UNIVERSITÉ DU QUÉBEC À MONTRÉAL

OPTIMISATION CONJOINTE DU DÉCHARGEMENT DE TÂCHES ET D'ORDONNANCEMENT DE TRANSMISSIONS EN MULTIDIFFUSION DANS LES RÉSEAUX MEC

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

ISMAIL BAGAYOKO

MARS 2024

REMERCIEMENTS

Je tiens à exprimer ma profonde gratitude envers toutes les personnes qui ont joué un rôle essentiel dans la réalisation de ce mémoire.

Je commence par exprimer mes remerciements sincères à mon directeur de mémoire, Elmahdi Driouch, pour son encadrement attentif, ses conseils perspicaces et son soutien constant tout au long de ce projet.

Je souhaite également exprimer ma reconnaissance envers Karim Bagayoko, mon père, pour son soutien inconditionnel. Sa présence et ses encouragements, surtout dans les moments difficiles, ont été une source inestimable de motivation. Merci d'avoir pris le temps de me soutenir et d'être présent à mes côtés lors de cette étape importante.

Je tiens à exprimer ma profonde reconnaissance envers ma famille pour son soutien inconditionnel, son encouragement constant, ainsi que sa compréhension et sa patience tout au long de cette période d'études.

Un grand merci à mes collègues pour leur collaboration et leur soutien précieux tout au long de ce projet. Leur contribution a enrichi mon expérience et a rendu cette aventure encore plus significative.

Enfin, ma gratitude s'adresse à toutes les personnes qui, de près ou de loin, ont participé à la réalisation de ce mémoire.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

3GPP 3rd Generation Partnership Project.

AG Algorithme génétique.

AGOR Algorithme génétique pour ordonnancement des requêtes.

AHG Algorithme heuristique glouton.

AR Réalité augmentée (en anglais augmented reality).

ABC Colonie d'abeilles artificielles (en anglais artificial bee colony).

AVG Groupe moyen (en anglais average group).

BILP Programme linéaire binaire (en anglais binary integer linear program).

BS Station de base (en anglais base station).

CPU Processeur central (en anglais central processing unit).

CR1 Croisement à un point.

CR2 Croisement à deux points.

CRU Croisement uniforme.

CEGA Cost Effective Genetic Algorithm for Tasks Offloading.

DRL Apprentissage profond par renforcement (en anglais deep reinforcement learning).

DMRO Apprentissage profond par renforcement méta.

DRC Contrôle de débit des données (en anglais Data Rate Control).

EC Informatique en périphérie (en anglais Edge Computing).

HD échéance stricte (en anglais hard deadline, HD).

IoT Internet des objets (en anglais internet of things).

IP Protocole internet (en anglais Internet Protocol).

LCG Gain de canal minimum (en anglais least channel gain).

MEC Multi-accès informatique en périphérie (en anglais multi-access edge computing).

MD Utilisateur mobile (en anglais mobile device).

MCS schéma de codage de modulation (en anglais modulation coding scheme).

OSS système de support opérationnel(en anglais operations support system).

PW-DDPG Wolpertinger Deep Deterministic Policy Gradient.

PD Échéance proportionnelle (en anglais proportional deadline).

QoE Qualité de l'expérience (en anglais quality of experience).

QoS Qualité de service (en anglais quality of service).

SE Sélection des élites.

SR Sélection par roulette.

SNR Rapport signal sur bruit (en anglais signal-to-noise-ratio).

UE Équipement utilisateur, anglais user equipment.

VR réalité virtuelle (en anglais virtual reality).

RÉSUMÉ

L'adoption de l'informatique multi-accès en périphérie découle de la nécessité de surmonter les défis liés aux ressources limitées des appareils mobiles et d'optimiser les performances des applications gourmandes en calcul comme la réalité virtuelle et augmentée. Par ailleurs, en exploitant la propriété de réutilisation des contenus, la transmission en multidiffusion se positionne comme une approche efficiente pour soulager la charge du réseau, facilitant la diffusion de contenus communs à plusieurs abonnés. Ce mémoire explore les défis et les opportunités liés à l'intégration de la multidiffusion dans les réseaux multi-accès de calcul en périphérie (MEC). Nous avons formulé un problème complexe de prise de décision de déchargement de tâches au niveau du MEC, la formation de groupes d'envoi en multidiffusion, et l'ordonnancement des envois. La formulation mathématique du problème, incluant son analyse de complexité, est exposée. Nous développons une solution heuristique gloutonne ainsi qu'une approche basée sur l'algorithmique génétique. Les résultats obtenus fournissent des aperçus précieux sur les avantages fondamentaux de la multidiffusion, illustrant son potentiel d'optimisation des ressources réseau et de réduction significative de la duplication des données. Ils montrent aussi la supériorité de notre approche génétique, soulignant l'importance d'un équilibre entre la multidiffusion et les envois individuels. Au-delà de ces contributions, des perspectives sont envisagées, notamment la gestion dynamique de la mise en cache et la coordination entre plusieurs serveurs.

Mots clés: calcul périphérique mobile, ordonnancement, allocation des ressources, déchargement de tâches, algorithme génétique, heuristique, multidiffusion.

CHAPITRE 1
INTRODUCTION

1.1 Mise en contexte et motivation

L'informatique mobile est devenue omniprésente dans notre vie quotidienne, et de nombreuses applications exigent des performances élevées pour exécuter des tâches sur des appareils mobiles. Cependant, les ressources limitées de ces appareils, tels que la capacité de stockage et l'autonomie de la batterie, rendent l'exécution de certaines tâches complexes, difficile et parfois impossible. De plus, les problèmes de latence et d'énergie peuvent affecter négativement l'expérience utilisateur.

De nombreux types d'applications mobiles émergentes, tels que la réalité virtuelle (en anglais virtual reality, VR) et augmentée (en anglais augmented reality, AR), la reconnaissance faciale et gestuelle, les jeux interactifs en ligne, etc., gagnent rapidement en popularité sur le marché de l'informatique mobile. Généralement, ces applications sont gourmandes en calcul, sensibles à la latence et nécessitent une consommation énergétique élevée. Cependant, en raison de la durée de vie limitée de la batterie et des capacités de calcul limitées des dispositifs mobiles (en anglais user equipment, UE), il leur est souvent très difficile de satisfaire les exigences de ces applications et garantir une meilleure qualité d'expérience (en anglais quality of experience, QoE) (Sun *et al.*, 2020).

Pour combler le fossé entre les UE aux ressources limitées et les applications gourmandes en calcul et sensibles à la latence, l'informatique multi-accès en périphérie (en anglais multi-access edge computing, MEC) a récemment émergé en tant que technologie prometteuse. Contrairement au système d'infonuagique (en anglais, cloud computing) conventionnel, qui repose souvent sur des centres de données publics distants tels qu'Amazon Web Services et Microsoft Azure, le MEC augmente la capacité de calcul à la périphérie des réseaux mobiles. Il peut en l'occurrence se baser sur le déploiement de serveurs à haute performance densément répartis à proximité des UE ou tout simplement tirer profit des ressources distribuées et abondantes des dispositifs périphériques tels que les stations de base et les points d'accès. Cette approche vise à réduire la latence de trans-

mission, améliorant ainsi la réponse des applications en temps réel (Nath et Wu, 2020).

Récemment, trois approches efficaces, prometteuses et surtout complémentaires ont été envisagées pour résoudre le problème de la limitation de la bande passante sans fil : le déchargement de calcul vers le MEC (en anglais edge task offloading), la mise en cache en périphérie (en anglais edge caching) et la multidiffusion (en anglais multicast) (Sun *et al.*, 2020; Huang *et al.*, 2016).

Le trafic de données moderne présente une forte réutilisation de contenu. La stratégie de mise en cache de contenu se révèle être une approche particulièrement efficace pour atténuer la congestion du réseau pendant les heures de pointe. Cette méthode consiste à précharger des contenus en avance, les rapprochant ainsi des utilisateurs pendant les périodes de moindre activité. En conséquence, elle permet de réduire les besoins en capacité, de la liaison de retour et de la liaison sans fil, ce qui a pour effet de diminuer le délai de transmission. Parallèlement, la transmission en multidiffusion offre une solution efficace pour alléger la charge du réseau lors de la diffusion de contenus communs à plusieurs abonnés sur un même bloc de ressources (Nath et Wu, 2020; Sun *et al.*, 2020; Chandra *et al.*, 2013).

1.2 Problématique

La multidiffusion se démarque par plusieurs avantages majeurs, ce qui en fait une solution efficace pour diverses applications. Trois de ces avantages notables sont les suivants: le premier concerne l'optimisation de la bande passante. La multidiffusion se révèle particulièrement efficace dans l'utilisation de la bande passante, permettant de transmettre un seul paquet à plusieurs destinataires en une seule transmission. Deuxièmement, l'évolutivité intrinsèque de la multidiffusion se manifeste par la constance de la charge sur l'émetteur, indépendamment du nombre de destinataires dans un groupe. Enfin, le troisième avantage réside dans la réduction de la latence par rapport à l'envoi individuel. La multidiffusion offre généralement une latence plus faible, les données étant diffusées simultanément à tous les membres du même groupe. Cependant, malgré ces avantages, l'emploi de la multidiffusion introduit un défi significatif lié à l'ordonnancement des transmissions, en particulier lorsqu'il est couplé à la formation des groupes de multidiffusion.

D'autre part, le déchargement de tâches présente également plusieurs avantages tels que l'amélioration des performances des équipements d'utilisateurs mobiles et la prolongation de la durée de vie de la batterie. Cette pratique permet aux appareils mobiles de transférer des charges de calcul complexes vers des serveurs en périphérie, améliorant ainsi leurs performances et leur réactivité. Le déchargement de tâches contribue à économiser l'énergie des batteries des appareils mobiles en déléguant certaines tâches à des serveurs en périphérie, prolongeant ainsi leur autonomie.

Néanmoins, le défi associé au déchargement de tâches réside dans la prise de décision liée à cette pratique. Face à diverses options de déchargement, notamment l'exécution locale ou au niveau du serveur MEC, il est essentiel d'explorer différentes approches pour déterminer celle qui convient le mieux aux exigences spécifiques du contexte.

Enfin, la nécessité d'une optimisation conjointe de la multidiffusion et du déchargement de tâches émerge comme une direction prometteuse. La question est de savoir comment maximiser les avantages de la multidiffusion tout en optimisant la prise de décision associée au déchargement de tâches. Notre travail se concentre sur la résolution coordonnée des défis liés à la formation des groupes de multidiffusion, à l'ordonnancement des transmissions et à la prise de décision de déchargement dans un réseau MEC.

1.3 Objectifs

L'objectif général de notre recherche réside dans l'amélioration du processus de déchargement de tâches dans les réseaux MEC en se basant sur une combinaison efficace de transmissions en multidiffusion et d'envois individuels. L'atteinte de cet objectif passe par le développement de solutions algorithmiques qui maximisent le nombre de tâches exécutées par l'ensemble des composantes du réseau de périphérie, à savoir les serveurs MEC et les équipements des utilisateurs.

Notre démarche englobe plusieurs objectifs spécifiques. Tout d'abord, nous cherchons à développer des solutions algorithmiques pour la prise de décision de déchargement et l'ordonnancement des groupes de multidiffusion, visant à décider efficacement quelles tâches décharger d'un groupe

spécifique. En outre, nous explorons la possibilité d'optimiser conjointement le processus de déchargement des tâches et l'attribution des ressources de communication. Cette optimisation a pour but de garantir une utilisation efficace des capacités de communication tout en respectant des exigences strictes en termes de délai d'exécution des tâches.

