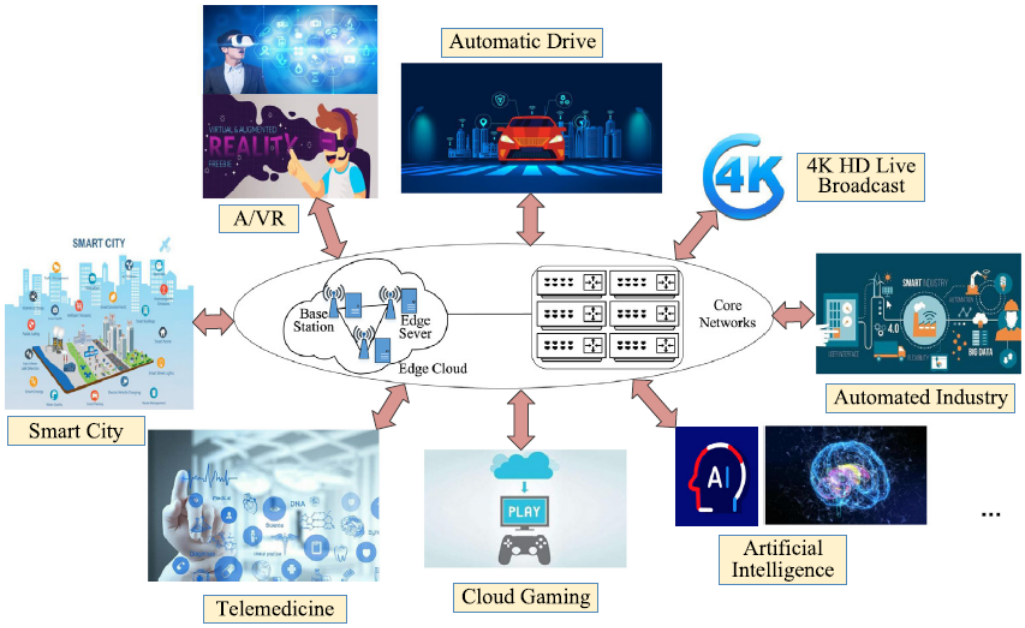

FIGURE 1.8 – Cas d'utilisation du offloading de calculs dans les réseaux MEC.

1.4.4 Computation Offloading dans les réseaux MEC

Computation offloading ou le déchargement de calcul est une technologie clé dans le MEC, visant à décharger (offloafing) des tâches intensives en calcul et sensibles à la latence vers des serveurs Edge riches en ressources et/ou des serveurs de cloud pour les traiter.
Cette approche peut aider à prolonger la durée de vie de la batterie des appareils mobiles et à réduire la latence de traitement des tâches.[47] Les applications émergentes en 5G/6G dépendent également de cette technologie pour une fourniture efficace de services aux utilisateurs.
L'industrie et le monde académique ont mené de nombreuses recherches sur les méthodes de offloading de calculs dans les réseaux MEC, utilisant diverses approches et techniques.[15]

1.4.4.1 Décision de Computation Offloading

Les problèmes clés de Computation Offloading résident dans la décision de savoir s'il faut décharger, la quantité de la tâche de calcul à décharger et vers quel serveur décharger. Fondamentalement, le offloading de calcul peut entraîner les deux types de décisions suivants :

— **Binary Offloading** : Pour le mode binaire, l'ensemble de données d'une tâche doit être exécuté totalement soit localement, soit à distance sur un serveur EC.
Lorsque l'appareil ne choisit pas le déchargement (c'est-à-dire, l'exécution locale), le délai d'exécution de la tâche ne comprend que le temps de calcul local de la tâche.
Lorsque l'appareil choisit le déchargement, le délai d'exécution de la tâche comprend trois parties : (1) le temps de transmission sans fil de la tâche de calcul de l'appareil vers le serveur Edge, (2) le temps de calcul de la tâche passé sur le serveur Edge , et (3) le temps de transmission sans fil du résultat du calcul du serveur Edge vers l'appareil.
De même, lorsque l'appareil ne décharge pas la tâche, la consommation d'énergie totale nécessaire pour achever la tâche comprend uniquement la consommation d'énergie de calcul locale de la tâche.
Si l'appareil décharge la tâche de calcul, la consommation d'énergie totale se compose de deux parties : la consommation d'énergie de la transmission sans fil de l'appareil vers le serveur Edge et la consommation d'énergie du calcul sur le serveur Edge. En pratique, le déchargement binaire est plus facile à mettre en œuvre et convient aux tâches simples qui ne peuvent pas être partitionnées.[25]

— **Partial Offloading** : Dans le cas de du déchargement partiel, une partie de la tâche de calcul est traitée localement tandis que le reste est déchargé sur un serveur Edge, et donc la partition de la tâche est autorisée. Ainsi, une tâche est d'abord divisée en plusieurs composants, qui sont ensuite déchargés vers des serveurs EC ou exécutés localement.
Le déchargement partiel est favorable pour certaines tâches complexes composées de plusieurs segments parallèles.[25]

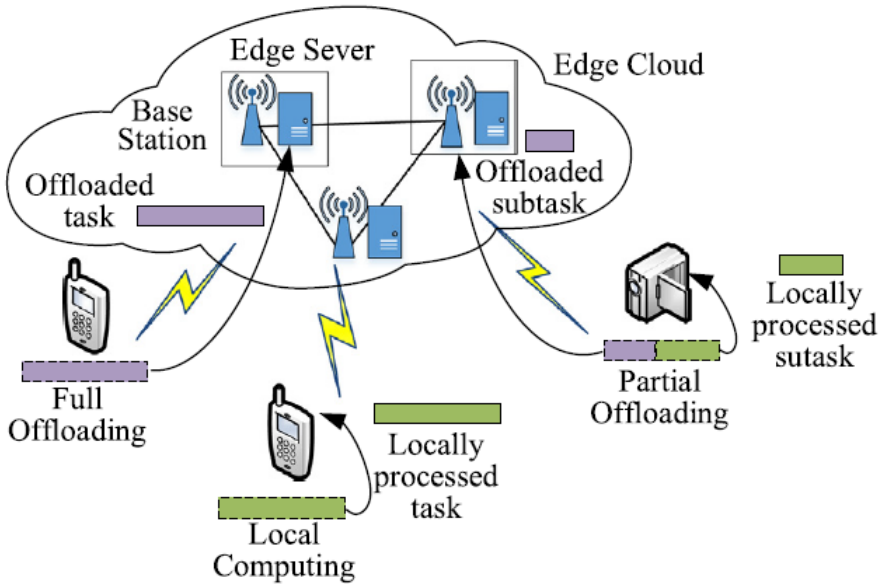

FIGURE 1.9 – Déchargement binaire et déchargement partial.

1.4.4.2 Processus d'exécution du déchargement de calcul pour les utilisateurs

Le processus de déchargement de calcul pour les utilisateurs implique cinq étapes suivantes :[15]

1. **Division des tâches** : Les tâches intensives en calcul sont divisées en plusieurs sous-tâches en fonction de la nature des tâches spécifiques.
2. **Prise de décision de déchargement** : Dans ce processus, les tâches à forte intensité de calcul sont évaluées pour déterminer si un déchargement est nécessaire, ainsi que la manière et l'endroit où le décharger.
3. **Déploiement des tâches** : Cela signifie que les sous-tâches sont déployées sur des serveurs Edge avec ces ressources.
4. **Traitement des tâches** : Il s'agit d'exécuter les tâches sur les serveurs edge.
5. **Renvoi des résultats** : Après que les serveurs edge ont exécuté une tâche, les résultats sont renvoyés aux utilisateurs.

Le processus général d'exécution du déchargement des calculs pour les utilisateurs est illustré à la Figure 1.10.

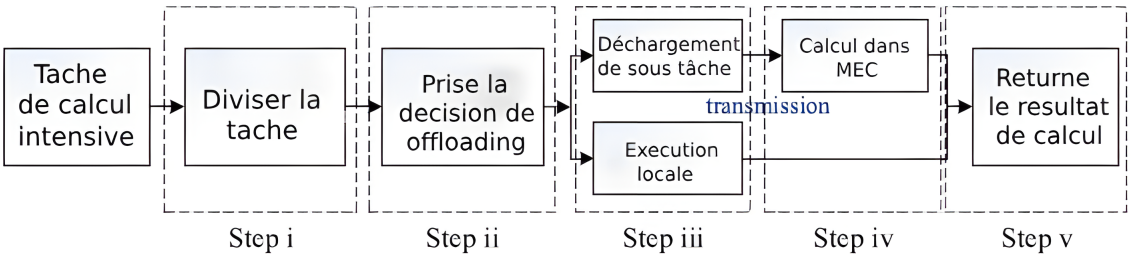

FIGURE 1.10 – Processus d'exécution du déchargement de calcul pour les utilisateurs.

1.4.4.3 Défis soulignés

Pendant le processus de déchargement, le problème clé est de déterminer s'il faut décharger et où décharger en fonction de la disponibilité spécifique des ressources des réseaux MEC.
De nombreux facteurs doivent être pris en compte pour la prise de décision de déchargement, notamment l'emplacement et la mobilité des appareils des utilisateurs, les conditions du réseau, l'hétérogénéité des dispositifs réseau et les préférences des utilisateurs. Les conditions du réseau incluent la qualité du canal sans fil, la capacité de bande passante et les interférences du signal. Les préférences des utilisateurs déterminent la manière dont les tâches sont déchargées, comme décrit ci-dessous.

1.4.4.4 Métriques d'évaluation de déchargement de calculs dans les réseaux MEC

Dans la recherche, différents critères sont utilisés pour évaluer l'efficacité des systèmes de déchargement des calculs. Ces critères, qu'ils soient centrés sur l'utilisateur ou sur le service, aident à formuler des problèmes de déchargement des calculs pour améliorer la qualité de l'expérience (QoE) et de service (QoS). Certains problèmes évaluent le système avec une seule métrique, tandis que d'autres prennent en compte plusieurs paramètres de métrique. Les chercheurs développent des métriques basées sur divers critères, tels que la latence, consommation d'énergie, etc.

1. **Latence (Délai)** : Lorsque les tâches sont exécutées localement, le délai dépend uniquement du traitement sur les appareils locaux. Cependant, lorsque les tâches sont déchargées vers les serveurs Edge, le délai comprend le temps de transmission des données vers les serveurs, le temps de

traitement sur les serveurs et le temps de transmission des résultats. Par conséquent, le délai causé par la déchargement des tâches intensives en calcul vers les serveurs Egde affecte directement la QoS des utilisateurs.[15]

2. **Énergie** : Réduire la consommation d'énergie est essentiel pour la durabilité des appareils. Elle dépend généralement de l'énergie utilisée pour la transmission des données vers les serveurs Edge et la réception des résultats retournés. Cela implique d'optimiser la puissance de transmission et de prendre en compte la qualité des canaux sans fil. L'utilisation de différents états de puissance pour les serveurs Edge, tels que l'état actif et l'état de veille, peut réduire la consommation totale d'énergie en mettant les serveurs sous-utilisés en mode veille.[43] La répartition de la puissance de transmission des appareils peut être transformée en une optimisation convexe pour une meilleure efficacité.[24] De plus, la technologie NOMA (Non-Orthogonal Multiple Access) est reconnue comme une technique clé dans les réseaux cellulaires 5G, permettant à plusieurs utilisateurs de communiquer avec une seule station de base simultanément.
3. **Bande passante** : L'utilisation de la bande passante est cruciale pour la performance de déchargement des calculs. Une allocation stratégique de la bande passante est nécessaire pour optimiser les performances, car elle influe sur le débit de transmission, la consommation d'énergie des appareils utilisateurs, et l'exécution à distance ou locale.[10]
4. **Coût** : La métrique du coût système dans la mise en œuvre de MEC reconnaît deux dimensions telles que le coût de performance système et le coût de déploiement.
   Le coût de performance système prend en compte le coût généré par un seul paramètre ou une combinaison de ces paramètres, tels que le délai/latence, consommation d'énergie, le temps de réponse, utilisation de la bande passante, etc.
   Le coût de déploiement englobe les coûts liés au développement et au déploiement d'un système fonctionnel, y compris les ressources physiques telles que le développement de l'infrastructure, les besoins en réseau et en matériel, l'installation de points d'accès, le déploiement de serveurs, etc. L'objectif est de minimiser l'utilisation des ressources physiques tout en maximisant les services.[10]
5. **Temps de réponse** : Cette métrique évalue le temps nécessaire pour qu'un système de communication envoie une demande à un serveur distant, traite cette demande, et renvoie la réponse au client. Elle est cruciale pour les applications nécessitant des réponses rapides. Dans le contexte

de déchargement de calcul, elle mesure la rapidité avec laquelle un serveur de périphérie peut traiter efficacement une tâche déchargée.[10]

1.4.4.5 Classifications de déchargement dans les réseaux MEC

Dans cette partie, nous organisons les travaux précédents sur le déchargement de calcul en différentes catégories. Shan et al. [39] classifient ces travaux en fonction des objectifs de déchargement, tandis que Shi et al. [41] les divisent en déchargement statique et dynamique.
Notre approche propose deux autres perspectives de classification : le flux de offloading et les scénarios de offloading, Comme le montre le Schéma 1.11.

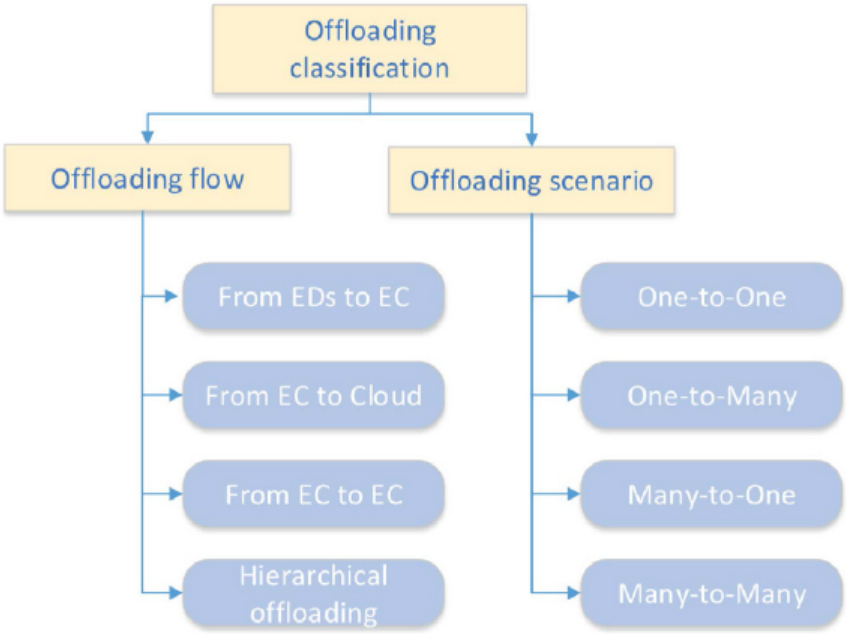

FIGURE 1.11 – Classification de offloading.

1. **Selon le flux de déchargement**
   - **De utilisateur final (ED) à EC** : Dans cette première catégorie, les ED et les EC forment un système où les ED exécutent localement ou déchargent des tâches de calcul vers les EC.
   - **De EC au CC** : Dans cette deuxième catégorie, EC est considérée comme un complément au CC. Les appareils finaux (ED) envoient toujours leurs tâches à un serveur EC proche, qui décide d'exécuter les tâches reçues lui-même ou de les décharger vers le cloud. Bien qu'il existe un principe selon lequel les tâches sensibles à la latence doivent être traitées par l'EC et que les tâches tolérantes au retard doivent être

transférées vers le cloud, l'optimisation de la distribution des tâches peut encore améliorer les performances.

— **D'un serveur EC à un autre (autres)** : Dans la troisième catégorie, plusieurs serveurs EC forment un système Edge (ou cluster Edge). Lorsqu'un serveur EC reçoit une tâche de calcul, il décide de l'exécuter localement ou de la décharger sur d'autres serveurs EC au sein du même cluster. Étant donné que le déchargement des calculs est effectué au sein d'un cluster, la formation du cluster (comment regrouper les serveurs EC) a un impact direct sur les performances du déchargement.

— **Déchargement hiérarchique** : Dans le déchargement hiérarchique, une tâche de calcul peut être déchargée vers différentes couches du système informatique. Par exemple, dans une hiérarchie à trois niveaux, un ED peut décider d'accomplir une tâche localement ou de la décharger vers un EC ou vers un cloud (Guo et al., 2018b ; Liu et al., 2018b). Si une tâche est partitionnable, elle peut être divisée en trois parties : une partie pour l'exécution locale, une partie pour l'EC, une partie pour le cloud.[26] Le déchargement hiérarchique peut également se produire dans un système à deux niveaux composé d'EC et d'ED. Un ED peut exécuter des tâches de calcul localement, ou les décharger vers un autre ED, ou les décharger vers le serveur EC, formant ainsi un déchargement hiérarchique.[48]

2. **Selon les scénarios de déchargement**

— **Un-à-un** : Dans le cadre de déchargement un à un, seule une entité ED décide de décharger une tâche de calcul vers une autre entité (par exemple, un serveur EC). Les développeurs ont développé un système composé d'un seul ED et d'un seul serveur EC pour démontrer comment optimiser les performances de déchargement.[28]
Ce scénario est généralement applicable dans des environnements limités, tels que de petits bureaux, où un serveur MEC coopère avec un seul ED pour établir un réseau MEC à haute vitesse et faible latence.[44]

— **Un-à-plusieurs** : Dans le cadre de déchargement un à plusieurs, de nombreux serveurs EC sont disponibles. L'ED décide s'il doit décharger et décide également vers quel(s) serveur(s) la tâche doit être déchargée. Si la tâche peut être partitionnée, l'ED doit distribuer les composants à plusieurs serveurs EC.[41]
cette disposition du réseau peut équilibrer la charge de travail entre les serveurs. Cette configuration pourrait apparaître dans certains scénarios d'application spécifiques. Par exemple, un BS peut fournir des inter-

faces réseau, et plusieurs serveurs peuvent exécuter des tâches de calcul (par exemple, simulation scientifique ou traitement d'images) en parallèle pour améliorer l'efficacité de traitement. [44]

— **Plusieurs-à-un** : Dans le cadre du déchargement plusieurs vers un, plusieurs ED déchargent leurs tâches vers un seul serveur EC. La modélisation de ce type de scénario doit prendre en compte toutes les entités, c'est-à-dire que la décision doit être prise dans le but d'optimiser l'ensemble du système. Comme il n'y a qu'un seul serveur, il est logiquement que le serveur soit le décideur pour tous les ED.
Une BS sert de décideur, Elle recueille les informations nécessaires de l'environnement et de tous ses ED, puis prend des décisions pour eux.[45]

— **Plusieurs-à-plusieurs** : Le scénario de déchargement Plusieurs à plusieurs est plus complexe. Il s'agit d'une combinaison des scénarios de déchargement un à plusieurs et plusieurs à un. Les informations provenant à la fois des terminaux (ED) et des serveurs EC sont nécessaires pour la prise de décision.[9]
Par conséquent, le modèle de plusieurs ED et serveurs peut mieux simuler les réseaux MEC du monde réel. En d'autres termes, les tâches de calcul peuvent être déchargées vers les serveurs EC appropriés, en fonction des stratégies de déchargement. Ce modèle évite non seulement la surcharge d'un seul serveur, mais bénéficie également de la coopération de plusieurs serveurs.[44]

1.4.5 Comparaison etre le MCC et MEC

La principale limitation du MCC réside dans la longue distance de transmission entre les appareils mobiles et le cloud, entraînant ainsi des latences d'exécution prolongées qui ne peuvent pas répondre aux exigences de temps des applications nécessitant une faible latence.
Il existe des distinctions significatives entre les systèmes MEC et les systèmes MCC. Alors que le MEC intègre le CC dans les réseaux mobiles pour offrir des capacités de calcul et de stockage aux utilisateurs finaux en Edge.
Les principales différences entre le MCC et le MEC peuvent être résumées dans le tableau 1.4 :

Aspect	MCC	MEC
Emplacement des serveurs	Sur Internet	Dans le réseau Edge
Latence	Élevée	Faible
Modèle de distribution	Centralisé	Distribué
Architecture système	Configuration sophistiquée	Configuration simple, répartition dense
Serveurs physiques	Capacités de calcul et de stockage élevées, situées dans des centres de données à grande échelle	Capacités limitées
Distance de transmission	Éloignée des utilisateurs, de quelques kilomètres à des milliers de kilomètres	Assez proche des utilisateurs, de quelques dizaines à quelques centaines de mètres
Risques d'attaques	Plus élevés (chemin plus long)	Moins élevés (chemin plus court)
Caractéristiques des applications	Tolérance aux retards, intensives en calcul, par exemple Facebook, Twitter	Sensibles à la latence, peu intensives en calcul, par exemple conduite autonome, jeux en ligne

TABLE 1.4 – Comparaison entre MCC et MEC.[47]

1.5 Conclusion

En résumé, ce chapitre a présenté les concepts du Cloud Computing, de l'Edge Computing et du Mobile Edge Computing (MEC). Nous avons examiné leurs architectures, services, et applications, ainsi que les avantages du traitement des données en périphérie. Le MEC a été présenté comme une solution prometteuse pour améliorer l'efficacité des réseaux, notamment à travers le déchargement de calculs. Ces technologies jouent un rôle clé dans le développement des systèmes informatiques modernes.
Dans le chapitre suivant, nous entamerons les notions de l'intelligence Artificielle et Apprentissage Automatique.

Chapitre 2

Intelligence Artificielle et Apprentissage Automatique : Fondements et Techniques

2.1 Introduction

L'intelligence artificielle (IA) et la machine learning (ML) sont des disciplines qui visent à rendre les systèmes informatiques capables de résoudre des tâches complexes de manière autonome.

Dans ce chapitre, nous commencerons par introduire les concepts fondamentaux de l'IA, en décrivant ses principales branches et son évolution. Par la suite, nous examinerons les bases du ML ainsi que ses différents types d'apprentissages. Enfin, nous approfondirons l'apprentissage par renforcement, en détaillant ses composants, ses approches et ses applications pratiques.

2.2 Intelligence Artificielle

2.2.1 Définition

L'intelligence artificielle (IA) représente à la fois la science et l'ingénierie nécessaires à la création de machines intelligentes, en particulier de programmes informatiques intelligents. Cette discipline est étroitement liée à l'utilisation de l'informatique pour comprendre l'intelligence humaine.[29]
Elle implique un ensemble de composants techniques et algorithmiques d'apprentissage permettant à un système informatique de reproduire le comportement humain et de prendre des décisions similaires à celles des humains.[35]
Aujourd'hui, de nombreux spécialistes de l'IA définissent celle-ci comme l'étude des agents intelligents. L'agent intelligent interagit avec l'environnement, il peut identifier l'état d'un environnement à travers ses capteurs, puis agir sur cet état à travers ses actionneurs.[13]

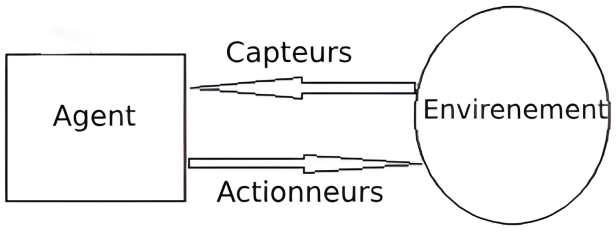

FIGURE 2.1 – Cycle de perception de l'IA.

2.2.2 Évolution historique de l'IA

L'IA est née officiellement en 1956 lors d'une conférence à Dartmouth (USA) intitulée "Dartmouth Summer Research Project on Artificial Intelligence" même si les recherches ont en réalité commencé dès la fin de la Seconde Guerre mondiale. Cette conférence est historique car elle a rassemblé en un même lieu presque toutes les figures emblématiques de l'IA, notamment John McCarthy, Herbert Simon, Allen Newell, Claude Shannon et Marvin Minsky. Ces pionniers aspiraient à créer des machines capables d'égaler l'intelligence humaine en exploitant les nouveaux développements de l'informatique naissante. C'était un projet d'une ambition folle. Malgré les énormes défis auxquels ils étaient confrontés, ces chercheurs, ainsi que de nombreux autres, ont rapidement transformé leurs idées en résultats révolutionnaires qui ont marqué l'histoire de l'informatique.[34]

La table 2.1 offre un bref résumé de quelques étapes de l'histoire de l'IA. Ce résumé n'est pas exhaustif, mais il devrait donner une idée de l'évolution de l'IA et de sa présence croissante dans notre vie quotidienne.