Nous partons du principe qu'une prédiction des requêtes futures est disponible. Notre recherche se focalise essentiellement sur le scénario hors ligne, où la station de base est préalablement informée de l'intégralité des requêtes à traiter. Nous considérons cela comme une base pour évaluer les performances, étant donné que les résultats obtenus dans le cadre hors ligne représentent une borne supérieure pour les performances du problème en ligne équivalent. Ainsi, notre recherche offre une perspective sur les défis et les opportunités liées au problème, tout en explorant des solutions pour améliorer l'efficacité des systèmes MEC.

1.4 Méthodologie

La méthodologie adoptée pour cette recherche repose sur une approche systématique, intégrant une exploration de la littérature, une modélisation mathématique élaborée, et une mise en œuvre des solutions algorithmiques envisagées. Cette démarche méthodologique vise à adresser les défis spécifiques liés à notre problématique, à savoir la formation des groupes de multidiffusion et la prise de décision de déchargement des tâches dans le réseau MEC.

Nous entamons notre démarche en réalisant un état de la littérature récente, se concentrant sur les travaux en lien avec les problématiques de notre étude. Cette première étape permet de positionner notre étude dans le contexte de recherche pertinente, tout en identifiant les tendances, les lacunes, et les opportunités de recherche.

La modélisation mathématique constitue une étape cruciale. En utilisant la théorie de l'optimisation, nous formalisons le modèle du système étudié. Cela englobe la spécification précise des composants du réseau, de l'environnement et des paramètres pertinents. Le problème étudié est formulé en utilisant la théorie d'optimisation sous la forme d'un programme linéaire binaire (en anglais binary integer linear program, BILP) qui maximise le nombre de requêtes exécutées, tout

en respectant les contraintes liées à la disponibilité des ressources, à l'échéance des requêtes et en assurant une qualité de service (en anglais quality of service, QoS) satisfaisante pour les utilisateurs.

Pour résoudre le problème formulé, ce travail propose deux approches qui diffèrent par leur façon de résoudre le compromis entre les performances et la complexité algorithmique. La première proposition repose sur la conception d'un algorithme vorace qui privilégie la réduction du temps d'exécution. La deuxième approche repose sur l'utilisation de la métaheuristique génétique. Cette dernière offre des solutions flexibles et efficaces pour l'optimisation dans des environnements dynamiques et complexes.

L'étape suivante implique l'implémentation des algorithmes développés, réalisée à l'aide du langage de programmation Python. Cette phase pratique nous permet de conduire des simulations et de recueillir des données tangibles, afin d'évaluer les performances de nos solutions. L'analyse comparative approfondie inclut des ajustements de paramètres et d'opérations génétiques pour déterminer la combinaison optimale. Les performances de nos solutions seront comparées à un cas de référence utilisant exclusivement l'envoi individuel.

Cette méthodologie, conjuguant recherche approfondie, modélisation mathématique rigoureuse, algorithmes spécifiques, et évaluation par le biais de simulations, vise à générer des solutions novatrices pour répondre de manière intégrée aux défis posés par notre étude.

1.5 Contributions

Nos contributions principales sont les suivantes :

1. Notre première contribution réside dans la formulation rigoureuse du problème d'ordonnancement au sein d'un environnement intégrant une mise en cache de contenu et la capacité de sa transmission en multidiffusion.
2. Nous avons développé une solution sous la forme d'un algorithme glouton qui permet d'atteindre des performances acceptables avec une complexité algorithmique assez réduite.

3. Parallèlement, nous avons élaboré un algorithme génétique, en introduisant plusieurs variations de la fonction de compatibilité, les opérations de croisement et de sélection. Cette diversification de notre approche nous a permis d'explorer un éventail plus large de solutions potentielles.

4. Enfin, nous avons réalisé une évaluation approfondie des performances de l'algorithme glouton et de l'algorithme génétique. Cette évaluation a englobé l'exploration de différentes configurations et paramètres de l'algorithme génétique.

1.6 Organisation du mémoire

Le reste de ce mémoire est organisé comme suit :

- **Chapitre 2 : Généralités** : Ce chapitre offre une introduction complète aux concepts de base du MEC ainsi qu'au mode de fonctionnement de la multidiffusion.

- **Chapitre 3 : État de l'art** : Dans ce chapitre, nous passons en revue les travaux de recherche existants liés à notre problématique, mettant en évidence les avancées récentes dans ce domaine.

- **Chapitre 4 : Modèle de système et formulation du problème** : Ce chapitre se penche sur le modèle de système que nous avons utilisé, expose la formulation mathématique de notre problème, et inclut une analyse de sa complexité.

- **Chapitre 5 : Solutions proposées** : Dans ce chapitre, nous présentons en détail les deux solutions que nous avons développées pour résoudre notre problème.

- **Chapitre 6 : Résultats des simulations** : Ce chapitre contient une évaluation complète des performances de nos approches, basée sur les résultats de nos simulations.

- **Chapitre 7 : Conclusion** : Enfin, le dernier chapitre résume nos contributions, tire des conclusions importantes, et présente des pistes de recherche futures dans ce domaine.

CHAPITRE 2
GÉNÉRALITÉS

Dans ce chapitre, nous aborderons deux concepts fondamentaux qui jouent un rôle essentiel dans notre étude : la multidiffusion et le MEC. Nous commencerons par explorer les caractéristiques et les avantages de la multidiffusion, puis nous nous pencherons sur le MEC en examinant ses caractéristiques et son rôle dans les réseaux de communication modernes.

2.1 La multidiffusion

La multidiffusion est un moyen d'envoyer les mêmes données à plusieurs destinataires spécifiques simultanément sans que la source ait à générer une copie distincte pour chaque destinataire. Alors que le trafic de diffusion est envoyé à chaque appareil, qu'ils le veuillent ou non, la multidiffusion permet aux destinataires de s'abonner au trafic qu'ils souhaitent recevoir. En conséquence, le taux d'utilisation des ressources est amélioré, puisque le trafic n'est envoyé qu'une seule fois, et les coûts de réseau sont réduits. Contrairement aux réseaux câblés, les systèmes sans fil sont confrontés à des conditions de propagation variables où la qualité de canal peut différer grandement d'un équipement utilisateur (en anglais user equipment, UE) à l'autre. Par conséquent, une station de base (en anglais base station, BS) qui tente de servir un groupe d'UE en multidiffusion se trouve contrainte à utiliser la technique de modulation et de codage (MCS) la moins performante dans le but d'accommoder l'UE avec le pire canal, réduisant ainsi l'efficacité de la communication. Cette problématique a constitué le sujet de nombreux travaux de recherche visant à optimiser la formation des groupes de multidiffusion dans des contextes et environnements variés qui considère le développement technologique du moment (Chukhno *et al.*, 2023).

2.1.1 Avantages

La multidiffusion dans les réseaux sans fil offre de nombreux avantages, tels que l'économie de la bande passante, l'évolutivité, la réduction de la latence, l'efficacité énergétique, l'économie des ressources réseau, l'optimisation du trafic, la simplicité de gestion, et le support d'applications en

temps réel. De plus, elle est largement utilisée pour des applications nécessitant une diffusion en temps réel à plusieurs destinataires, comme les communications vidéo en direct, les mises à jour logicielles, la diffusion de contenu en continu, les conférences en ligne, et bien d'autres. Par conséquent, la multidiffusion constitue un choix judicieux pour de nombreuses applications sans fil.

Nous détaillons dans ce qui suit les avantages de la multidiffusion :

1. **Économie de bande passante :** L'un des avantages les plus importants de la multidiffusion est son efficacité en termes d'utilisation de la bande passante. Contrairement à l'envoi individuel (en anglais unicast), où chaque paquet est envoyé séparément à chaque destinataire, la multidiffusion permet d'envoyer un seul paquet à plusieurs destinataires en une seule transmission. Cet avantage est plus marqué dans le contexte des systèmes de communication sans fil, en raison de la nature de diffusion du média utilisé. Cela réduit considérablement la quantité de bande passante nécessaire, en particulier lorsque de nombreux destinataires partagent le même contenu, ce qui est courant dans des applications telles que la diffusion de vidéos ou de mises à jour logicielles.

2. **Évolutivité :** La multidiffusion est intrinsèquement évolutive. Peu importe le nombre de destinataires, la charge sur l'émetteur, reste la même, ce qui la rend adaptée aux applications de diffusion en direct à large audience, comme les mises à jour de logiciels pour de nombreux utilisateurs.

3. **Latence réduite :** La multidiffusion offre généralement une latence plus faible par rapport à l'envoi individuel, car les données sont diffusées simultanément à tous les membres du groupe. Cela est particulièrement important pour les applications en temps réel, telles que les communications vidéo en direct ou les jeux en ligne, où une latence minimale est essentielle.

4. **Efficacité énergétique :** Dans un contexte où de nombreux appareils sans fil sont alimentés par batterie, la multidiffusion peut contribuer à l'efficacité énergétique. Le fait de recevoir

un seul paquet plutôt que plusieurs économise de l'énergie pour les destinataires, ce qui prolonge la durée de vie des batteries des UE.

5. **Économie de ressources réseau :** En réduisant la duplication de données, la multidiffusion réduit la charge sur l'infrastructure réseau. Cela permet aux réseaux de mieux gérer leurs ressources et de minimiser la congestion.

6. **Simplicité de gestion :** La gestion d'un groupe de multidiffusion est généralement plus simple que de gérer un grand nombre de connexions individuelles. Cela réduit la charge de gestion pour les administrateurs réseau.

7. **Optimisation du trafic :** La multidiffusion réduit la charge du trafic sur le réseau en évitant la transmission redondante des mêmes données à plusieurs destinataires. Cela améliore les performances globales du réseau et garantit une utilisation efficace de la bande passante.

2.1.2 Défis

L'emploi de la multidiffusion dans un réseau sans fil présente plusieurs défis spécifiques (Qadir *et al.*, 2014), (Afolabi *et al.*, 2012):

1. **Gestion des interférences :** Les réseaux sans fil sont sujets aux interférences, en particulier dans des environnements à grande densité et bande passante limitée. Lorsque plusieurs appareils émettent en mode multidiffusion, cela peut entraîner des collisions et des interférences radio. Gérer ces interférences tout en maintenant la qualité du service est essentiel.

2. **Problèmes de mobilité :** Dans un environnement sans fil, les appareils peuvent être en mouvement. Lorsqu'ils font partie d'un groupe de multidiffusion, leur mobilité peut entraîner des problèmes d'association et de synchronisation. Les membres du groupe de multidiffusion peuvent entrer ou quitter la zone de couverture du réseau, ce qui nécessite une gestion intelligente pour éviter les interruptions de la communication.

3. **Qualité de service (QoS) :** La qualité de service est cruciale pour de nombreuses applications. Les conditions du canal sans fil sont variables, ce qui peut affecter la qualité de la

transmission. Assurer une QoS constante malgré ces variations est un défi important qui devient plus difficile à surmonter dans un système à multidiffusion.

4. **Sécurité :** La sécurité est un défi majeur pour les transmissions multidiffusion, en particulier dans les réseaux sans fil. Les membres du groupe peuvent être géographiquement dispersés, ce qui rend la gestion des autorisations et la protection contre les intrusions plus complexes. Il est essentiel de mettre en œuvre des mécanismes de sécurité robustes pour protéger ce type de transmissions.

5. **Gestion des groupes :** La gestion des groupes de multidiffusion peut être complexe, en particulier lorsque les membres du groupe changent fréquemment. Il est essentiel de disposer de mécanismes de gestion efficaces pour ajouter ou supprimer des membres du groupe et garantir que toutes les parties reçoivent les données de manière cohérente.