Année	Jalons de l'IA
1956	La conférence de Dartmouth "Dartmouth Summer Research Project on Artificial Intelligence"
1966	Le chatbot ELIZA a été créé par Joseph Weizenbaum
1986	Le van robotisé autonome a été créé par Ernst Dickmanns.
1994	Le programme de dames Chinook remporte le championnat contre un être humain.
1995	Le chatbot ALICE a été développé par Richard Wallace
1997	Le programme Deep Blue d'IBM bat le champion du monde d'échecs Garry Kasparov
2000	La FDA approuve le premier chirurgien robotique pour les procédures laparoscopiques.
2002	L'aspirateur automatique Roomba est introduit par la société iRobot Corporation.
2011	Apple lance SIRI (Speech Interpretation and Recognition Interface) sur l'iPhone 4S.
2014	Amazon lance son propre assistant de reconnaissance vocale : Alexa.
2017	AlphaGo de Google remporte trois parties contre le meilleur joueur de Go au monde, Ka Jie.
2018	Facebook utilise l'IA pour aider à filtrer le contenu explicite.
2019	L'IA améliore la détection du cancer du poumon, surpassant les radiologistes.
2020	Les taxis sans conducteur de Waymo commencent au sud-ouest de Phoenix.
2020	OpenAI a lancé GPT-3, un modèle de traitement du langage naturel,
2024	OpenAI a lancé GPT-4,5, un modèle de traitement du langage naturel avancé, extension de GPT-4

TABLE 2.1 – Évolution de l'IA.

2.2.3 Compréhension du fonctionnement de l'IA

Les systèmes d'IA traitent de vastes ensembles de données, les structurant en objets compatibles avec les technologies d'algorithme de traitement rapide, permettant au logiciel d'apprendre automatiquement à partir des caractéristiques des données.
L'IA se concentre sur trois compétences cognitives :

- **Les processus d'apprentissage**, qui impliquent la collecte de données et le développement d'algorithmes visant à les transformer en connaissances exploitables, fournissant ainsi aux dispositifs informatiques des instructions détaillées pour accomplir une tâche spécifique.
- **Les processus de raisonnement**, qui consistent à choisir l'algorithme approprié pour atteindre un résultat souhaité.
- **Les processus d'auto-correction**, qui visent à améliorer en permanence les algorithmes pour garantir la précision maximale des résultats fournis.

2.2.4 Branches de l'IA

L'IA englobe plusieurs domaines différents, chacun caractérisé par ses propres ensembles d'algorithmes spécifiques. Il peut être divisée en deux grands domaines : Machine Learning (ML) et Deep Learning (DL), qui sont tous les deux des sous-domaines de l'IA.
Pour simplifier, la Figure 2.2 illustre les relations et les interactions entre trois disciplines :

- **L'intelligence artificielle (IA)** consiste à simuler les réponses d'experts dans des domaines spécifiques.
- **L'apprentissage automatique** est une branche d'IA qui repose sur des techniques statistiques pour tirer des conclusions à partir des données disponibles. Il ajuste ses algorithmes en fonction des données, mais nécessite l'expertise humaine pour la préparation et la sélection de ces données.
- **L'apprentissage profond** ou Deep Learning (DL) est une méthode avancée de ML qui utilise des réseaux de neurones pour extraire des informations pertinentes des données et adapter ses algorithmes en conséquence.

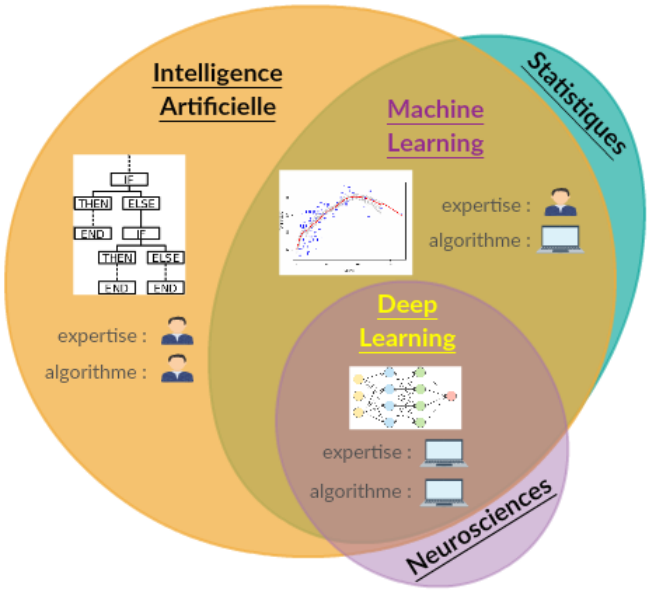

FIGURE 2.2 – Composants de l'IA.

2.3 Machine Learning

2.3.1 Définition

L'apprentissage automatique, ou Machine Learning (ML), désigne un ensemble de techniques qui permettent aux ordinateurs d'acquérir des connaissances à partir de données, afin d'exécuter une tâche, faire des prédictions et de les améliorer par la suite.
ML marque un changement de paradigme par rapport à la programmation normale, où toutes les instructions doivent être spécifiées explicitement à l'ordinateur, vers une approche indirecte qui se produit en fournissant des données (apprendre sans être explicitement programmés).[32]
Tom Mitchell a formulé une définition plus précise, décrivant un programme informatique comme apprenant par l'expérience (E) par rapport à une tâche (T) et une mesure de performance (P) ; Si la performance sur la tâche T, évaluée par P, s'améliore avec l'expérience E, alors le programme est qualifié de programme de ML. Dans l'exemple du jeu de dames, l'expérience E consistait à faire jouer au programme des parties contre lui-même, la tâche T étant de jouer aux dames, et la mesure de performance P étant la probabilité de gagner la prochaine partie contre un nouvel adversaire.[13]

2.3.2 Types de Machine Learning

Les types de ML peuvent être classés en quatre catégories basées sur différents algorithmes pour résoudre des problèmes.
Le choix de l'algorithme dépend du type de problème, du nombre de variables et du modèle le plus adapté.
La Figure 2.3 présente ces types d'apprentissage.

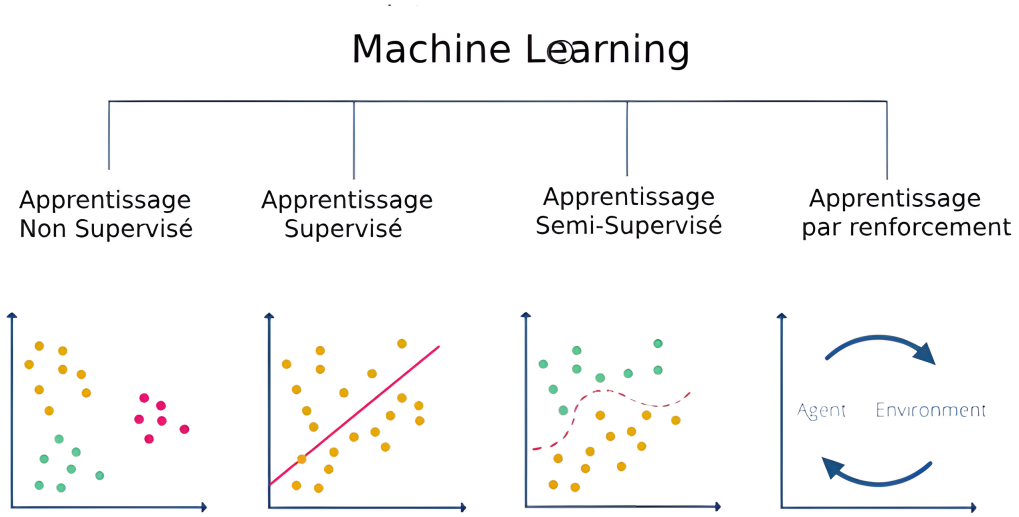

FIGURE 2.3 – Aperçu des catégories de ML.

2.3.2.1 Apprentissage Supervisé

Dans le cadre de l'apprentissage supervisé (Supervised Learning), les données fournies sont étiquetées, ce qui signifie que pour chaque ensemble de données en entrée (X), nous fournissons une étiquette réelle (Y) afin que l'algorithme puisse utiliser ces valeurs pour apprendre à prendre une décision pour de nouvelles entrées non étiquetées.
L'objectif principal de cette méthode est de trouver une fonction qui traite les entrées (X) et les sorties (Y), appelée modèle de prédiction, c'est à dire, utiliser un algorithme pour apprendre la fonction de mapping de l'entrée à la sortie.

$$Y = f(X)$$

Les données d'entrée peuvent être des valeurs numériques, alphanumériques ou des images. Les données de sortie (variable à prédire) peuvent être une variable discrète ou une variable continue. L'ensemble de données est généralement divisé en deux ensembles : l'ensemble d'entraînement (Training Set) et l'ensemble de test (Test Set). L'ensemble d'entraînement est utilisé pour apprendre

la relation entre l'entrée et la sortie, tandis que l'ensemble de test est utilisé pour valider le modèle en mesurant sa précision.

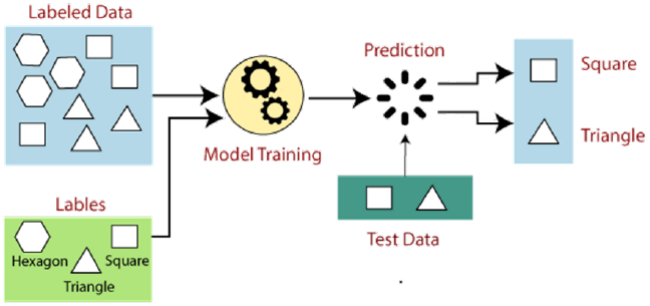

FIGURE 2.4 – Structure d'apprentissage supervisé.

L'objectif de l'apprentissage supervisé se concentre principalement sur deux catégories d'algorithmes :

1. **Classification** :
   - Un problème d'apprentissage supervisé dans lequel l'espace des étiquettes est binaire, autrement dit Y = {0,1} est appelé un problème de classification binaire.
     par exemple : Prédire si un email est un spam (classe y = 1) ou non (classe y = 0) selon le nombre de liens présent dans l'email (x) [5]
   - Un problème d'apprentissage supervisé dans lequel l'espace des étiquettes est discret et fini, autrement dit Y = {1,2,...,C} est appelé un problème de classification multi-classe. C est le nombre de classes.
     par exemple : Identifier en quelle langue un texte est écrit ; Identifier les objets présents sur une photographie.[5]

   L'image 2.5 illustre les deux types de classifications.
   Les cercles bleus symbolisent une catégorie (par exemple, les e-mails authentique), tandis que les croix rouges peuvent représenter des SPAM.
   À droite, l'image illustre une classification multi-classes, car elle comporte trois catégories possibles (triangles, croix et carrés).

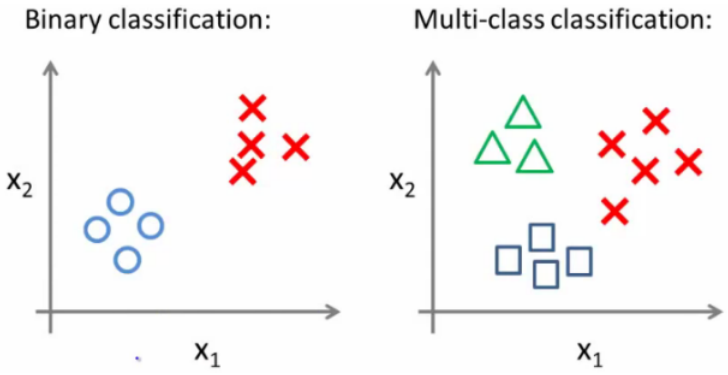

FIGURE 2.5 – Types de classification.

2. **Régression** :
   Dans les problèmes de régression, on cherche à prédire la valeur d'une variable continue, c'est-à-dire une variable qui peut prendre une infinité de valeurs.
   par exemple : Prédire le prix d'un appartement (x) selon sa surface habitable (y) ; Prédire la quantité d'essence consommée (x) selon la distance parcourue (y).[37]
   Les formes d'algorithmes de régression varient en fonction du modèle souhaité, allant de la régression linéaire simple, qui consiste à trouver la meilleure droite qui s'approche le plus de données d'entrée, à des formes plus complexes comme la régression polynomiale, régression crête, etc.

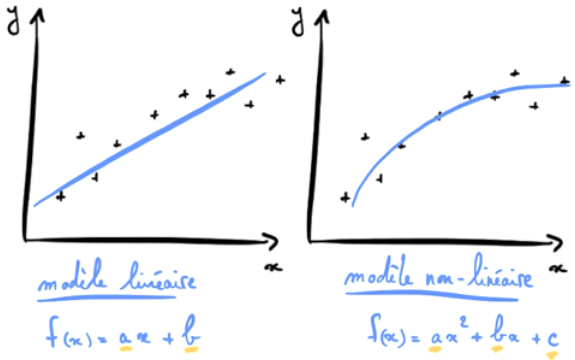

FIGURE 2.6 – Formes de régression.