6. **Bande passante :** Bien que la multidiffusion permette d'économiser de la bande passante par rapport aux envois individuels, l'utilisation intensive de la multidiffusion peut saturer le réseau et entraîner une utilisation inefficace de la bande passante. .

2.1.3 Fonctionnement général et approches de transmission

Dans chaque intervalle de temps, un UE peut avoir un rapport signal/bruit (en anglais signal-to-noise ratio, SNR) différent, ce qui détermine le débit maximal auquel cet utilisateur peut recevoir des données de manière fiable. Pour les communications en envoi individuel, à chaque intervalle de temps, chaque UE envoie un message de contrôle de débit des données (en anglais data rate control, DRC) spécifiant ce débit maximal à la BS. Comme chaque UE peut spécifier un DRC différent, le planificateur de la BS doit décider quel UE servir et à quel débit en se basant sur les retours DRC des UE. Il est important de noter que si la BS envoie à un débit supérieur au débit DRC de l'utilisateur, alors ce dernier ne pourrait pas recevoir correctement les données qui lui sont destiné (Won *et al.*, 2009).

Dans le cas de la multidiffusion, à chaque intervalle de temps, la BS peut transmettre uniquement à un groupe de multidiffusion à un débit donné. Il existe plusieurs groupes de multidiffusion dans

une cellule, et chaque groupe de multidiffusion peut contenir un nombre différent d'utilisateurs situés à des endroits divers dans une cellule. La principale difficulté de la planification de la multidiffusion pour les réseaux sans fil provient de l'hétérogénéité des débits réalisables par les UE au sein du même groupe de multidiffusion. Étant donné que tous les UE d'un groupe de multidiffusion doivent être soumis au même débit de transmission choisi par la BS à chaque intervalle de temps, il est difficile de déterminer le débit auquel la BS transmet à un groupe. Si elle envoie au débit le plus élevé parmi ceux demandés par les utilisateurs, plusieurs utilisateurs risquent de ne pas pouvoir décoder le signal reçu. Cependant, si la BS effectue la transmission au débit le plus bas demandé par les utilisateurs d'un groupe, alors d'autres utilisateurs avec des débits DRC plus élevés (meilleures conditions de canal) seront sévèrement pénalisés. Par conséquent, le travail difficile de la BS pour la multidiffusion consiste, à chaque intervalle, à décider de la meilleure formation du groupe et déterminer débit pour transmettre à ce groupe.

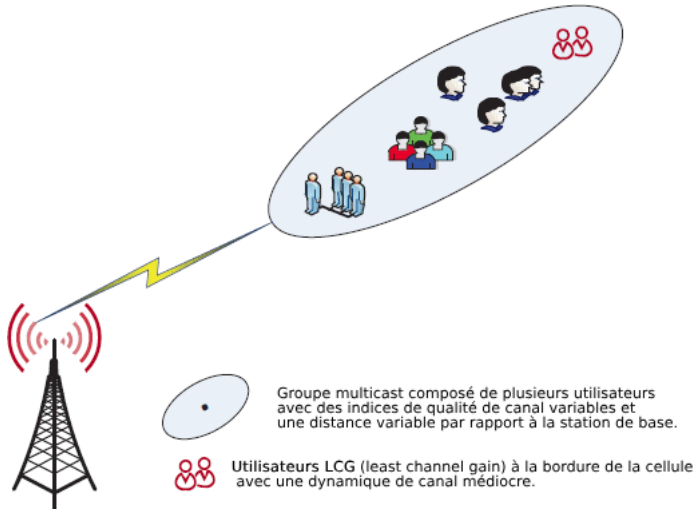

Figure 2.1 Groupe de multidiffusion avec le débit qui peut être un débit fixe, un débit moyen du groupe ou le débit de l'utilisateur le moins performant du groupe.

Source (Afolabi *et al.*, 2012)

Pour un groupe de multidiffusion, comme celui illustré dans la figure 2.1, la littérature (Afolabi *et al.*, 2012) présente trois approches simples et largement adoptées, pour la sélection du débit d'envoi :

1. **Débits fixes prédéfinis :** L'idée est l'utilisation de débits de groupe de transmission préalablement définis, avec tous les envois effectués à un même débit. Cette approche simple est spécialement conçue pour satisfaire favorablement les utilisateurs en bordure de cellule, qui sont censés avoir des canaux à faibles gains due à leur distance plus éloignée de la BS.

2. **Débit de l'UE à gain de canal minimal (en anglais least channel gain, LCG) :** Cette approche définit de manière adaptative les débits de groupe de transmission à celui de l'UE ayant la qualité de canal la plus faible. Ici, tous les UE arrivent à décoder le signal transmis, cependant les UE bénéficiant d'un bon canal se voient sérieusement pénalisés.

3. **Débit moyen du groupe (en anglais average group, AVG) :** Pour améliorer la capacité du système et exploiter la dynamique des canaux, la BS transmet au groupe en fonction du débit moyen à long terme. Il existe plusieurs variations de cette approche, dont une consiste à utiliser un débit permettant de servir la moitié des membres du groupe.

Il est clair que la maximisation de la capacité basée sur le débit moyen du groupe (AVG) permet d'atteindre une capacité supérieure à l'approche LCG. Cependant, l'approche LCG peut être privilégiée dans plusieurs scénarios puisqu'elle offre non seulement des avantages pratiques en matière d'implémentation, mais permet de satisfaire également la QoS de tous les utilisateurs dans le groupe.

2.2 Informatique multi-accès en périphérie (MEC)

Au cours de la dernière décennie, l'infonuagique (en anglais cloud computing) a émergé comme un modèle informatique révolutionnaire, devenu rapidement incontournable à cause de ses nombreux avantages, dont principalement sa très grande flexibilité. Ce modèle se caractérise par la centralisation de la gestion des traitements, du stockage et des fonctionnalités réseau dans les centres de données, auxquels on réfère par le terme « nuage ». Cette approche permet l'exploitation des vastes ressources disponibles dans ce nuage informatique pour fournir une puissance de calcul et un espace de stockage élastique afin de soutenir les dispositifs des utilisateurs finaux, qui sont souvent confrontés à des contraintes de ressources (Mao *et al.*, 2017).

De nos jours, avec la généralisation et le développement croissant de l'internet des objets (en anglais internet of things, IoT) dans la vie quotidienne, le nombre d'appareils connectés à cette infrastructure ne cesse d'augmenter, engendrant ainsi une quantité considérable de données. La capacité du réseau dans le domaine de l'informatique en nuage s'est révélée insuffisante pour répondre aux exigences des systèmes sensibles au délai (Cao *et al.*, 2020) et requérants des performances élevées en temps réel. Ainsi, le modèle d'infonuagique présente des lacunes significatives en matière de charge, de temps réel, de bande passante de transmission, de consommation d'énergie, ainsi que de sécurité et de protection de la vie privée des données.

Une nouvelle tendance émerge depuis quelques années dans le domaine de l'informatique, où les fonctionnalités et les services du modèle infonuagique se déplacent de plus en plus vers les périphéries du réseau. Cette évolution ouvre des perspectives pour exploiter les vastes ressources de calcul et l'espace de stockage répartis dans presque tous les équipements aux bords du réseau. De telles ressources peuvent répondre aux besoins en matière de tâches intensives en calcul et de faible latence sur les appareils mobiles. Ce concept est connu sous le nom d'informatique multi-accès en périphérie (en anglais multi-access edge computing, MEC) (Mao *et al.*, 2016; Mao *et al.*, 2017; Cao *et al.*, 2020).

2.2.1 Caractéristiques clés du MEC

Le MEC présente plusieurs caractéristiques clés, notamment (Kong *et al.*, 2022; Cao *et al.*, 2020) :

1. **Proximité de la périphérie du réseau :** Les ressources de calcul et de stockage sont situées à proximité des utilisateurs finaux, réduisant la latence et améliorant la réactivité des applications.
2. **Latence réduite :** Grâce à la proximité de la périphérie, les applications peuvent offrir une faible latence, ce qui est essentiel pour les applications en temps réel telles que la réalité virtuelle et les jeux en ligne.
3. **Traitement décentralisé :** Les tâches de calcul sont décentralisées et réparties sur des serveurs en périphérie, réduisant la charge sur le réseau d'infrastructure (en anglais core network).

4. **Intégration avec la mobilité :** Le MEC est conçu pour prendre en charge la mobilité des utilisateurs, ce qui est essentiel pour les applications mobiles et IoT.

5. **Efficacité des réseaux :** Le MEC permet une utilisation plus efficace des ressources réseau en réduisant le trafic vers le réseau d'infrastructure, ce qui libère de la bande passante.

6. **Support de la 5G :** Le MEC est conçu pour s'intégrer harmonieusement avec les technologies 5G, offrant des performances accrues pour les applications en temps réel.

7. **Exploitation des appareils mobiles :** Il permet aux appareils mobiles d'accéder à des ressources de calcul en périphérie, offrant ainsi des fonctionnalités avancées sans surcharger les appareils.

2.2.2 Défis du MEC

Le MEC est confronté à plusieurs défis de conception (Mao *et al.*, 2017; Cao *et al.*, 2020):

1. **Hétérogénéité des périphériques :** La diversité des appareils en périphérie, tels que les téléphones intelligents, les capteurs IoT, et les véhicules connectés, nécessite une prise en charge adéquate.

2. **Gestion de la mobilité :** Assurer une continuité de service transparente lors du déplacement des utilisateurs est un défi, en particulier dans le contexte des applications en temps réel.

3. **Gestion de la sécurité :** Protéger les données sensibles et les applications exécutées en périphérie est essentiel pour gérer les vulnérabilités.

4. **Coordination entre les serveurs :** La coordination entre les serveurs en périphérie et le réseau d'infrastructure pour assurer une utilisation efficace des ressources constitue un défi technique.

5. **Ordonnancement des tâches :** L'ordonnancement des tâches sur les serveurs en périphérie doit être optimisé pour garantir une utilisation efficace des ressources.

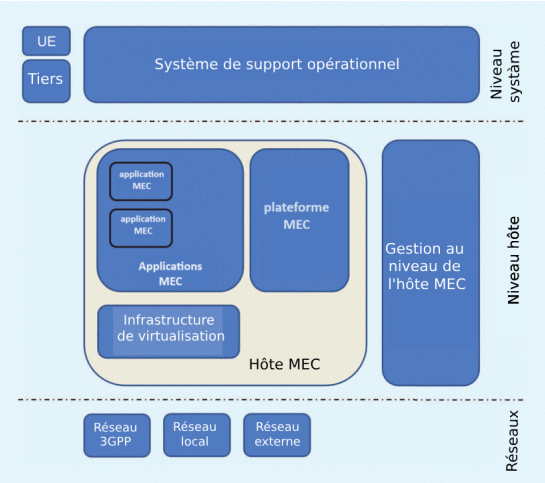

Figure 2.2 Architecture de l'écosystème MEC

Source (Sabella *et al.*, 2016)

2.2.3 Écosystème

Comme illustré dans la figure 2.2, nous présentons une vue de haut niveau du fonctionnement du MEC. Plusieurs entités fonctionnelles sont nécessaires et sont regroupées en trois niveaux : système, hôte, et réseau (Sabella *et al.*, 2016).

1. Niveau hôte : Comprend un hôte MEC et une entité de gestion de niveau MEC. L'hôte MEC est divisé par la plateforme MEC, les applications MEC et une infrastructure de virtualisation.
2. Niveau réseau : On trouve des entités externes, telles que le réseau cellulaire du projet de partenariat de la troisième génération (en anglais 3rd Generation Partnership Project, 3GPP), les réseaux locaux et les réseaux externes. Cette couche représente la connectivité aux réseaux locaux, aux réseaux cellulaires et aux réseaux externes tels que l'internet.
3. Niveau système : Il a une vue d'ensemble du système et se compose des hôtes MEC et de la gestion MEC nécessaire pour exécuter des applications MEC au sein d'un réseau opérateur ou d'un sous-ensemble d'un réseau opérateur.

Le fonctionnement du MEC repose sur plusieurs entités interconnectées, chacune jouant un rôle

spécifique dans l'écosystème. Voici une explication détaillée de ces éléments clés :

1. **Applications MEC:** Les applications MEC s'exécutent sous forme de machines virtuelles au sein de l'infrastructure de virtualisation fournie par l'hôte MEC.
2. **Infrastructure de virtualisation:** L'infrastructure de virtualisation comprend un plan de données qui exécute les règles de transfert reçues par la plateforme MEC et achemine le trafic entre les applications, les services et les réseaux.
3. **Plateforme MEC :** Les fonctionnalités de base essentielles de la plateforme MEC sont nécessaires pour diriger le trafic entre les applications, les services et les réseaux.
4. **Hôte MEC :** L'hôte MEC est une entité composée de la plateforme MEC et de l'infrastructure de virtualisation. Cette infrastructure fournit des ressources de calcul, de stockage et de réseau aux applications MEC.
5. **Gestion au niveau de l'hôte MEC :** La gestion au niveau de l'hôte MEC est une entité de niveau hôte qui se divise en gestion des éléments de la plateforme MEC, gestion du cycle de vie des applications MEC, et gestion des règles et des exigences des applications MEC. La gestion du cycle de vie des applications inclut les procédures d'instanciation et de terminaison des applications.
6. **Gestion au niveau du système :** La gestion au niveau du système est utilisée pour la configuration et la gestion des performances de la plateforme MEC. Elle contribue à exécuter les applications MEC à l'emplacement souhaité dans le réseau.

La gestion au niveau du système reçoit des demandes d'instanciation et de terminaison des applications MEC de l'UE. Les demandes approuvées par le système de support opérationnel (en anglais operations support system, OSS) sont transmises à la gestion au niveau de l'hôte pour un traitement ultérieur. La gestion au niveau du système peut également avoir la capacité de déplacer les applications entre différents hôte MEC.

Ces différentes entités interagissent de manière synergique pour fournir un environnement de calcul en périphérie capable de répondre aux besoins des applications tout en optimisant l'utilisation des ressources disponibles.

2.3 Conclusion

Ce chapitre introduit les bases essentielles pour la compréhension de notre étude. Nous avons exploré deux concepts clés qui jouent un rôle fondamental dans notre recherche : la communication en mode multidiffusion et le calcul en périphérie. La communication en mode multidiffusion offre la possibilité d'envoyer efficacement des données à un groupe de destinataires sans générer de copies inutiles, tandis que le calcul en périphérie exploite les capacités de calcul informatiques près des utilisateurs, répondant ainsi aux exigences croissantes de faible latence et de calcul intensif.

Ces concepts forment la base de notre travail, car nous cherchons à tirer parti de la multidiffusion pour améliorer la communication dans les réseaux sans fil et à utiliser le calcul en périphérie pour optimiser la gestion des ressources. Dans les chapitres suivants, nous approfondirons ces concepts et les appliquerons à notre étude de cas.

CHAPITRE 3
ÉTAT DE L'ART

Ce chapitre offre une revue de l'état actuel de la recherche dans les domaines clés liés à notre sujet d'étude, établissant ainsi le contexte nécessaire à la compréhension de notre contribution. Nous examinerons la littérature récente en relation avec la multidiffusion dans les réseaux sans fil, la conception d'algorithmes pour le problème de déchargement de tâches dans les systèmes MEC et la combinaison de la multidiffusion dans ces derniers.

3.1 Multidiffusion dans les réseaux sans fil

La littérature qui étudie la multidiffusion dans les réseaux sans fil est très riche et englobe un très grand nombre de travaux de recherche. Les articles (Vella et Zammit, 2012) et (Araniti *et al.*, 2017) présentent des survols pertinents de ces ensembles de travaux.

L'étude de (Vella et Zammit, 2012) plonge dans les efforts de recherche déployés depuis la fin des années 90 pour examiner la multidiffusion sur les réseaux d'accès sans fil. Mettant l'accent sur la multidiffusion vidéo, cette étude explore les défis inhérents au support sans fil propice aux erreurs. L'étude souligne l'importance du codage extensible et explore des propositions pour optimiser la transmission en multidiffusion.

L'étude bibliographique présentée dans (Araniti *et al.*, 2017) se concentre sur les défis croissants de la multidiffusion dans les réseaux 5G. Les auteurs mettent en évidence l'expansion rapide des applications mobiles de multidiffusion, touchant à la fois les utilisateurs finaux et les communications de type machine pour l'internet des objets (en anglais internet of things IoT). L'article examine les défis liés à la charge croissante des communications de type machine dans la 5G, remettant en question l'efficacité des services actuels. Il offre une analyse critique des réalisations de pointe en multidiffusion, mettant en avant l'amélioration des débits de données et l'utilisation du spectre. L'étude effectuée souligne toutefois le manque de contributions significatives dans les communications de type machine basées sur des groupes, appelant à des améliorations substantielles de

l'architecture actuelle.

Le problème de la multidiffusion a attiré une attention particulière, avec des travaux antérieurs se concentrant sur des aspects tels que l'optimisation de l'ordonnancement des requêtes, l'efficacité énergétique, la gestion de la mémoire cache, et bien plus encore.

La plupart des travaux antérieurs traitant de la multidiffusion se sont concentrés sur le développement d'algorithmes efficaces de planification et d'allocation de ressources, en supposant l'arrivée synchronisée des demandes, sans tenir compte du cas pratique des arrivées asynchrones. Le travail dans (Huang *et al.*, 2016) comble cette lacune en tenant compte des arrivées aléatoires de demandes et en prenant en considération la consommation d'énergie à la station de base, explorant le compromis entre le délai et la consommation d'énergie. L'article étudie le compromis entre le délai et l'efficacité énergétique dans un problème d'ordonnancement au sein d'un réseau de multidiffusion, où les contenus sont mis en cache au niveau des stations de base. Il propose une approche markovienne pour la formulation d'une solution optimale, ainsi qu'un algorithme sous-optimal basé sur les propriétés de la solution optimale avec une complexité réduite.

Le problème d'ordonnancement en multidiffusion pour plusieurs messages sur plusieurs canaux est également traité par (Li *et al.*, 2022b). L'article présente un algorithme modifié d'apprentissage profond par renforcement (en anglais deep reinforcement learning, DRL) appelé le « Wolpertinger Deep Deterministic Policy Gradient » basé sur les permutations (PW-DDPG). Leur approche en DRL atteint des résultats comparables à l'optimal qui peut être atteints en utilisant l'algorithme de (Huang *et al.*, 2016).

L'article (Somuyiwa *et al.*, 2019) aborde le problème de la minimisation du coût moyen à long terme des communications sans fil, en se concentrant sur la transmission en multidiffusion et la mise en cache proactive. Contrairement aux articles (Huang *et al.*, 2016) et (Li *et al.*, 2022b), la mise en cache est proactive, ce qui signifie que les contenus ont des durées de vie et nécessitent une politique de mise en cache. Le problème est formulé comme un processus de décision markovien, et les auteurs proposent une politique basée sur le DRL, en utilisant spécifiquement la méthode du gradient de politique déterministe profonde pour le résoudre. La solution proposée surpasse

les techniques de transmission multidiffusions et de mise en cache de référence.

L'article (Guo *et al.*, 2018) explore l'utilisation de la multidiffusion pour résoudre deux problèmes: 1) minimiser la consommation d'énergie moyenne tout en garantissant une qualité vidéo VR à 360 degrés, et 2) maximiser la qualité de réception vidéo avec un budget d'énergie donné. Les auteurs ont obtenu des solutions de forme fermée (en anglais closed-form solution) pour ces deux problèmes non convexes complexes. Leurs résultats surpassent ceux de deux méthodes de référence, la première utilise exclusivement l'envoi individuel, ainsi que la deuxième se base sur une multidiffusion avec une allocation égale du temps de transmission.

Les auteurs de (Hao *et al.*, 2020) explorent la planification coopérative de la multidiffusion entre plusieurs stations de base dans des réseaux 5G dotés de mémoire cache. L'objectif de ce travail est de répondre aux demandes des utilisateurs tout en minimisant la consommation d'énergie. À la différence des articles précédents, plusieurs stations de base sont prises en compte. Les auteurs proposent un algorithme centralisé en ligne pour obtenir la stratégie optimale, ainsi qu'un algorithme distribué présentant des performances similaires, mais avec une complexité bien moindre. En comparaison avec l'état de l'art, leurs algorithmes se révèlent plus performants en termes de consommation d'énergie et de délais de service.

Le travail de (Zhou *et al.*, 2018) aborde le problème de la prise de décision de multidiffusion et de la formation de faisceaux consciente de la mémoire cache de contenu dans les réseaux hétérogènes. Le travail propose un algorithme heuristique reposant sur plusieurs hypothèses qui optimise conjointement le coût de la communication et le coût de la mémoire cache. Les résultats de simulation ont confirmé la supériorité des techniques proposées.

3.2 Métaheuristiques pour le déchargement de tâches vers le MEC

Comme discuté dans le chapitre 2, pour répondre aux capacités limitées des UE, le déchargement vers le MEC est employé pour les tâches gourmandes en calcul ou présentant des contraintes de latence. Le MEC a constitué, pendant plus d'une décennie, un domaine de recherche qui a suscité beaucoup d'attention et d'intérêt dans les mondes de la recherche et de l'industrie. Une revue

récente de quelques travaux de recherche pertinents consacrés à la problématique du déchargement vers les serveurs MEC sera présentée dans ce qui suit. Cette section entreprend une revue de quelques travaux récents se consacrant au déchargement vers les serveurs MEC, en se penchant particulièrement sur ceux qui reposent sur l'utilisation des métaheuristiques, et notamment sur les approches génétiques.

Plusieurs algorithmes métaheuristiques sont utilisés pour résoudre les défis du déchargement dans le MEC. L'article (He *et al.*, 2021) se concentre sur le problème du déchargement partiel et de l'allocation des ressources dans le but de minimiser le délai sous contrainte énergétique dans un système MEC avec plusieurs utilisateurs. Afin de réduire le retard total de traitement tout en respectant les contraintes énergétiques, l'article développe une métaheuristique de planification efficace basée sur l'algorithme de la colonie d'abeilles artificielles (en anglais artificial bee colony, ABC). Cette approche comprend un opérateur de mappage basé sur la position et une stratégie de déchargement de tâches gloutonne. Les performances de l'algorithme proposé surpassent celles des algorithmes de référence en diminuant le délai de traitement des tâches.

Un autre article (Abbas *et al.*, 2021) aborde la problématique de l'optimisation de la consommation d'énergie et de la minimisation de la latence d'exécution dans un contexte de déchargement de tâches en MEC. Il propose une sélection optimale des tâches de déchargement en utilisant des métaheuristiques bien connues, à savoir l'algorithme d'optimisation de la colonie de fourmis, l'algorithme d'optimisation des baleines et l'algorithme d'optimisation du loup gris, en adaptant la conception de ces algorithmes selon les besoins du problème. Les résultats démontrent que l'optimisation du loup gris surpasse les autres en termes d'optimisation de la consommation d'énergie et de la latence d'exécution, tout en sélectionnant un ensemble optimal de tâches de déchargement.