2.3.2.2 Apprentissage Non Supervisé

Dans le cadre de l'apprentissage non supervisé (Unsupervised Learning), les données ne sont pas étiquetées. Il s'agit alors de modéliser les observations pour mieux les comprendre. Il implique uniquement des données d'entrée (X) sans variables de sortie correspondantes.

Son objectif est de modéliser la structure ou la distribution sous-jacente des données pour en tirer des informations. Les algorithmes sont laissés à eux-mêmes pour découvrir et présenter la structure intéressante des données.
Il se divise en deux catégories principales d'algorithmes : les algorithmes de regroupement et les algorithmes d'association.

1. **Partitionnement (Clustering)** : Le clustering dans l'apprentissage non-supervisé est une technique qui consiste à regrouper des données similaires entre elles en fonction de leurs caractéristiques communes.
   L'objectif est de diviser un ensemble de données en groupe de sorte que les ensembles de données appartenant aux mêmes groupes se ressemblent davantage que ceux d'autres groupes. Par exemple : Identifier des groupes parmi les patients présentant les mêmes symptômes permet d'identifier des sous-types d'une maladie, qui pourront être traités différemment.[5]
2. **Association** : L'association consiste à découvrir des relations intéressantes entre les différentes variables dans un ensemble de données.
   Par exemple, en analysant les transactions d'achat dans un supermarché pour identifier des combinaisons d'articles qui sont souvent achetés ensemble.
3. **Réduction de dimension** : La réduction de dimension constitue une autre catégorie essentielle de problèmes en apprentissage non supervisé. Son objectif est de trouver une représentation des données dans un espace de dimensions inférieures à celui de l'espace d'origine. Cette approche vise à réduire les temps de calcul, l'espace mémoire requis pour stocker les données, et souvent à améliorer les performances d'algorithme d'apprentissage supervisé entraîné ultérieurement à ces données.

Voici un exemple sur le principe d'apprentissage non supervisé illustré dans la Figure 2.7.

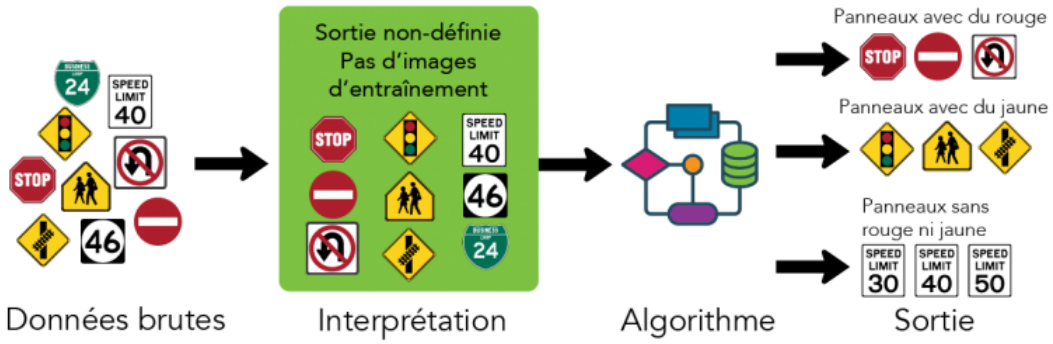

FIGURE 2.7 – Exemple d'apprentissage non supervisé.

2.3.2.3 Apprentissage Semi-Supervisé

Comme son nom l'indique, l'apprentissage semi-supervisé (Semi Supervised Learning) se situe entre les techniques d'apprentissage supervisé et non supervisé. Ces algorithmes utilisent une combinaison de données étiquetées et non étiquetées pour l'entraînement. Généralement, on dispose d'une petite quantité de données étiquetées et d'une grande quantité de données non étiquetées.
La procédure de base consiste d'abord à regrouper les données similaires à l'aide d'un algorithme non supervisé, puis à utiliser les données étiquetées existantes pour étiqueter les données non étiquetées restantes.[36]

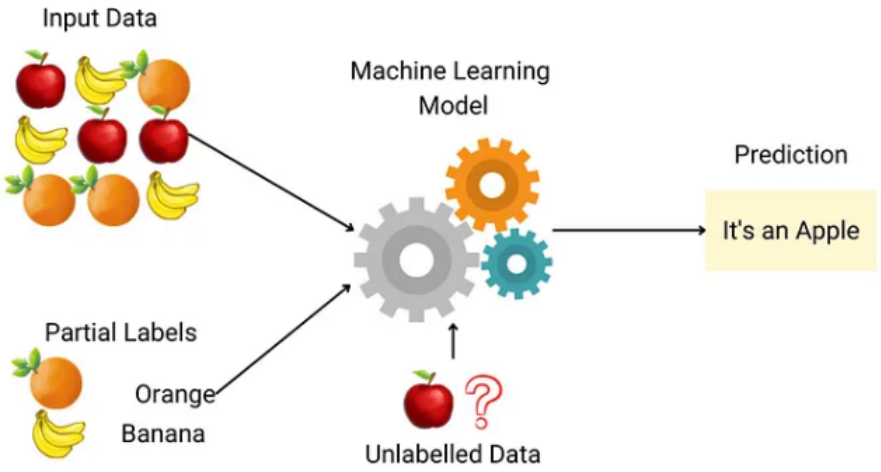

FIGURE 2.8 – Exemple d'apprentissage semi supervisé.

2.3.2.4 Apprentissage par Renforcement

Ce type d'apprentissage sera détaillé dans la section suivante.

2.4 Apprentissage par Renforcement

Dans le cadre de l'apprentissage par renforcement, ou Reinforcement Learning (RL), le système d'apprentissage repose sur un agent autonome capable d'interagir avec son environnement et d'accomplir des actions. En retour de ces actions, il obtient une récompense, qui peut être positive si l'action était un bon choix, ou négative dans le cas contraire (une pénalité).[5]

RL implique un agent qui explore, interagit et apprend de l'environnement dynamique pour trouver la meilleure séquence d'actions générant le résultat optimal pour un problème donné.

2.4.1 Processus de décision markovien

Le processus décisionnel de Markov (Markov Decision Process, MDP) constitue la base de la structuration des problèmes abordés par l'apprentissage par renforcement. Il permet de formaliser la prise de décision séquentielle.
L'objectif d'un MDP ou d'un agent d'apprentissage par renforcement est de maximiser la récompense en trouvant une politique optimale. Ce processus de prise de décision, suivant une politique dans chaque état, se réduit à une chaîne de Markov.

Un MDP est défini par un tuple $\langle S, A, R, P \rangle$ où S est un ensemble fini d'états, A un ensemble fini d'actions, R est une fonction de récompense immédiate avec $R : S \times A \to R$ et P est une fonction de transition $P(s'|s, a)$ avec $P : S \times A \times S \to [0, 1]$. Une politique stationnaire π déterministe est une fonction $S \to A$ où $\pi(s)$ définit l'action à réaliser dans l'état s.[14]

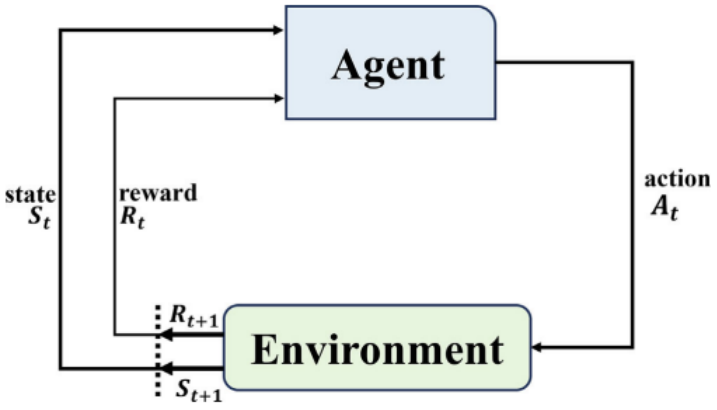

FIGURE 2.9 – Interaction entre l'agent et l'environnement dans le cadre d'un MDP.

2.4.2 Composants et Phases de RL

RL est une méthode de ML où un agent apprend à prendre des décisions en interagissant avec un environnement. Ce processus repose sur plusieurs composants clés et suit des étapes spécifiques pour maximiser les récompenses cumulées.

— **Les composants du RL**

1. **Agent** : L'entité qui prend des décisions et effectue des actions dans l'environnement (agent autonome).
2. **Environnement** : Une représentation du monde réel dans lequel l'agent exploite les observations. Il répond aux actions de l'agent et fournit des

récompenses.

3. **État (S)** : Une représentation de la situation actuelle de l'agent dans l'environnement.
4. **Action (A)** : Les mouvements que l'agent peut effectuer à partir de chaque état.
5. **Récompense (Reward, R)** : Un signal de retour que l'agent reçoit de l'environnement après avoir effectué une action, elle peut être positive ou négative.
6. **Politique (π)** : Une stratégie que l'agent utilise pour déterminer quelles actions à prendre dans chaque état. Elle peut être déterministe (une action spécifique pour chaque état) ou stochastique (une distribution de probabilités sur les actions possibles).
7. **Fonction de valeur (V)** : Une fonction qui estime la quantité totale de récompense qu'un agent peut espérer recevoir à partir d'un état donné, en suivant une politique particulière.
8. **Exploration vs Exploitation** : Un dilemme des agents RL : explorer pour découvrir de nouvelles actions ou exploiter les actions connues pour maximiser les récompenses.

— **Les phases du RL**

1. **Observation et Exploration** : L'agent observe l'état actuel de l'environnement et, à partir de cette observation, décide de l'action à entreprendre. Il examine ensuite les effets de ses actions.
2. **Récompense et Exploitation** : Après avoir exécuté une action, l'agent reçoit un retour d'information de l'environnement sous forme de récompense ou de pénalité. La récompense est positive si l'action était correcte et efficace, ou négative (pénalité) si l'action était incorrecte. L'environnement change alors d'état en conséquence. L'étape de l'exploitation consiste à utiliser ce retour d'information pour guider les actions futures de l'agent.
3. **Apprentissage** : L'agent améliore alors ses futures actions en utilisant les informations reçues, c'est l'étape d'apprentissage. Le cycle observation-action-récompense se poursuit jusqu'à ce que l'apprentissage soit terminé.
4. **Évaluation des Actions** : Grâce à la récompense reçue, l'agent peut déterminer si une action doit être répétée dans des situations similaires (si elle est bonne) ou évitée (si elle est mauvaise). Cette boucle d'interaction continue, permettant à l'agent de s'améliorer au fil du temps.

2.4.3 Classification des algorithmes du RL

Les algorithmes RL peuvent être classés en deux grandes catégories : les algorithmes sans modèle et les algorithmes basés sur un modèle.

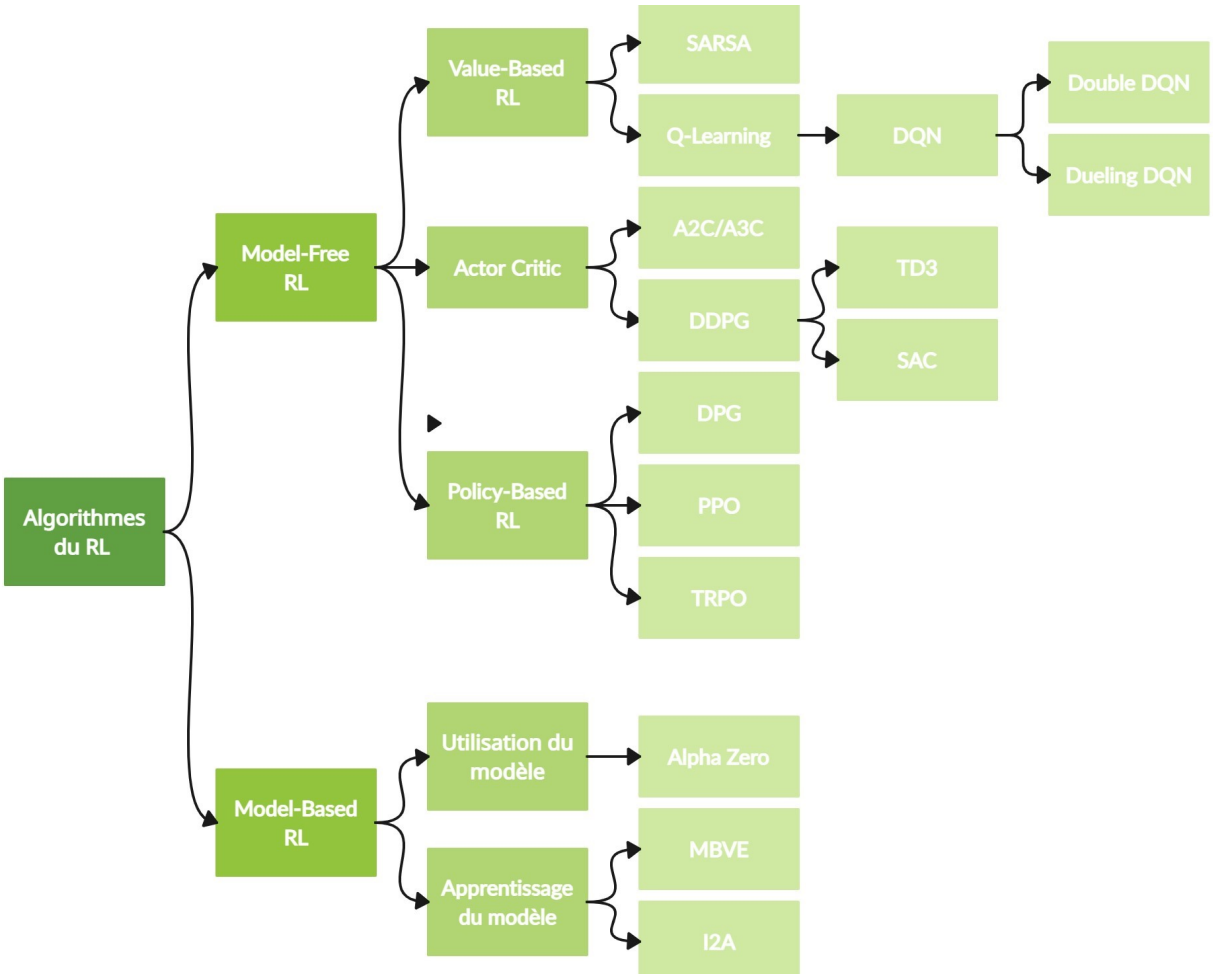

FIGURE 2.10 – Hiérarchie des algorithmes de Reinforcement Learning.