L'algorithme génétique est largement utilisé dans le contexte du déchargement de tâches en MEC. Par exemple, (Zalat *et al.*, 2022) se concentre sur la prise de décision de déchargement de tâches dans un environnement d'informatique en nuage mobile sur plusieurs sites. Une combinaison d'un algorithme génétique niché avec un processus de décision markovien est proposée pour la prise

de décision de déchargement au niveau du nuage. Le processus de décision markovien est utilisé pour déterminer l'emplacement le plus optimal pour chaque tâche, tandis que l'algorithme génétique est utilisé pour déterminer la probabilité de transition optimale pour les composants opérant sur plusieurs sites. Les résultats révèlent que la technique proposée consomme peu d'énergie et s'exécute rapidement tout en déterminant la décision de déchargement optimale avec un nombre de générations plus faible.

Dans un autre scénario, l'article (Zhu et Wen, 2021) examine l'impact de la décision de déchargement et de l'allocation des ressources sur les performances du déchargement de calcul, soulignant la difficulté d'atteindre une efficacité optimale du déchargement de calcul. Pour remédier à cela, l'article propose une stratégie de déchargement de tâches de calcul en périphérie informatique basée sur un algorithme génétique amélioré. L'approche adopte la somme pondérée du retard d'exécution des tâches et de la consommation d'énergie en tant que fonction objectif. Bien que l'algorithme proposé affiche de meilleurs résultats en termes de retard, ses performances en matière de consommation d'énergie sont considérées comme moyennes. Il est à noter que l'impact de la mobilité de l'utilisateur sur le système n'est pas pris en compte, tout comme dans notre travail.

L'algorithme génétique est également utilisé dans un environnement de déchargement en informatique en nuage (Pirozmand *et al.*, 2021). Il propose également un algorithme génétique qui réduit à la fois la consommation énergétique et le retard des tâches. L'algorithme proposé affiche un meilleur temps d'exécution total et une meilleure consommation d'énergie par rapport à plusieurs algorithmes métaheuristiques.

L'article (Sinthiya *et al.*, 2022) se penche sur la problématique de prise de décision optimale du serveur MEC, orientant les utilisateurs vers le déchargement sur le serveur MEC ou le serveur en nuage. En raison de la NP-dureté du problème, une métaheuristique génétique nommée "Cost Effective Genetic Algorithm for Tasks Offloading" (CEGA) a été présentée, permettant au MEC de catégoriser les utilisateurs en deux groupes et de les guider vers le déchargement soit sur le MEC, soit sur le serveur nuage. L'analyse des performances de (CEGA) a révélé des améliorations signi-

ficatives en termes de coût moyen pour les utilisateurs, de retard moyen et d'exigence énergétique moyenne pour le déchargement, se démarquant par rapport aux travaux de pointe. Il est à noter que l'exécution locale n'a pas été prise en considération.

L'article (Du *et al.*, 2019) étudie le problème de minimisation de la charge totale du système MEC en optimisant conjointement la prise de décision de déchargement des calculs et l'allocation des canaux de communication. Les utilisateurs mobiles doivent non seulement déterminer s'ils doivent décharger, mais aussi décider du canal à décharger. Ils utilisent un algorithme génétique (AG) pour résoudre ce problème d'optimisation en minimisant la somme pondérée du délai d'exécution et de la consommation d'énergie de chaque utilisateur mobile. Les résultats montrent que l'algorithme peut obtenir de meilleures performances que d'autres algorithmes de référence.

La prise de décision pour le déchargement dans un réseau MEC, impliquant des utilisateurs dans des scénarios de mobilité élevée, est traitée dans l'article (Li *et al.*, 2021). Ce dernier présente un schéma prédictif de déchargement de calcul et d'ordonnancement des tâches basés sur un algorithme génétique pour résoudre les problèmes de déchargement et d'ordonnancement des tâches des utilisateurs dans des environnements à mobilité élevée. L'algorithme détermine le temps de séjour des utilisateurs pour planifier les dechargements tâches. Les résultats montrent que l'algorithme proposé parvient à réduire efficacement le taux d'échec du déchargement et surpasse d'autres algorithmes traditionnels en termes d'amélioration des performances. Cet article (Al-Habob *et al.*, 2020) se penche sur la planification des tâches pour le déchargement séquentiel et parallèle d'une tâche à la fois intensive en calcul et sensible à la latence vers plusieurs serveurs MEC. Dans le schéma de déchargement séquentiel, le dispositif mobile décharge les sous-tâches vers les serveurs de manière séquentielle sur un canal partagé. En revanche, dans le schéma de déchargement parallèle, le dispositif mobile décharge simultanément les sous-tâches vers les serveurs par des sous-canaux orthogonaux. Des solutions heuristiques basées sur un algorithme génétique et des modèles de graphe de conflit ont été élaborées. Le déchargement séquentiel présente une probabilité d'échec de déchargement plus faible et nécessite un nombre moins élevé de serveurs. D'un autre côté, le déchargement parallèle offre une latence moindre. Ces algorithmes démontrent des performances se rapprochant de la solution optimale, obtenue par une

recherche exhaustive.

L'article (Li *et al.*, 2022a) explore le déchargement de tâches au sein des systèmes informatiques embarqués dans les véhicules. Les tâches peuvent être attribuées non seulement au serveur MEC proche, mais aussi à des utilisateurs riches en ressources à proximité. Les auteurs ont proposé une délégation de tâches basée sur un algorithme génétique pour les véhicules électriques intelligents, visant à minimiser la consommation d'énergie du système. Les résultats démontrent que notre algorithme parvient à atteindre l'objectif de conservation d'énergie et surpasse la référence en termes de performances, par rapport aux scénarios où les tâches sont exécutées localement.

Une autre catégorie d'approches pour le déchargement de tâches vers le MEC s'appuie sur l'apprentissage profond par renforcement (en anglais deep reinforcement learning, DRL). Ces méthodes utilisent des réseaux neuronaux profonds pour apprendre des politiques de déchargement de manière autonome. L'étude (Zhang *et al.*, 2019) porte sur la prise de décision de déchargement dans les réseaux hétérogènes avec mobilité de l'UE. Elle propose un cadre de déchargement basé sur l'apprentissage par renforcement en ligne afin d'obtenir une politique optimale de contrôle de déchargement. La méthode démontre une robustesse face aux divers contextes environnementaux mobiles. Les résultats montrent que l'algorithme se rapproche de l'optimal. Il convient de noter que cette étude se concentre spécifiquement sur le cas d'un seul UE. De même, les auteurs de (Wang *et al.*, 2020) réalisent une étude conjointe du déchargement de tâches et de la migration dans un réseau MEC conscient de la mobilité. Ils proposent également une approche basée sur l'apprentissage par renforcement pour résoudre le problème MINLP formulé pour la décision de déchargement. Les performances surpassent celles des méthodes de référence ainsi que d'un algorithme génétique présenté.

Enfin, (Qu *et al.*, 2021) examine la prise de décision de déchargement dans un environnement MEC connecté au nuage. Une tâche peut s'exécuter soit localement sur le serveur MEC, soit au niveau du nuage. L'article propose un algorithme de déchargement basé sur l'apprentissage profond par renforcement méta (DMRO), qui combine plusieurs réseaux de neurones profonds parallèles avec le Q-learning pour prendre des décisions de déchargement fines. L'apprentissage méta permet au

modèle de s'adapter plus rapidement aux changements d'environnement. La solution proposée a un effet plus bénéfique sur les décisions de déchargement de tâches par rapport aux schémas de déchargement binaire et aux schémas de déchargement partiel basés sur l'apprentissage profond par renforcement (DRL) conventionnel. D'autres approches méritent également d'être mentionnées. L'article (Yang *et al.*, 2020) a étudié l'optimisation du temps de déchargement de tâches dans le MEC avec plusieurs serveurs en considérant la mobilité. Il propose une stratégie de sélection optimale des serveurs de déchargement basée sur une formulation du problème markovien afin de réduire le temps total de déchargement. La stratégie proposée est plus performante comparée à d'autres algorithmes de sélection de serveurs.

De plus, l'article (Mao *et al.*, 2016) examine les décisions de déchargement de tâches dans un système MEC avec des UE capables de récupérer de l'énergie. Il propose un algorithme dynamique à faible complexité basé sur l'optimisation de Lyapunov, qui détermine à la fois la décision de déchargement, les fréquences de cycles CPU et la puissance de transmission à utiliser. Les performances dépassent celles des méthodes de référence, et une réduction des échecs d'exécution de tâches est également observée.

3.3 Approches combinées

La majorité des études des problèmes de déchargement et d'allocation de ressources dans le MEC considèrent uniquement les envois individuels. Toutefois, l'emploi de la multidiffusion peut apporter un avantage considérable dans ce type de réseaux et servir à améliorer les performances de plusieurs de ses applications telles que le traitement de vidéos à haute résolution et la réalité virtuelle. Dans cette section, nous présentons les quelques travaux qui considèrent la combinaison gagnante de l'informatique en périphérie et de la multidiffusion.

Le problème de l'allocation des ressources dans un environnement MEC considérant l'envoi par multidiffusion dans un réseau hétérogène est étudié dans l'article (Hao *et al.*, 2021). Les auteurs proposent une solution intelligente pour la prise de décision de la mise en mémoire cache (en anglais content caching) et l'allocation des ressources. L'évaluation des performances montre que la solution proposée surpasse des solutions alternatives dans différentes conditions en termes

de délai moyen et de consommation d'énergie. Il est important de noter que même si ce travail considère une coopération entre les stations de base, il ne prend pas de décisions de multidiffusion ; il suppose que les requêtes seront répondues avec la multidiffusion.

L'article (Yuan *et al.*, 2023) étudie le problème de maximisation de la capacité d'un système MEC, en faisant face aux défis de la dynamité des sessions des utilisateurs et de la popularité du contenu demandé. L'article a analysé les bornes supérieures et inférieures de la capacité du système. La borne supérieure est déterminée en considérant l'arbre k-aire complet, et la borne inférieure en supposant que la communication en multidiffusion se produit uniquement au dernier saut. Les résultats confirment la supériorité de la solution proposée. Il convient de souligner que le problème de déchargement de tâches n'est pas pris en compte dans ce contexte de coopération entre stations de base, où un contenu peut transiter par plusieurs stations de base.

Une collaboration entre les stations de base dans un réseau hétérogène est également exploitée dans l'article (Huang *et al.*, 2017). L'article propose une stratégie de mise en cache coopérative consciente de la multidiffusion visant à réduire la latence moyenne de livraison du contenu. Les performances de la solution proposée surpassent celles sans coopération. Toutefois, de même que (Hao *et al.*, 2021), les décisions de multidiffusion ne sont pas pleinement explorées ici.

Le problème de déchargement de tâches en multidiffusion est étudié dans l'article (Li *et al.*, 2020) dans les réseaux véhiculaires. Il se concentre sur les services de calcul en périphérie fournis par les véhicules équipés de capacités de calcul. Les unités routières envoient en multidiffusion un certain nombre de tâches aux véhicules pour traitement. L'article propose un algorithme basé sur la méthode du point intérieur pour résoudre le problème d'optimisation en optimisant le retard moyen des tâches. L'algorithme proposé réduit significativement le retard des tâches par rapport à d'autres solutions de base.

Une nouvelle approche est adoptée dans l'article (Sun *et al.*, 2020), où toutes les données d'entrée et de sortie des tâches sont disponibles dans la mémoire cache du serveur MEC. Les auteurs de l'article abordent le problème de minimisation de la bande passante de transmission moyenne tout en optimisant la décision de déchargement dans un réseau MEC prenant en compte la mul-

tidiffusion, le tout sous contrainte de latence, de consommation d'énergie et de mise en cache. Ils proposent un algorithme à faible complexité basé sur la procédure convexe concave en conjonction avec la méthode des multiplicateurs de direction alternée. Les performances de cette solution surpassent celles de l'algorithme de référence, et pour un nombre restreint d'utilisateurs et de tâches, elle tend vers l'optimal.

Contrairement à l'article (Sun *et al.*, 2020), notre approche diffère en ce sens que le serveur MEC n'est pas uniquement dédié au stockage de données. Notre étude porte à la fois sur la prise de décision en matière de déchargement des tâches, et selon cette décision, un transfert de données peut concerner aussi bien l'entrée que la sortie. La sortie des tâches de déchargement est souvent négligée dans d'autres travaux, cependant, nous la considérons comme un aspect pratique important, notamment dans des applications telles que la VR.

3.4 Position de ce travail par rapport à la littérature

Une différence significative de notre recherche par rapport aux travaux présentés plus haut réside dans la prise en compte de la combinaison efficace de transmissions en multidiffusion et d'envois individuels. Alors que de nombreuses approches se limitent à un seul mode de transmission, notre étude reconnaît l'importance d'une approche hybride. Cette combinaison offre une flexibilité accrue, permettant de répondre de manière adaptative aux divers besoins des utilisateurs.

Une autre caractéristique distinctive de notre recherche réside dans la gestion des données au niveau du serveur MEC. Contrairement à la majorité des approches proposées dans la littérature qui supposent souvent la disponibilité des données à traiter au niveau des UE, nous nous penchons sur l'optimisation de la transmission des données depuis le serveur vers les utilisateurs. Cela implique une considération particulière du lien descendant (en anglais downlink) pour garantir une efficacité optimale du déchargement, qu'il soit effectué localement ou au niveau du MEC. Les entrées et sorties des tâches sont transférées du MEC aux utilisateurs en fonction de l'emplacement choisi pour l'exécution, que ce soit en local ou au niveau du MEC.

Une contribution de notre travail réside dans l'application de métaheuristiques à un problème

combinant le MEC et la multidiffusion. Tandis que de nombreuses études se sont concentrées sur l'optimisation individuelle de ces deux aspects, notre approche consiste à les intégrer dans un problème unifié. Nous investiguons comment les métaheuristiques peuvent être adaptées pour résoudre efficacement ce problème complexe, ouvrant ainsi la voie à des solutions optimisées et flexibles pour les scénarios MEC/multidiffusion.

En somme, ces éléments définissent la spécificité et l'originalité de notre recherche par rapport à l'état de l'art.

3.5 Conclusion

En synthèse, notre exploration de l'état de l'art révèle une attention croissante envers la multidiffusion et le MEC. Les études se sont concentrées sur l'optimisation de la multidiffusion et les défis liés aux décisions de déchargement dans le MEC. Les approches basées sur l'apprentissage, les métaheuristiques, et la coopération entre stations de base ont émergé comme des solutions prometteuses.

Cependant, des lacunes persistent, notamment dans la prise de décision de déchargement en multidiffusion. Notre travail se positionne pour combler ces lacunes en proposant une approche intégrant la multidiffusion avec des décisions de déchargement, visant à améliorer les performances des réseaux MEC, en particulier pour des applications intensives en calcul.

CHAPITRE 4

MODÈLE DE SYSTÈME ET FORMULATION DU PROBLÈME

La planification des tâches dans un environnement MEC est un défi complexe qui nécessite une compréhension approfondie des interactions entre les principaux composants du réseau. Ce chapitre pose les bases essentielles de notre étude en présentant la modélisation du système considéré. Nous décrirons en détail les composants clés de ce système, notamment la modélisation des tâches, des requêtes, de la mise en cache, de la communication, et du calcul. Ensuite, nous formulons le problème de planification des requêtes de manière mathématique en précisant la fonction objectif et les contraintes de notre système. La formulation du problème est cruciale pour développer des algorithmes efficaces pour sa résolution.

4.1 Modèle du système

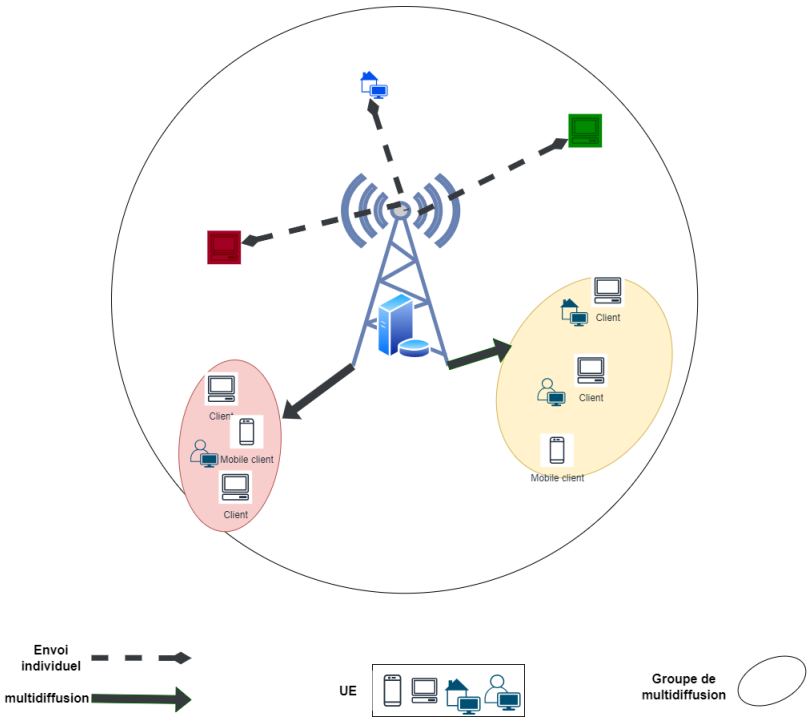

Figure 4.1 Modèle de système illustrant l'intégration de la multidiffusion, du MEC et de la mise en cache dans un réseau sans fil

Comme illustré dans la figure 4.1, nous considérons un système MEC composé d'une seule sta-

tion de base (BS) ayant un accès direct à un serveur MEC et de N équipement utilisateur (UE). L'ensemble des UE est représenté par $\mathcal{N} = \{1, 2, \ldots, N\}$. Le système MEC fonctionne sur une bande de fréquence ayant une largeur de B (en Hz). Le serveur MEC dispose d'une mémoire de cache de D bits et une capacité de calcul de F cycles CPU par seconde. Un modèle en temps discret est adopté, où le temps est divisé en intervalles de longueur égale T_s (en secondes) et indexés par $\mathcal{T} = \{0, 1, \ldots, T\}$.

Dans notre modèle, nous adoptons une approche hors ligne, présupposant que la station de base possède une connaissance complète des requêtes entrantes. Cette supposition vise à explorer les limites supérieures des performances dans des conditions idéalisées. Nous reconnaissons que les performances obtenues dans un scénario hors ligne peuvent différer de celles observées dans des situations en ligne, où les informations sur les requêtes peuvent évoluer dynamiquement. Cependant, notre choix méthodologique offre une base solide pour évaluer les algorithmes et les stratégies dans des conditions de planification plus prévisibles.

L'ensemble des notations utilisées dans ce chapitre est répertorié dans le Tableau 4.1 pour référence.

Tableau 4.1 Notations utilisées dans la formulation du problème

Notation	Description
$N, \mathcal{N}$	Nombre, ensemble des utilisateurs mobiles
B	Bande passante disponible à la BS (en Hz)
D	Taille du cache de la BS (en bits)
F	Capacité de calcul de la BS (en cycles CPU par seconde)
$\mathcal{I}$	Ensemble des données d'entrée
T_s	Taille de l'intervalle de temps (en secondes)
$\mathcal{T}$	Ensemble des intervalles de temps
$\mathcal{K}$	Ensemble des tâches
ψ_k	Ressources de calcul requises pour la tâche k (Hz)
I_k	Taille des données d'entrée de la tâche k (en bits)
O_k	Taille de la sortie de la tâche k (en bits)
β_k	Rapport de la taille de sortie à la taille de l'entrée pour la tâche k
ω_k	Rapport de la ressource de calcul requise pour la tâche k à la taille de l'entrée
$\mathcal{A}$	Ensemble des requêtes
κ_a	Tâche à exécuter dans la requête a
ι_a	Entrée de la tâche dans la requête a
τ_a	Intervalle de temps d'arrivée de la requête a
δ_a	Échéance de la requête a
$Z_i(t)$	Indicateur de mise en cache pour l'entrée i
$Z_{ki}^{\text{out}}(t)$	Indicateur de mise en cache pour la sortie de la tâche k sur i
$X_{k,i}(t)$	Indicateur de transmission d'entrée
$Y_{k,i}(t)$	Indicateur de transmission de sortie
$U_{n,a}$	Indicateur de suivi des requêtes qui respectent les contraintes de latence
$T_{n,k,i}$	Temps total d'exécution de la requête $A_{a,n}$

4.1.1 Modèle des tâches et des requêtes

L'ensemble de tâches est désigné par $\mathcal{K} = \{1, 2, \ldots, k, \ldots, K\}$ et nous supposons que l'ensemble $\mathcal{I} = \{1, 2, \ldots\}$ des données d'entrée de toutes les tâches est stocké dans la mémoire cache du serveur MEC. Chaque tâche $k \in \mathcal{K}$ est caractérisée par deux fonctions : l'une qui transforme la taille I (en bits) de l'entrée i en besoins en ressources, et l'autre qui transforme la taille I de l'entrée i en taille de sortie O (en bits). Pour simplifier la présentation, ces deux fonctions sont exprimées en termes de rapports, respectivement ω_k et β_k, définis comme suit :

$$\omega_k = \frac{\psi_k}{I_i} \quad \text{et} \quad \beta_k = \frac{O_k}{I_i}, \tag{4.1}$$

où ψ_k (en cycles par seconde) représente les besoins en ressources de la tâche avec l'entrée i.

L'ensemble des requêtes est représenté par $\mathcal{A}(t) = \{a_1(t), a_2(t), \ldots\}$. Chaque requête α peut être définie par le tuple $(n_\alpha, \kappa_\alpha, \iota_\alpha, \tau_\alpha, \delta_\alpha)$ où:

- n_α est le UE ayant généré la requête;
- κ_α représente la tâche à exécuter avec $\kappa_\alpha \in \mathcal{K}$;
- ι_α désigne les données d'entrée, appartenant à l'ensemble $\mathcal{I}$;
- τ_α correspond à l'intervalle de temps d'arrivée de la requête;
- δ_α est l'échéance de la requête, indiquant l'intervalle de temps auquel la tâche doit être terminée au plus tard.

Comme mentionné précédemment, la taille de la sortie est calculée en utilisant le rapport β_k.

Un exemple de scénario d'application pour de telles requêtes inclut la planification de trajectoires, où toutes les informations nécessaires sont disponibles au niveau du serveur. D'autres exemples de requêtes similaires et qui se caractérisent par des besoins de calcul intensif et des échéances

strictes sont présents dans le domaine de la réalité virtuelle (en angalis virtual reality VR), où les utilisateurs mobiles effectuent le rendu de scènes depuis différents points de vue. Ces applications nécessitent des capacités de calcul importantes, et les données sont souvent stockées au niveau du serveur. En outre, des tâches telles que la reconnaissance et le suivi d'objets entrent également dans cette catégorie.