2.4.3.1 Model-Free RL

Les algorithmes sans modèle (MFRL) ne créent pas de modèle explicite de l'environnement. Ils fonctionnent plutôt avec **la méthode d'essais et d'erreurs** de l'agent, en interagissant directement avec l'environnement par des actions pour déterminer la politique optimale.

Il est préférable d'utiliser le RL sans modèle lorsque l'environnement est vaste, complexe et difficile à décrire. Cette solution est également recommandée lorsque l'environnement est inconnu et changeant.

Pour le Model-Free RL, trois classifications des méthodes existent, comme le montre la Figure 2.12.

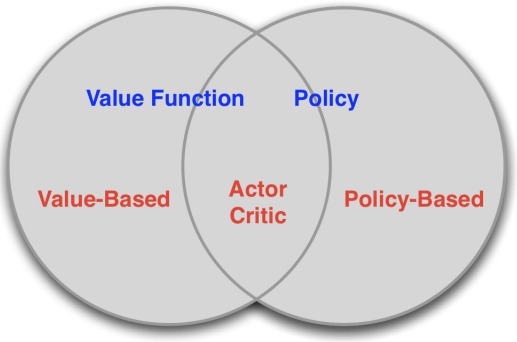

FIGURE 2.11 – Classifications des méthodes pour Model-free RL.

— **Value-based RL (Basés sur la valeur)** :

Aucune politique n'est stockée directement. Au lieu de cela, la politique est dérivée de la fonction de valeur. L'apprentissage par renforcement basé sur les valeurs construit une fonction Q pour les paires état-action. Ces valeurs Q quantifient la récompense attendue pour entreprendre une action particulière dans un état donné. La politique est dérivée de cette fonction, correspondant généralement à l'action avec la meilleure valeur Q.

Certains algorithmes courants basés sur la valeur : Quality Learning (Q-Learning) ,Deep Q-Network (DQN), State-Action-Reward-State-Action (SARSA), etc.

— **Policy-based RL (Basés sur la politique)** :

Dans la méthode basée sur les politiques, il n'y a pas de construction de fonction pour la valeur de l'état. Au lieu de cela, l'accent est mis sur l'élaboration d'une politique pour les paires état-action, définissant la probabilité d'entreprendre une action particulière dans un état donné pour maximiser la récompense. Cette probabilité peut être soit stochastique, soit déterministe.

Certains algorithmes courants basés sur les politiques : Deterministic Policy Gradient (DPG), Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO), etc.

— **Actor Critic** :

L'algorithme Actor-Critic (AC) combine les approches basées sur les valeurs et sur les politiques en RL tel que :

$$\text{AC} = \text{Policy-based (Actor)} + \text{Value-based (Critic)}$$

1. **Actor (Acteur)** : Choisit les actions selon une politique paramétrée, mise à jour en fonction des retours de l'environnement.

2. **Critic (Critique)** : Évalue les actions de l'acteur en calculant une valeur qui mesure la qualité de ces actions. Il met à jour une fonction de valeur pour aider l'acteur à améliorer sa politique.

Certains algorithmes courants basés sur Actor Critic : Deep Deterministic Policy Gradient (DDPG), Advantage Actor-Critic (A2C), Asynchronous Advantage Actor-Critic (A3C), etc.

2.4.3.2 Model-Based RL

Model-Based RL (MBRL) est une méthode d'apprentissage par renforcement dans laquelle l'agent vise à comprendre et modéliser la dynamique de l'environnement. Contrairement aux approches MFRL, où l'agent apprend uniquement à travers des interactions directes avec l'environnement.

Le MBRL consiste à **modéliser l'environnement** pour **simuler ses dynamiques** (tester et d'évaluer différentes stratégies de manière plus rapide et sécurisée sans interagir directement avec l'environnement réel). Grâce à cette simulation, l'agent peut **planifier ses actions** en anticipant les résultats, ce qui optimise la prise de décision et réduit les interactions réelles nécessaires.

Le principal avantage des approches MBRL réside dans leur efficacité. Elles nécessitent moins d'interactions avec l'environnement pour établir une politique efficace.

Pour MBRL, il existe deux classifications de méthodes :

— **Apprendre le modèle** : Ces méthodes tentent de construire un modèle de l'environnement en apprenant comment les transitions et les récompenses fonctionnent. Certains algorithmes courants basés sur la méthode d'apprendre le modèle incluent : Model-Based Value Estimation (MBVE), Imagination-Augmented Agents (I2A), etc.

— **Utilisation du modèle** : Cette approche suppose que le modèle de l'environnement est déjà connu ou fourni. L'agent utilise ce modèle pour planifier et optimiser ses actions. Par exemple l'algorithme de Alpha Zero.

2.4.4 Avantages et Inconvénients des Approches Model-Based et Model-Free en Apprentissage par Renforcement

RL peut être abordé de deux manières principales : les algorithmes Model-Based et Model-Free. Chacune de ces approches présente des avantages et des inconvénients distincts en termes de complexité, d'efficacité et d'adaptabilité.

Critère	Model-Based	Model-Free
Avantages	- **Efficacité** : Nécessite moins d'interactions avec l'environnement. - **Planification** : Permet d'anticiper à l'avance les résultats des actions. - **Flexibilité** : Peut simuler différents scénarios sans interactions physiques.	- **Simplicité** : Simple à implémenter car ils n'exigent pas la modélisation explicite de l'environnement. - **Rapidité d'apprentissage** : Apprend directement à partir des expériences. - **Applications variées** : Dynamique de l'environnement est trop complexe ou inconnue.
Inconvénients	- **Complexité** : Nécessite la construction et la maintenance d'un modèle précis. - **Sensibilité aux erreurs du modèle** : Un mauvais modèle peut conduire à de mauvaises décisions. - **Temps de calcul** : Peut être plus coûteux en termes de temps de calcul.	- **Inefficacité** : Nécessite plus d'interactions avec l'environnement. - Ne peut pas anticiper les résultats futurs sans interaction directe. - Moins efficace pour des environnements très complexes ou dynamiques.

TABLE 2.2 – Avantages et Inconvénients des algorithmes Model-Based et Model-Free.

2.4.5 Domaines d'applications de RL

RL est en effet très polyvalent et a des applications dans divers domaines. La Figure 2.12 montre les principaux domaines d'application de RL.

- **Robotique** : RL aide les robots à apprendre à effectuer des tâches complexes, comme manipuler des objets ou naviguer dans des environnements inconnus, en optimisant leurs actions à travers des essais et des erreurs.
- **Jeux vidéo** : Dans les jeux, RL est utilisé pour créer des intelligences artificielles qui apprennent à jouer de manière stratégique, en améliorant leur performance contre des adversaires humains ou d'autres IA.
- **Transport** : Les systèmes de conduite autonome utilisent RL pour apprendre à naviguer dans des environnements routiers complexes et à prendre des décisions en temps réel pour la sécurité et l'efficacité.
- **Finance** : RL peut optimiser les stratégies de trading en apprenant à maximiser les rendements financiers tout en minimisant les risques, en s'adaptant aux fluctuations du marché.
- **Autres domaines** : RL est également appliqué dans la gestion de l'énergie, la médecine pour personnaliser les traitements, et les systèmes de recommandation, entre autres.

FIGURE 2.12 – Applications du RL.

2.5 Conclusion

Dans ce chapitre, nous avons exploré les fondements de l'IA et du ML, en examinant leurs définitions, évolutions et différentes branches. Nous avons également abordé les types d'apprentissage en ML, notamment supervisé, non supervisé et semi-supervisé. Enfin, l'apprentissage par renforcement a été présenté en détail, mettant en évidence ses principes, ses approches et ses nombreuses applications.
Dans le chapitre suivant, nous entamerons la partie de simulation et performance des algorithmes RL dans les Systèmes MEC.

Chapitre 3

Simulation et Performance des Algorithmes RL dans les Systèmes MEC

3.1 Introduction

Ce chapitre examine la modélisation et la simulation des systèmes de calcul à travers la comparaison des algorithmes (RL) dans les systèmes MEC afin d'analyser ces performances. Nous commençons par la formulation des problèmes liés au traitement local et au traitement parallèle dans les serveurs MEC. Ensuite, nous discutons d'une méthode décentralisée de offloading, en présentant l'espace d'état, l'espace d'action et la fonction de récompense. Enfin, nous exposons les résultats et la discussion des simulations comparant l'approche basée sur DDPG avec DQN dans deux scénarios : un seul utilisateur et plusieurs utilisateurs.

3.2 Modèle de système et formulation du problème

Le système étudié illustré dans la Figure 3.2 , est un système MIMO (Multiple Input Multiple Output) multi-utilisateurs avec une station de base (BS) à N antennes, un serveur MEC et plusieurs utilisateurs mobiles à antenne unique M = { 1,2,...,M}.
Chaque utilisateur a des tâches de calcul intensives, et un serveur MEC est utilisé pour réduire la charge de calcul en permettant le déchargement via des liens sans fil.

Le système est modélisé en temps discret, ce qui signifie que le temps est divisé en intervalles égaux, appelés périodes ou intervalles de temps τ_0. Chaque période est indexée par T = {0,1,...}.

Les conditions de canal et les arrivées de tâches varient à chaque période, nécessitant un équilibre entre exécution locale et déchargement pour optimiser la consommation d'énergie (puissance) et le délai de mémoire tampon.

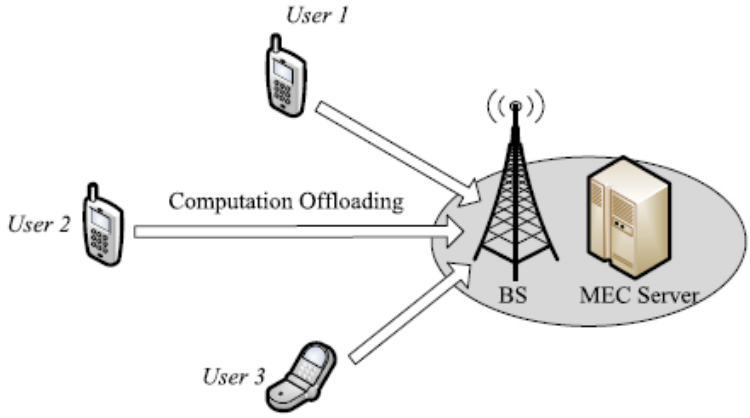

FIGURE 3.1 – Modèle de système.

3.2.1 Modèle de calcul

Nous utilisons a_m(t) pour représenter la quantité de tâches arrivant pour l'utilisateur m en bits pendant l'intervalle de temps t, en supposant que ces arrivées sont indépendantes et identiquement distribuées sur tous les intervalles. À l'arrivée, les bits de tâches sont d'abord stockés dans le tampon de mise en file d'attente de l'utilisateur, puis traités dans les intervalles suivants à partir de t+1. Ainsi, pendant l'intervalle t, une partie des bits de tâches en tampon, notée $d_{l,m}$(t), sera traitée localement sur l'appareil mobile, tandis qu'une autre partie, notée $d_{o,m}$(t), sera déchargée vers le serveur MEC pour être traitée par celui-ci.