4.1.2 Modèle de mise en cache

Dans le modèle de mise en cache, deux variables principales sont définies pour indiquer si certaines données sont actuellement stockées en cache ou non.

$$Z_i(t) = \begin{cases} 1, & \text{si l'entrée } i \text{ est disponible en cache pendant } t, \\ 0, & \text{sinon.} \end{cases} \tag{4.2}$$

$$Z_{ki}^{\text{out}}(t) = \begin{cases} 1, & \text{si la sortie } o \text{ de la tâche } k \text{ sur l'entrée } i \text{ est disponible en cache pendant } t, \\ 0, & \text{sinon.} \end{cases} \tag{4.3}$$

Ces variables sont essentielles pour gérer le comportement de la mise en cache dans l'environnement de calcul étudié. La mise en cache peut améliorer considérablement les performances en stockant les données fréquemment utilisées ou récemment calculées dans une mémoire à accès rapide, réduisant ainsi le temps nécessaire pour récupérer ces données dans le futur.

4.1.3 Modèle de communication

La communication entre la station de base et l'UE se déroule conformément à l'illustration de la figure 4.2, en trois étapes. Au début, (1) la BS envoie une trame d'annonce (en anglais Beacon frame) en diffusion afin d'atteindre tous les UE; elle permettrait notamment d'effectuer une vérification de chaque UE pour déterminer s'il dispose de requêtes qui doivent être exécutées.

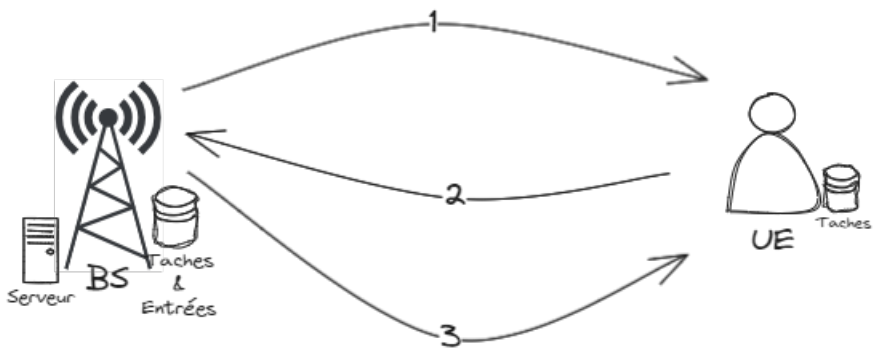

Figure 4.2 Étapes de communication pour le déchargement de tâches

(2) Chaque UE, par le biais de la liaison montante (en anglais uplink), transmet les informations nécessaires pour déclencher une tâche de calcul sur une donnée. L'ordonnanceur du serveur MEC décide soit de traiter la tâche au niveau du serveur, soit de transmettre les données afin que la tâche soit traitée au niveau de l'UE. (3) La BS communique ces décisions, incluant les formations de groupes et décisions d'ordonnancement aux différents UE qui se sont manifestés à l'étape (2).

Après les trois étapes de la figure 4.2, la communication entre la BS et un UE dont la tâche est ordonnancée, est réalisée en utilisant soit un mode d'envoi individuel (figure 4.3), où chaque UE transmet sa requête à la BS, qui la traite puis renvoie la réponse. Il s'agit d'une communication un-à-un.

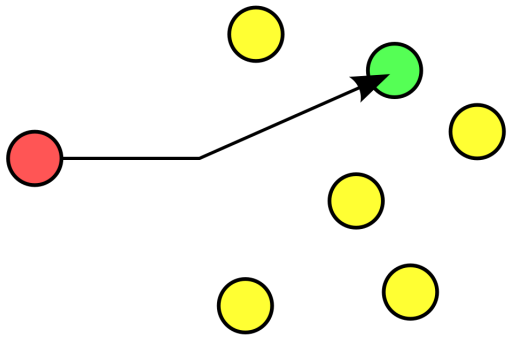

Figure 4.3 Mode de communication : Envoi individuel

La BS peut également former des groupes de UE en fonction des données qu'ils sont censés re-

cevoir, que ce soit une entrée, ou une sortie après l'exécution d'une tâche. La BS effectue une transmission en multidiffusion (figure 4.4) pour répondre à plusieurs UE du même groupe qui doivent recevoir le même contenu.

Le débit de transmission utilisé dans la multidiffusion doit tenir en considération la qualité du canal des UE à servir. Ce débit est choisi de manière à ce que tous les UE du groupe multidiffusion puissent recevoir la transmission de manière efficace.

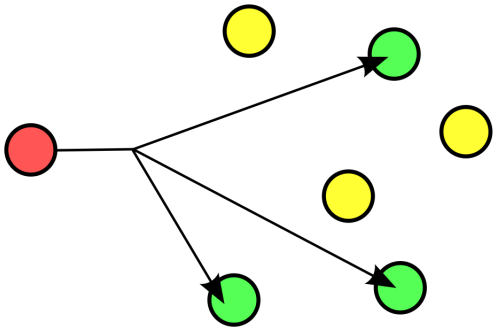

Figure 4.4 Mode de communication : Multidiffusion

Nous utilisons un modèle de canal de communication classique, couramment adopté dans la littérature (Lei *et al.*, 2023). Notre schéma de transmission repose sur des intervalles de temps pour les communications entre la BS vers les UE, et les requêtes des UE arrivent de manière stochastique. En supposant que $h_n(t)$ représente le gain de canal entre la BS et le UE n, le débit de transmission entre la BS et le UE est calculé comme suit :

$$R_n(t) = B \cdot \log\left(1 + \frac{P h_n(t)}{N_0}\right) \tag{4.4}$$

Ici, B représente la bande passante disponible pour la transmission, P est la puissance de transmission de la BS vers les UE, et N_0 est la puissance totale du bruit sur le canal. Le débit d'une multidiffusion dépendra des UE membres du groupe de transmission, noté $\mathcal{G} \subseteq \mathcal{N}$. Cette formule permet de déterminer le débit minimal parmi tous les utilisateurs d'un groupe de multidiffusion

pour les requêtes d'une tâche particulière:

$$R_{k,i}(t) = \min_{\forall n \in \mathcal{G}} R_n(t) \tag{4.5}$$

À chaque intervalle, la BS planifie la multidiffusion d'un contenu pour répondre aux requêtes en attente des UE. Le contenu transmis dépend de l'un des deux modes d'exécution dans notre système. Soit la BS transmet les données d'entrée des requêtes, soit elle exécute la tâche de la requête avec les données d'entrée au niveau serveur MEC et transmet les données de sortie. Nous utilisons des indicateurs pour différencier les décisions pour chaque requête. Désignons par $X_{n,i}(t) \in \{0, 1\}$ l'indicateur de transmission de l'entrée i au UE n pendant l'intervalle de temps t :

$$X_{n,i}(t) = \begin{cases} 1, & \text{si l'entrée } i \text{ est transmise au UE } n \text{ pendant } t. \\ 0, & \text{sinon.} \end{cases} \tag{4.6}$$

De manière similaire, la variable $Y_{n,k,i}(t)$ dénote un indicateur de transmission de la sortie de la tâche k sur les données d'entrée i pendant l'intervalle de temps t au UE n:

$$Y_{n,k,i}(t) = \begin{cases} 1, & \text{si la sortie } o \text{ de la tâche } k \text{ avec l'entrée } i \text{ est transmise au UE } n \text{ pendant } t. \\ 0, & \text{sinon.} \end{cases} \tag{4.7}$$

Il est essentiel de noter qu'à chaque intervalle de temps, le délai de transmission de la liaison montante est négligeable, et toutes les requêtes des dispositifs parviennent au serveur MEC au début de l'intervalle de temps.

4.1.4 Modèle de calcul

Chaque élément α de l'ensemble $\mathcal{A}$ définie par le tuple $(n_\alpha, \kappa_\alpha, \iota_\alpha, \tau_\alpha, \delta_\alpha)$ peut être traité de deux manières différentes. La décision est de savoir s'il faut exécuter la tâche localement ou au niveau du serveur. Si la décision prise est d'exécuter la requête localement, alors les données i de la requête sont envoyées vers le UE qui les traitent localement. Sinon, la tâche est exécutée au niveau du serveur et la sortie est ensuite téléchargée par le UE.

Exécution locale Dans l'approche d'exécution locale, le n-ième UE exécute sa tâche k avec l'entrée i localement en utilisant son propre CPU. Soit f_n la capacité de calcul (en cycles CPU par seconde) du UE n. Cette approche consiste en deux étapes. Tout d'abord, UE n télécharge les données d'entrée depuis le MEC, puis exécute la tâche. Le temps de téléchargement des données d'entrée est donné par :

$$T_\alpha^{\text{in}} = \frac{I_{\iota_\alpha}}{R_{\kappa_\alpha, \iota_\alpha}}, \tag{4.8}$$

Ensuite, le temps d'exécution de la tâche k lors de l'exécution locale avec l'entrée i peut être exprimé comme suit:

$$T_\alpha^{\text{exec}} = \frac{I_{\iota_\alpha} \cdot w_{\kappa_\alpha}}{f_{n_\alpha}}. \tag{4.9}$$

Le temps total pour une traitement local est donnée par:

$$T_\alpha^{\text{local}} = \frac{I_{\iota_\alpha}}{R_{\kappa_\alpha, \iota_\alpha}} + \frac{I_{\iota_\alpha} \cdot w_{\kappa_\alpha}}{f_{n_\alpha}} \tag{4.10}$$

Exécution au serveur MEC

Dans cette approche, le serveur MEC exécute la tâche au nom du UE. Cette approche se compose de deux étapes. Tout d'abord, le serveur MEC utilise ses ressources de calcul pour exécuter la tâche. Cependant, s'il s'avère que le résultat de la tâche est déjà en cache, Il peut être directement utilisé. Enfin, le serveur MEC renvoie les résultats de l'exécution de la tâche au UE n_α.

Au cours de la première étape de l'exécution de la tâche, le serveur MEC détermine le délai de traitement de la tâche κ_α avec l'entrée ι_α:

$$T_{\alpha}^{\text{exec}} = \frac{(1 - Z_{k,i}^{\text{out}}(t)) \cdot I_{\iota_\alpha} \cdot w_{\kappa_\alpha}}{F}, \tag{4.11}$$

Dans la deuxième étape, le UE télécharge les données de sortie depuis le serveur MEC ce qui ajoute le délai suivant:

$$T_{\alpha}^{\text{out}} = \frac{I_{\iota_\alpha} \cdot \beta_{\kappa_\alpha}}{R_{\kappa_\alpha, \iota_\alpha}}, \tag{4.12}$$

Le temps total pour un traitement au niveau du serveur MEC est donnée par:

$$T_{\alpha}^{\text{MEC}} = \frac{(1 - Z_{\kappa_\alpha, \iota_\alpha}^{\text{out}}(t)) \cdot I_i \cdot w_{\kappa_\alpha}}{F} + \frac{I_{\iota_\alpha} \cdot \beta_{\kappa_\alpha}}{R_{\kappa_\alpha, \iota_\alpha}} \tag{4.13}$$

En considérant les indicateurs pour les transmissions d'entrée et de sortie, $X_{n,i}$ et $Y_{n,k,i}$, le temps total d'exécution de chaque requête est défini comme suit :

$$T_{\alpha} = X_{k,i}(t) \cdot T_{\alpha}^{\text{local}} + Y_{k,i}(t) \cdot T_{\alpha}^{\text{MEC}} \tag{4.14}$$

4.2 Formulation du problème

L'objectif de cette étude est de maximiser le nombre de requêtes traitées tout en respectant les contraintes de latence et la taille de la mémoire cache du serveur MEC. Pour modéliser ce problème, trois ensembles de variables d'optimisation sont utilisés. En plus des variables $X_{n,i}(t)$ et $Y_{n,k,i}(t)$, nous définissons les variables U_α avec définie par le tuple $(n_\alpha, \kappa_\alpha, \iota_\alpha, \tau_\alpha, \delta_\alpha)$ qui servent à comptabiliser le nombre de requêtes qui respectent les contraintes de latence. Elle permettent notamment de quantifier précisément le niveau de conformité de ces requêtes aux délais.