Il est important de noter que la quantité totale de bits de tâches exécutés, c'est-à-dire $d_{l,m}$(t) + $d_{o,m}$(t), ne doit pas dépasser la longueur de la file d'attente du tampon de tâches.
Si B_m(t) représente la longueur de la file d'attente du tampon de tâches de l'utilisateur m au début de l'intervalle t, elle évoluera comme suit :

$$B_m(t+1) = [B_m(t) - (d_{l,m}(t) + d_{o,m}(t))] + a_m(t), \quad \forall t \in T$$

avec $B_m(0) = 0$.

3.2.1.1 Traitement local

Pour l'utilisateur m, si $p_{l,m}$(t) $\in [0, P_{l,m}]$ est la puissance allouée à l'exécution locale à l'intervalle t, les bits traités localement peuvent être calculés par :

$$d_{l,m}(t) = \frac{\tau_0 f_m(t)}{L_m}$$

où $P_{l,m}$ est la puissance maximale d'exécution locale, f_m(t) est la fréquence du CPU planifiée avec :

$$f_m(t) = \sqrt[3]{\frac{p_{l,m}(t)}{\kappa}}$$

Notez que κ est la capacité commutée effective dépendant de l'architecture du processeur, et donc $f_m(t) \in [0, F_m]$ étant la fréquence maximale du cycle CPU autorisée pour l'appareil de l'utilisateur m. De plus, L_m désigne le nombre de cycles CPU nécessaires pour traiter un bit de tâche.

3.2.1.2 Traitement parallèle des tâches dans les serveurs MEC

Tous les bits de données de tâches déchargés vers le serveur MEC via la station de base (BS) seront traités. Par conséquent, étant donné la puissance de transmission montante $p_{o,m}(t)$ la quantité de données déchargées de l'utilisateur m pendant l'intervalle de temps t peut être calculée par :

$$d_{o,m}(t) = \tau_0 W \log_2 \left(1 + \gamma_m(t)\right)$$

où W est la largeur de bande du système et $\gamma_m(t)$ est le rapport signal sur bruit.

3.2.1.3 Coût de calcul

Le coût de calcul global pour chaque utilisateur est une somme pondérée de la consommation d'énergie et du délai de mise en file d'attente des tâches. Selon le théorème de Little, la longueur moyenne de la file d'attente est proportionnelle au délai. Ainsi, le coût moyen de calcul à long terme pour chaque utilisateur m est défini comme suit :

$$C_m = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \left(w_{m,1} \left[p_{l,m}(t) + p_{o,m}(t)\right] + w_{m,2} B_m(t)\right),$$

où $w_{m,1}$ et $w_{m,2}$ sont des facteurs de pondération non négatifs.

Au lieu de compter sur une station de base (BS) qui doit observer et contrôler tout le système, chaque utilisateur m prend des décisions de manière autonome (décentralisées) en utilisant ses propres données locales. Cela permet une gestion plus flexible et réactive des ressources, réduisant le coût de calcul et les exigences de communication pour le système global.
Pour cela, les utilisateurs adaptent la puissance de traitement pour l'exécution locale et le déchargement de calcul à chaque intervalle t comme suit :

$$\min_{p_{l,m}(t),\, p_{o,m}(t)} C_m$$

3.3 Méthode décentralisée de déchargement dynamique des calculs

Pour réduire le coût de calcul pour chaque utilisateur, une méthode décentralisée utilise l'algorithme DDPG est proposée pour former des politiques décentralisées qui contrôlent l'allocation de puissance pour l'exécution locale et le déchargement. Ces politiques se basent uniquement sur les observations locales et prennent des décisions dans un espace continu.

Bien que le DQN puisse résoudre des problèmes dans des espaces d'état de haute dimension, il ne prend en charge que les espaces d'action discrets et peu dimensionnés. Le DDPG a été développé pour gérer les espaces d'action continus.Il utilise une approche acteur-critique avec deux réseaux de neurones profonds (DNN) : le critique $Q(s, a \mid \theta_Q)$, similaire au DQN, et l'acteur $\mu(s \mid \theta_\mu)$, qui mappe l'état s à une action continue spécifique. Les détails sur l'espace d'état, l'espace d'action et la fonction de récompense seront présentés par la suite.

3.3.1 Espace d'état

L'espace d'état représente l'environnement du point de vue de chaque utilisateur m.
Il se compose de trois éléments : la longueur de la file d'attente des données $B_m(t)$, le SINR (rapport signal sur interférence et bruit) normalisé $\gamma_m(t-1)$ estimé à partir du SINR précédent, et le vecteur de canal $h_m(t)$. Donc, pour chaque utilisateur, l'espace d'état de l'intervalle t est défini comme :

$$s_{m,t} = [B_m(t), \phi_m(t-1), h_m(t)]$$

Ces informations locales permettent à chaque utilisateur m de prendre des décisions d'allocation de puissance pour l'exécution locale et le déchargement.

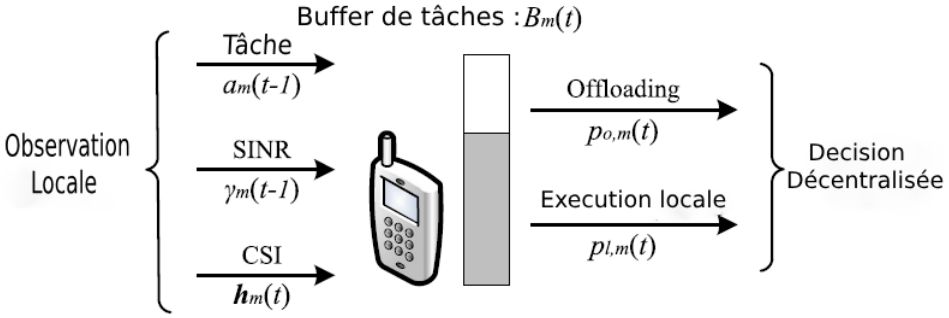

FIGURE 3.2 – Concept d'observation locale et de décision décentralisée.

3.3.2 Espace d'action

une action $a_{m,t}$ comprenant les puissances allouées pour l'exécution locale et le déchargement de calcul sera sélectionnée pour chaque intervalle $t \in T$ comme suit :

$$a_{m,t} = [p_{l,m}(t), p_{o,m}(t)]$$

3.3.3 Fonction de récompense

La fonction de récompense joue un rôle clé dans la performance.
Afin d'apprendre une politique de déchargement de calcul dynamique et économe en énergie, qui minimise le coût de calcul à long terme, la fonction de récompense $r_{m,t}$ reçue par chaque utilisateur après le pas de temps t peut être définie comme suit :

$$r_{m,t} = -w_{m,1} \cdot (p_{l,m}(t) + p_{o,m}(t)) - w_{m,2} \cdot B_m(t)$$

qui est la somme négative pondérée de la consommation totale de puissance instantanée et de la longueur de la file d'attente du tampon de tâches, avec $w_{m,2} = 1 - w_{m,1}$.
w_m : Facteur de pondération utilisé pour ajuster l'importance relative de deux objectifs concurrents : la consommation d'énergie et le délai de mise en mémoire tampon [0,1].

3.4 Entraînements et tests

Pour apprendre et évaluer les politiques de transfert de calcul décentralisé, un environnement simulé avec plusieurs agents est utilisé. Les données sont générées à partir des interactions entre les agents et l'environnement. Il est également à noter que cette interaction est généralement une tâche d'apprentissage par renforcement continu.

- Afin d'obtenir une meilleure performance en exploration, l'interaction sera commencée manuellement avec un état initial aléatoire $s_{m,1}$ pour chaque utilisateur m et se terminera à un nombre maximal de pas préalablement défini T_{Max} pour chaque épisode.
- À chaque étape temporelle t durant un épisode, le tuple d'expérience de chaque agent ($s_{m,t}$, $a_{m,t}$, $r_{m,t}$, $s_{m,t+1}$) sera stocké dans son propre tampon d'expérience B_m.
- Pendant ce temps, les réseaux d'acteur et de critique de l'agent utilisateur seront mis à jour.

— Après l'entraînement de K_{Max} épisodes, chaque agent utilisateur apprendra progressivement et indépendamment sa politique dynamique de transfert de calcul.
— Lors de l'étape de test, chaque agent teste son réseau d'acteur avec un environnement aléatoire, en sélectionnant des actions basées sur ses observations locales.

3.4.1 Outils et environnements utilisés

Pour réaliser les simulations et analyser les performances des différentes stratégies de offloadning computationnel, plusieurs outils et environnements ont été mobilisés. Ces outils ont permis de tester et de comparer les algorithmes de renforcement utilisés dans cette étude.

— **Spécifications matérielles de la machine**
Voici les spécifications matérielles de la machine utilisée pour exécuter ces simulations :

Composant	**Spécifications**
Processeur (CPU)	Intel(R) Core(TM) i3-4000M 2.40 GHz, 8 cœurs, 16 threads
Mémoire vive (RAM)	16 Go
Système d'exploitation	Ubuntu 22.04

TABLE 3.1 – Spécifications matérielles.

— **Environnement de développement : Anaconda**
Anaconda est une distribution libre et open source des langages de programmation Python appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement.[1]

FIGURE 3.3 – Environnement Anaconda.

— **Langage de programmation : Python 3.6.13**
L'implémentation des algorithmes et le contrôle des simulations ont été effectués à l'aide de Python 3.6.13. Python est largement utilisé dans le domaine de la recherche en intelligence artificielle.

— **Frameworks de Machine Learning : TensorFlow 1.15**
Cette version de TensorFlow a été utilisée pour construire et entraîner les modèles de réseaux neuronaux pour les algorithmes DDPG et DQN. TensorFlow est une bibliothèque populaire qui offre des fonctionnalités avancées pour l'optimisation et le calcul automatique des gradients dans les modèles de deep learning.

FIGURE 3.4 – Bibliothèque TensorFlow.

— **Visualisation des résultats : Matplotlib**
Cette bibliothèque a été utilisée pour la visualisation des résultats sous forme de graphiques. Matplotlib permet de créer des visualisations détaillées et personnalisées des résultats des simulations, telles que les courbes de récompense moyenne, la consommation d'énergie, et le délai moyen.

3.5 Résultats et discussion

Dans cette partie, nous montrons des simulations numériques pour illustrer le cadre proposé pour offloading de calcul dynamique décentralisé. Nous comparons les performances de l'approche basé sur DDPG avec DQN dans deux scénarios : un seul utilisateur et plusieurs utilisateurs.

3.5.1 Paramètres de simulation

Voici le tableau 3.2 résume les paramètres de la simulation.

Paramètre	**Valeur**
Tranche de temps (τ_0)	1 ms
Distribution des arrivées de tâches	$a_m(t) \sim \text{Poisson}(\lambda_m)$
Distance de référence (d_0)	1 m
Distance de l'utilisateur (d_m) vers BS	En mètres
Bande passante du système	1 MHz
Puissance de transmission maximale ($P_{o,m}$)	2 W
Puissance maximale pour l'exécution locale ($P_{l,m}$)	2 W
Fréquence maximale des cycles CPU (F_m)	2.4 GHz
Taux d'apprentissage (acteur)	0.0001
Taux d'apprentissage (critique)	0.001
Taille du tampon ($\|B_m\|$)	2.5×10^5
Taille du mini-lot (M)	16
Facteurs pondérés ($w_{m,1}$ et $w_{m,2}$)	$w_{m,1} = 10w_m$, $w_{m,2} = 1 - w_m$
Nombre d'épisodes ($K_{\max}$)	2000
Nombre maximal de pas par épisode ($T_{\max}$)	200

TABLE 3.2 – Paramètres de la simulation.

3.5.2 Stratégies de comparaison

Pour faire la comparaison, certaines stratégies sont présentées :

— **Exécution Locale Prioritaire (Local)** : L'utilisateur exécute tous les bits de données de tâche localement.

— **Offloading de Calcul Prioritaire (Offload)** : L'utilisateur décharge tous les bits de données vers le serveur MEC.

— **Offloading Dynamique Basé sur l'approche de DQN (DQN)** : L'algorithme DQN classique basé sur un espace d'actions discret est adopté. Afin d'assurer une comparaison équitable entre DDPG et DQN, nous utilisons la même configuration de réseau neuronal que dans DDPG.

3.5.3 Scénarios de comparaison

Nous avons basé notre travail sur deux scénarios :
— **Un seul utilisateur**.
— **Plusieurs utilisateurs**.

3.5.3.1 Scenario 1 : Un seul utilisateur

L'utilisateur est positionné aléatoirement à une distance $d_1 = 100$ m de la BS.

1. **Entraînement**

   Dans cette partie, on va voir le processus d'entraînement pour Offloading des calculs avec un seul utilisateur. Nous avons utilisé la valeur de facteur de pondération $W_1 = 0.8$.
   La figure 3.5 représente la comparaison entre deux scénarios (DDPG et DQN) avec des taux d'arrivée des tâches différents ($\lambda_1 = 2.0$ Mbps et 3.0 Mbps).

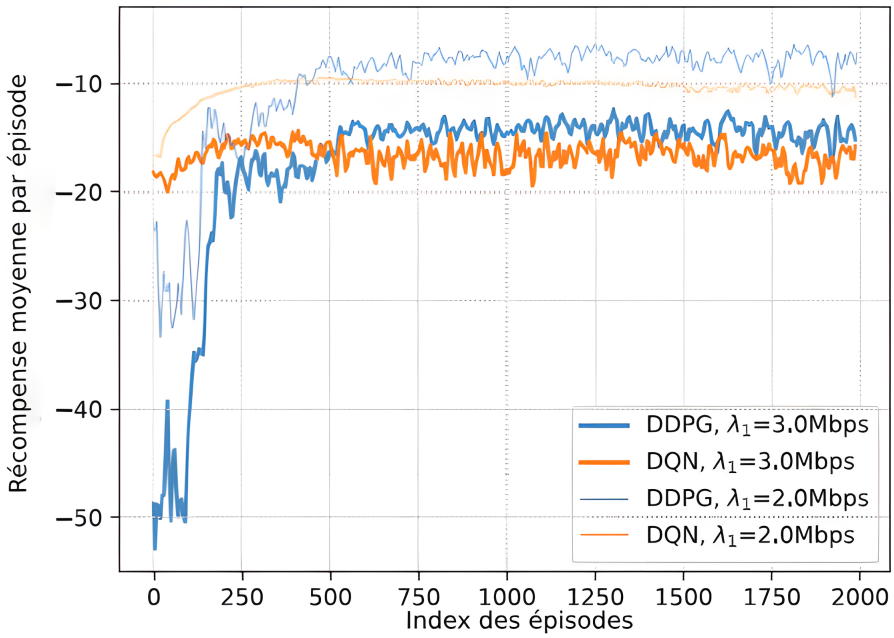

FIGURE 3.5 – Récompense moyenne par épisode pour un seul utilisateur.

On observe que pour chaque configuration, la récompense moyenne augmente au fur et à mesure que les épisodes se déroulent (environ 500 épisodes) , ce qui indique que l'agent apprend à améliorer sa politique de déchargement de calcul au fil du temps.

Les politiques apprises par DDPG semblent donner de meilleures performances que celles apprises par DQN. En particulier, les courbes DDPG (bleues) atteignent des niveaux de récompense plus élevés (-15) que les courbes DQN (oranges) avec des niveaux de récompense (-18). Donc le DDPG est plus efficace que DQN dans les environnements avec des actions continues.

Un taux d'arrivée plus bas ($\lambda_1 = 2.0$ Mbps) semble offrir de meilleures récompenses moyennes pour les deux algorithmes, DDPG (-5) et DQN (-10). On peut conclure que, un taux d'arrivée des tâches plus faible donne des meilleures performances en réduisant la complexité et la variabilité des tâches à traiter.

2. **Test**

Après un entraînement de $k_{Max} = 2000$ épisodes, on va présenter les résultats de tests pour un seul utilisateur avec un facteur de pondération $W_1 = 0.8$, où les performances de différentes stratégies (Local, Offload, DDPG et DQN) sont comparées en fonction du taux d'arrivée des tâches (Mbps). Deux mesures sont affichées : la puissance moyenne (Watt) et le délai de mise en tampon moyen (ms).

La Figure 3.6 montre la consommation d'énergie (puissance) des différentes stratégies en fonction de la charge de travail.

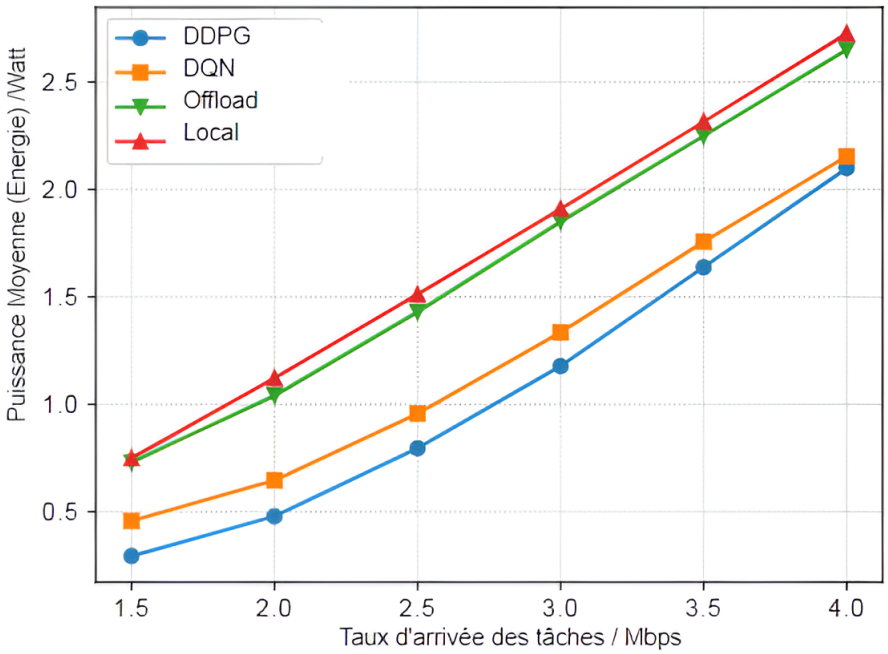

FIGURE 3.6 – Comparaison de puissance moyenne pour différentes stratégies.

On remarque que la puissance consommée augmente pour toutes les stratégies à mesure que le taux d'arrivée des tâches augmente.

La stratégie DDPG (ligne Bleue) affiche la consommation d'énergie la plus faible parmi toutes les stratégies. Cela montre que DDPG optimise mieux l'utilisation des ressources énergétiques, même lorsque la charge de travail augmente, ce qui en fait une stratégie efficace pour les espaces d'actions continus.

La stratégie DQN (ligne Orange) a une consommation d'énergie modérée mais elle est élevée par rapport au DDPG car DQN a moins de flexibilité dans l'optimisation dans les espaces d'actions continus.

La stratégie de offloading en priorité (ligne Verte) présente une consommation d'énergie plus élevée que DQN. Cela signifie qu'elle priorise l'envoi de toutes les tâches vers le serveurs MEC, ce qui entraîne des coûts énergétiques supplémentaires liés à la transmission de données.

La stratégie d'exécution locale (ligne Rouge) a une consommation énergétique la plus élevée parmi toutes les stratégies (presque similaire avec offload). Cela s'explique par le traitement local des tâches qui nécessite une puissance considérable pour gérer une grande quantité de données.

La Figure 3.7 montre la délai moyen dans la mémoire tampon en fonction du taux d'arrivée des tâches pour les différentes stratégies.

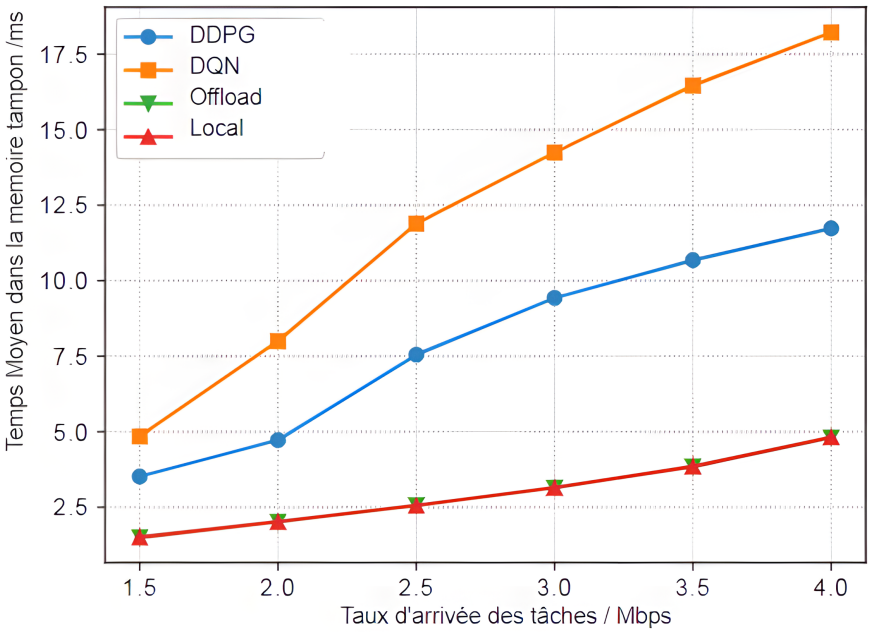

FIGURE 3.7 – Comparaison de délai moyen dans la mémoire tampon pour différentes stratégies.

On remarque que le délai moyen dans la mémoire tampon augmente pour toutes les stratégies à mesure que le taux d'arrivée des tâches augmente (4 Mbps).

Les deux stratégies, Local et Offload, présentent un délai de mise en tampon moyen presque équivalent et relativement faible (5 ms) par rapport aux autres stratégies, car elles traitent les tâches immédiatement ou de les décharger vers le MEC, minimisant les retards mais augmentant la consommation d'énergie, comme la montre la Figure 3.6 de puissance.

La stratégie DDPG a un délai moyen dans le tampon plus élevé (12 ms) par rapport aux deux stratégies précédentes, car elle accepte un léger sacrifice en termes de délai moyen en tampon afin de minimiser la consommation d'énergie.

La stratégie DQN a un délai moyen le plus élevée parmi toutes les stratégies (18 ms) car DQN offre une flexibilité dans les espaces d'actions discrets, mais pas dans les espaces continus.

3. Analyse du Compromis Énergie-Délai

Le facteur de pondération W_1 a un impact sur ces résultats, plus W_1 est élevé, plus la consommation d'énergie diminue, mais le délai de mise en tampon augmente. Cela signifie que W_1 peut être ajusté pour trouver un équilibre entre une faible consommation d'énergie et un délai acceptable.

De plus, pour chaque valeur de W_1, la stratégie basée sur DDPG offre toujours de meilleures performances en termes de consommation d'énergie et de délai de mise en tampon.

La Figure 3.8 décrit le compromis Énergie-délai pour un seul utilisateur avec un taux d'arrivée de tâches de $\lambda_1 = 3{,}0$ Mbps. Dans ce scénario, le facteur W_1 varie de 0,3 à 0,8, avec des points sur chaque courbe qui se déplacent de gauche à droite au fur et à mesure.

Deux stratégies : offload et local, n'ont pas de courbes, ce qui signifie qu'ils ne sont pas influencés par les facteurs de pondération.

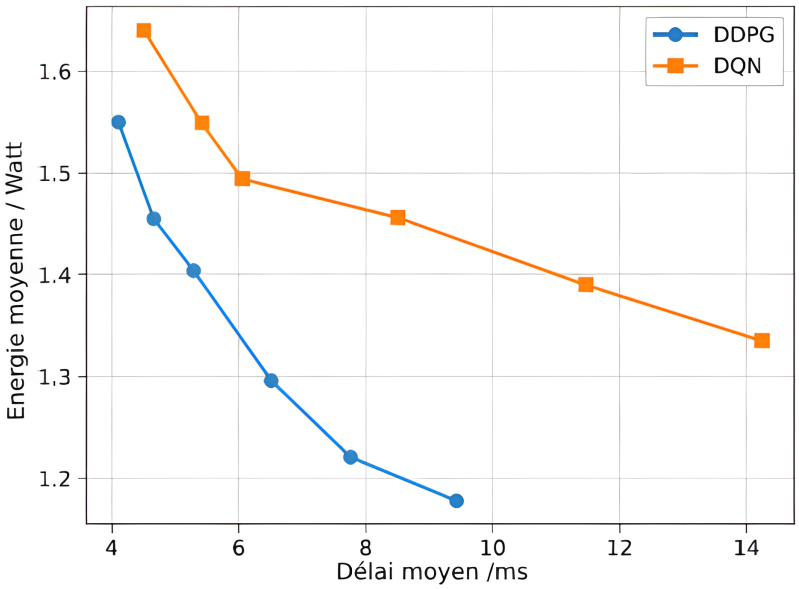

FIGURE 3.8 – Analyse du Compromis Énergie-Délai.

3.5.3.2 Scénario 2 : Plusieurs utilisateurs

On propose un scénario avec 3 utilisateurs mobiles dans le système ($M = 3$), chacun positionné aléatoirement à une distance de $d_m = 100$m de la BS.
Le taux d'arrivée des tâches : $\lambda_m = m * 1.0$ Mbps où m peut être 1, 2 ou 3.
Après $k_{Max} = 2000$ épisodes, les résultats de simulation de différentes politiques de offloading de calcul sont résumés dans le tableau 3.9.

	Récompense moyenne			Energie moyenne (Watt)			Délai moyen (ms)		
	m1	**m2**	**m3**	**m1**	**m2**	**m3**	**m1**	**m2**	**m3**
DDPG	-1,919	-6,366	-16,164	0,162	0,602	1,674	3,114	7,752	13,861
DQN	-2,781	-8,915	-18,675	0,284	0,915	1,954	2,539	7,973	15,216
Offload	-3,417	-10,893	-26,334	0,402	1,309	3,143	1,007	2,103	5,951
Local	-1,949	-13,871	-28,471	0,216	1,678	3,407	1,106	2,228	6,072

FIGURE 3.9 – Comparaison des résultats de simulation pour les trois utilisateurs mobiles.

Ce tableau présente une comparaison des résultats des simulation avec un facteur W_1=0.8 pour trois utilisateurs mobiles, selon plusieurs stratégies de offloading. Pour les trois utilisateurs, DDPG obtient des récompenses meilleures que DQN et les autres stratégies. Cela signifie que les algorithmes basés sur DDPG maximisent les récompenses et donc fournissent de meilleures solutions globales.

De plus, on observe que les stratégies basées sur DDPG et DQN peuvent atteindre une consommation d'énergie beaucoup plus faible avec un léger compromis sur le délai en mémoire tampon.

Les stratégies Offload et Local ont des délais plus courts, avec Offload étant la meilleure.

En résumé, DDPG est la meilleure stratégie pour un compromis efficace entre économie d'énergie et délai, tandis que les stratégies Offload sont plus adaptées aux scénarios où la réduction du délai est prioritaire, mais au prix d'une moindre efficacité énergétique.

3.6 Conclusion

Ce chapitre a examiné la simulation des systèmes de calcul dans les serveurs MEC, en mettant l'accent sur le traitement local et parallèle. Nous avons présenté une méthode décentralisée d'offloading et analysé les performances des algorithmes de RL par une comparaison entre DDPG et DQN dans des scénarios avec un ou plusieurs utilisateurs. Les résultats révèlent les avantages et limites de chaque approche, soulignant l'importance de choisir l'algorithme en fonction des spécificités des systèmes.

Conclusion Générale

Avec le développement rapide des technologies mobiles et des applications gourmandes en calcul, le Mobile Edge Computing (MEC) s'impose comme une solution innovante pour surmonter les limites des systèmes traditionnels. Cette recherche a exploré les défis de l'offloading dans les environnements MEC et a proposé des approches optimisées basées sur l'apprentissage par renforcement.

Dans ce travail, nous avons présenté un système activé par MEC avec des arrivées de tâches stochastiques dans le but de minimiser le coût moyen en termes de consommation d'énergie et de délai dans tampon, en adoptant l'algorithme DDPG basé sur un espace d'actions continues, une politique de offloading efficace a été apprise avec succès pour chaque utilisateur.
Premièrement, nous avons exposé les concepts fondamentaux du Cloud et de l'Edge Computing, ainsi que leurs applications. Nous avons mis en évidence le rôle clé du MEC dans l'amélioration des performances réseau.
Ensuite, nous avons concentré sur l'IA le ML, en mettant l'accent sur l'apprentissage par renforcement. Cette technique s'est montrée particulièrement efficace pour résoudre des problèmes complexes d'optimisation dynamique dans les réseaux MEC.
Finalement, nous avons simulé et analysé la performance des algorithmes de RL. Des simulations numériques ont été réalisées pour vérifier la supériorité de la stratégie basée sur le DDPG par rapport à d'autres schémas de référence.

A la fin de ce travail, nous avons conscience qu'il y a toujours des perspectives futures qui doivent être adaptées sur ce thème en raison que c'est un sujet en tendance et très utilisé dans les grands domaines.
Ces perspectives sont :

- La publication de notre revue dans un avenir proche.
- L'intégration l'algorithme DDPG à plus grande échelle, prenant en compte des réseaux hétérogènes et des infrastructures multi-utilisateurs.
- L'étude des stratégies décentralisées de Offloading binaire pour des tâches indivisibles.

— La collaboration entre les utilisateurs mobiles ou bien les serveurs edges pourrait également être intégrée dans le cadre du DDPG afin d'améliorer les performances de système.

Bibliographie

[1] Envirenement anaconda. https ://www.pythoniste.fr/anaconda/les-environnements

[2] Auday Al-Dulaimy, Yogesh Sharma, Michel Gokan Khan, and Javid Taheri. *Introduction to edge computing*. 09 2020.

[3] Ahmed M Alwakeel. An overview of fog computing and edge computing security and privacy issues. *Sensors*, 21(24) :8226, 2021.

[4] Rosy Aoun, Elias A Doumith, and Maurice Gagnaire. Resource provisioning for enriched services in cloud environment. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 296–303. IEEE, 2010.

[5] C.A. Azencott. *Introduction au Machine Learning*. Info sup. Dunod, 2019.

[6] Azure. Que sont les clouds publics, privés et hybrides ? ttps ://azure.microsoft.com/fr-fr/resources/cloud-computing-dictionary/what-are-

[7] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[8] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. An overview on edge computing research. *IEEE access*, 8 :85714–85728, 2020.

[9] Min Chen and Yixue Hao. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications*, 36(3) :587–597, 2018.

[10] Alok Choudhury, Manojit Ghose, Akhirul Islam, et al. Machine learning-based computation offloading in multi-access edge computing : A survey. *Journal of Systems Architecture*, page 103090, 2024.

[11] Google Cloud. Le cloud computing. https ://cloud.google.com/learn/what-is-cloud-computing ?hl=frsection-5 .

[12] Bastien Confais. *Conception d'un système de partage de données adapté à un environnement de Fog Computing*. PhD thesis, 07 2018.

[13] Sumit Das, Aritra Dey, Akash Pal, and Nabamita Roy. Applications of artificial intelligence in machine learning : review and prospect. *International Journal of Computer Applications*, 115(9), 2015.

[14] Thomas Degris, Olivier Sigaud, and Pierre-Henri Wuillemin. Apprentissage par renforcement exploitant la structure additive des mdp factorisés. In *JFPDA 2007-2e Journées Francophones Planification, Décision, Apprentissage pour la conduite de système*, pages 49–60. Cépaduès, 2007.

[15] Chuan Feng, Pengchao Han, Xu Zhang, Bowen Yang, Yejun Liu, and Lei Guo. Computation offloading in mobile edge computing networks : A survey. *Journal of Network and Computer Applications*, 202 :103366, 2022.

[16] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. Mobile cloud computing : A survey. *Future generation computer systems*, 29(1) :84–106, 2013.

[17] Pawel Garbacki and Vijay K Naik. Efficient resource virtualization and sharing strategies for heterogeneous grid environments. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 40–49. IEEE, 2007.

[18] Kiryong Ha and Mahadev Satyanarayanan. Openstack++ for cloudlet deployment. *School of Computer Science Carnegie Mellon University Pittsburgh*, (2014), 2015.

[19] Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, and Tie Qiu. Survey on fog computing : architecture, key technologies, applications and open issues. *Journal of network and computer applications*, 98 :27–42, 2017.

[20] Yaser Jararweh, Ahmad Doulat, Omar AlQudah, Ejaz Ahmed, Mahmoud Al-Ayyoub, and Elhadj Benkhelifa. The future of mobile cloud computing : integrating cloudlets and mobile edge computing. In *2016 23rd International conference on telecommunications (ICT)*, pages 1–5. IEEE, 2016.

[21] TVE Kämäräinen. *Design, Implementation and Evaluation of a Distributed Mobile Cloud Gaming System*. PhD thesis, Masters Thesis, 2014.

[22] Samee U Khan. The curious case of distributed systems and continuous computing. *IT Professional*, 18(2) :4–7, 2016.

[23] Ezhilmathi Krishnasamy, Sebastien Varrette, and Michael Mucciardi. Edge computing : An overview of framework and applications. *PRACE Technical Report*, 2020.

[24] Zhufang Kuang, Linfeng Li, Jie Gao, Lian Zhao, and Anfeng Liu. Partial offloading scheduling and power allocation for mobile edge computing systems. *IEEE Internet of Things Journal*, 6(4) :6774–6785, 2019.

[25] Hai Lin, Sherali Zeadally, Zhihong Chen, Houda Labiod, and Lusheng Wang. A survey on computation offloading modeling for edge computing. *Journal of Network and Computer Applications*, 169 :102781, 2020.

[26] Liqing Liu, Zheng Chang, and Xijuan Guo. Socially aware dynamic computation offloading scheme for fog computing system with energy harvesting devices. *IEEE Internet of Things Journal*, 5(3) :1869–1879, 2018.

[27] Margalith Mamou. Edge computing : Qu'est-ce que c'est ? À quoi ça sert ? https ://datascientest.com/edge-computing-tout-savoir .

[28] Yuyi Mao, Jun Zhang, and Khaled B. Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12) :3590–3605, 2016.

[29] John McCarthy et al. What is artificial intelligence. 2007.

[30] Mahshid Mehrabi, Dongho You, Vincent Latzko, Hani Salah, Martin Reisslein, and Frank HP Fitzek. Device-enhanced mec : Multi-access edge computing (mec) aided by end device computation and caching : A survey. *IEEE Access*, 7 :166079–166108, 2019.

[31] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[32] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[33] Zhengyuan Pang, Lifeng Sun, Zhi Wang, Erfang Tian, and Shiqiang Yang. A survey of cloudlet based mobile computing. In *2015 International conference on cloud computing and big data (CCBD)*, pages 268–275. IEEE, 2015.

[34] Nicolas P Rougier. Une brève histoire de l'intelligence artificielle. In *Pint of Science*, 2015.

[35] Stuart J Russell and Peter Norvig. *Artificial intelligence : a modern approach*. Pearson, 2016.

[36] Shagan Sah. Machine learning : a review of learning types. 2020.

[37] Guillaume Saint-Cirgue. *Apprendre le Machine Learning en une semaine*. 2019.

[38] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4) :14–23, 2009.

[39] Xiaoyu Shan, Hanxiao Zhi, Peng Li, and Zhijie Han. A survey on computation offloading for mobile edge computing information. In *2018 IEEE*

4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), pages 248–251, 2018.

[40] Usman Shaukat, Ejaz Ahmed, Zahid Anwar, and Feng Xia. Cloudlet deployment in local wireless networks : Motivation, architectures, applications, and open challenges. *Journal of Network and Computer Applications*, 62 :18–40, 2016.

[41] Yan Shi, Shanzhi Chen, and Xiang Xu. Maga : A mobility-aware computation offloading decision for distributed mobile cloud computing. *IEEE Internet of Things Journal*, 5(1) :164–174, 2018.

[42] Ivan Stojmenovic and Sheng Wen. The fog computing paradigm : Scenarios and security issues. In *2014 federated conference on computer science and information systems*, pages 1–8. IEEE, 2014.

[43] Shuo Wang, Xing Zhang, Zhi Yan, and Wang Wenbo. Cooperative edge computing with sleep control under nonuniform traffic in mobile edge networks. *IEEE Internet of Things Journal*, 6(3) :4295–4306, 2019.

[44] Jinming Yang, Awais Aziz Shah, and Dimitrios Pezaros. A survey of energy optimization approaches for computational task offloading and resource allocation in mec networks. *Electronics*, 12(17) :3548, 2023.

[45] Changsheng You, Kaibin Huang, Hyukjin Chae, and Byoung-Hoon Kim. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, 16(3) :1397–1411, 2017.

[46] Ben Zhang, Nitesh Mor, John Kolb, Douglas S Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. The cloud is not enough : Saving {IoT} from the cloud. In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, 2015.

[47] Yan Zhang. *Mobile edge computing*. Springer Nature, 2022.

[48] Zhicai Zhang, F. Richard Yu, Fang Fu, Qiao Yan, and Zhouyang Wang. Joint offloading and resource allocation in mobile edge computing systems : An actor-critic approach. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.