$$U_\alpha = \begin{cases} 1, & \text{si pour la requête } \alpha, \text{le temps total } T_\alpha \leq \delta_\alpha. \\ 0, & \text{sinon.} \end{cases} \tag{4.15}$$

Le problème étudié peut être formulé mathématiquement de la manière suivante :

$$\underset{\mathbf{X},\mathbf{Y},\mathbf{Z},\mathbf{Z^{out}},\mathbf{U}}{\text{Maximiser}} \quad \sum_{\alpha \in \mathcal{A}} U_\alpha \tag{4.16}$$

sous les contraintes

$$\sum_{i \in \mathcal{I}} X_{n,i}(t) \leq 1 \qquad \forall t \in \mathcal{T}, \forall n \in \mathcal{N} \tag{4.17}$$

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}} Y_{n,k,i}(t) \leq 1 \qquad \forall t \in \mathcal{T}, \forall n \in \mathcal{N} \tag{4.18}$$

$$X_{n,i}(t) + Y_{n,k,i}(t) \leq 1 \qquad \forall t \in \mathcal{T}, n \in \mathcal{N}, i \in \mathcal{I}, k \in \mathcal{K} \tag{4.19}$$

$$\sum_{i} Z_i(t) I_i + \sum_{o_{k,i}} Z^{\text{out}}_{ki}(t) O_{o_{k,i}} \leq D \quad \forall t \in \mathcal{T} \tag{4.20}$$

$$X_{n,i}(t), Y_{n,k,i}(t), U_\alpha \in \{0, 1\} \tag{4.21}$$

où $\mathbf{X}$ est une matrice de taille $N \times |I| \times T$ regroupant les variables de décision $X_{n,i}$, $\mathbf{Y}$ une matrice de taille $N \times |I| \times K \times T$ regroupant les variables de décision $Y_{n,k,i}$, la matrice $\mathbf{z}$ de taille $|I| \times T$ regroupe les $Z_i(t)$, la matrice $\mathbf{Z^{out}}$ de taille $|I| \times K \times T$ regroupe les variables décision Z^{out}_{ki}. De plus, $\mathbf{U}$ est une matrice de taille $|\mathcal{A}| \times T$ regroupant les U_α.

La contrainte (4.17) impose qu'au plus une seule transmission de la donnée i par intervalle ait lieu. La contrainte (4.18) assure qu'au plus une seule transmission de la donnée de sortie de la tâche k ait lieu par intervalle. La contrainte (4.19) exige la non-chevauchement des transmissions d'entrée et de sortie. La contrainte (4.20) garantit que la somme des données stockées dans la mémoire cache ne dépasse pas la taille maximale. Enfin, la contrainte (4.21) s'assure que les variables $X_{n,i}(t)$, $Y_{n,k,i}(t)$ et U_α sont binaires.

Pour les solutions de développement dans le chapitre 5, nous supposons que les $\mathbf{Z}$ sont prédéfinis, et nous nous concentrons principalement sur les décisions de déchargement et d'ordonnancement.

Notre problème partage des similitudes substantielles avec le problème classique d'ordonnancement avec échéances, communément désigné sous le nom de « Job Shop Scheduling With Deadlines ». Les variables de décision, telles que $\mathbf{X}$ et $\mathbf{Y}$, ainsi que les contraintes formulées, notamment (4.17), (4.18), (4.19), (4.20), et (4.21), reflètent des caractéristiques fréquemment rencontrées dans

les problèmes d'ordonnancement.

Cette analogie met en lumière le fait que notre problème partage la complexité inhérente aux problèmes d'ordonnancement bien établis, en particulier ceux impliquant des contraintes temporelles strictes. Étant donné que la résolution des problèmes d'ordonnancement est reconnue comme étant NP-difficile, la ressemblance entre notre problème classiques suggère une difficulté à sa résolution. Ainsi, la complexité de notre problème peut être interprétée en considération des défis bien connus dans le domaine de l'ordonnancement.

CHAPITRE 5
SOLUTIONS PROPOSÉES

Dans ce chapitre, nous décrirons deux algorithmes heuristiques qui permettent d'obtenir des solutions au problème d'ordonnancement des requêtes formulé mathématiquement dans le chapitre précédent. Les deux algorithmes développés sont: une heuristique gloutonne et un algorithme génétique. Nous discuterons également de la complexité de ces deux approches.

5.1 Algorithme heuristique glouton (AHG)

5.1.1 Description de AHG

En raison de la complexité inhérente au problème étudié, cette section propose un algorithme heuristique glouton qui privilégie la réduction de la complexité temporelle plutôt que la recherche d'une solution optimale. L'AHG privilégie le traitement des requêtes en fonction de leur échéance la plus proche et sélectionne le premier intervalle de temps disponible pour les planifier. Les caractéristiques de l'algorithme heuristique glouton peuvent être décrites comme suit :

1. **Sélection de la requête :** L'algorithme priorise les requêtes ayant la date d'échéance la plus proche.
2. **Regroupement des requêtes :** Une fois la première requête sélectionnée, l'algorithme explore deux possibilités de regroupement pour cette requête : l'envoi des données d'entrée pour une exécution locale ou la transmission des données de sortie pour une exécution au niveau du MEC. Il choisit l'option qui permet de satisfaire le plus grand nombre de requêtes tout en respectant les contraintes d'échéance.
3. **Choix du temps d'envoi :** Pour chaque option de regroupement, l'algorithme sélectionne le premier intervalle de temps disponible pour envoyer les données. Cela signifie qu'il choisit le temps d'envoi le plus proche sans générer de conflits avec d'autres requêtes déjà planifiées. Si un conflit est détecté pour tous les temps d'envoi possibles, la requête est rejetée.

4. **Planification et retrait :** Après avoir sélectionné les temps d'envoi pour chaque option de regroupement, l'algorithme recherche la configuration qui permet de satisfaire le plus grand nombre de requêtes tout en respectant leurs contraintes d'échéance. Une configuration d'envoi est un ensemble de requêtes qui peuvent être traitées simultanément, que ce soit en mode local ou MEC, sans violation des contraintes d'échéance.

   Une fois la configuration optimale identifiée, l'algorithme planifie l'exécution de cette configuration dans les intervalles temporels choisis. Toutes les requêtes planifiées dans cette configuration sont retirées de la liste des requêtes en attente.

5. **Répétition :** Les étapes 1 à 4 sont répétées à chaque intervalle de temps jusqu'à ce qu'il ne reste plus de requêtes qui peuvent être planifiées.

L'utilisation de cette approche basée sur la date d'échéance est un moyen efficace de réduire les temps d'attente des requêtes et de répondre aux contraintes de délai. Le pseudo-code de l'AHG est inclus dans l'algorithme 1. La fonction `selectionnerRequete` choisit une requête en privilégiant celles dont la date d'échéance est la plus proche. La fonction `formerGroupe` utilise la requête sélectionnée et rassemble les requêtes qui peuvent être satisfaites en répondant à cette requête, tout en prenant en compte la décision de déchargement. La fonction `choisirGroupeOptimal` sélectionne le groupe d'envoi contenant le plus grand nombre de requêtes. L'intervalle d'envoi est déterminé par la fonction `creneauDisponible` parmi les intervalles disponibles, tout en respectant les contraintes de délai du groupe. Enfin, la fonction `retirerRequetesPlanifiees` retire les requêtes du groupe des requêtes non planifiées.

5.1.2 Complexité

Cette section fournit une analyse de la complexité algorithmique au pire cas de AHG. Le pire scénario pour notre algorithme se produit lorsque toutes les requêtes sont satisfaites en des envois individuels. Dans ce scénario, chaque requête ne peut être groupée avec aucune autre, ce qui signifie que AHG doit vérifier chaque requête individuellement pour sa disponibilité dans les intervalles de temps.

Algorithm 1 Algorithme heuristique glouton (AHG)

Entrée: Ensemble de requêtes $\mathcal{A}_{\text{non planifiées}}$, Ensemble d'intervalles de temps $\mathcal{T}_{\text{disponibles}}$

Sortie: Planification finale $\mathcal{P}$

tant que $\mathcal{A}_{\text{non planifiées}}$ n'est pas vide **faire**
  $\mathcal{R}_{\text{courante}} \leftarrow$ selectionnerRequete$(\mathcal{A}_{\text{non planifiées}})$
  $\mathcal{G}_{\text{entrées}} \leftarrow$ formerGroupe$(\mathcal{R}_{\text{courante}}, \mathcal{A}_{\text{non planifiées}}, \text{'entrées'})$
  $\mathcal{G}_{\text{sorties}} \leftarrow$ formerGroupe$(\mathcal{R}_{\text{courante}}, \mathcal{A}_{\text{non planifiées}}, \text{'sorties'})$
  $\mathcal{G}_{\text{optimal}} \leftarrow$ choisirGroupeOptimal$(\mathcal{G}_{\text{entrées}}, \mathcal{G}_{\text{sorties}}, \mathcal{T}_{\text{disponibles}})$
  si $\mathcal{G}_{\text{optimal}}$ n'est pas vide **alors**
    $\mathcal{T}_{\text{créneau}} \leftarrow$ créneauDisponible$(\mathcal{G}_{\text{optimal}}, \mathcal{T}_{\text{disponibles}})$
    si $\mathcal{T}_{\text{créneau}}$ n'est pas vide **alors**
      $\mathcal{P}[\mathcal{T}_{\text{créneau}}] \leftarrow \mathcal{G}_{\text{optimal}}$
      retirerRequetesPlanifiees$(\mathcal{G}_{\text{optimal}}, \mathcal{A}_{\text{non planifiées}})$
    fin si
  fin si
fin tant que
return $\mathcal{P}$

Dans la boucle principale while (ligne 1), AHG parcourt l'ensemble des requêtes non planifiées $\mathcal{A}_{\text{non planifiées}}$. Dans le pire cas, cette boucle s'exécute $|A|$ fois, où $|\mathcal{A}|$ représente le nombre total de requêtes non ordonnancées.

À l'intérieur de la boucle, AHG examine toutes les requêtes non planifiées pour former des groupes d'envoi dans l'un des intervalles de temps disponibles pour la requête sélectionnée. Par conséquent, la complexité de l'AHG au pire cas est de l'ordre de $O(|\mathcal{A}|(|\mathcal{A}|+|\mathcal{T}|) = O(|\mathcal{A}|^2+|\mathcal{A}|\cdot|\mathcal{T}|)$, où $|\mathcal{T}|$ représente le nombre d'intervalles de temps considérés.

Il est important de noter que cette analyse de complexité se base sur le scénario le plus contraignant, où toutes les requêtes sont des envois individuels et doivent être vérifiées pour la disponibilité des intervalles de temps. Dans des scénarios moins contraignants, où les requêtes peuvent être regroupées plus efficacement, l'algorithme s'exécute plus rapidement grâce à une complexité beaucoup plus réduite au meilleur cas.

5.2 Algorithme génétique pour ordonnancement des requêtes (AGOR)

Compte tenu de la complexité intrinsèque du problème examiné, cette section propose un algorithme génétique, cherchant à équilibrer la réduction de la complexité temporelle avec la quête d'une solution satisfaisante.

5.2.1 Fonctionnement général des AG

Les algorithmes génétiques sont des algorithmes d'optimisation inspirés de la sélection naturelle. Ils fonctionnent comme des algorithmes de recherche basés sur une population, en utilisant le concept de survie des plus aptes (en anglais survival of fittest). Les AG créent de nouvelles populations en appliquant de manière itérative des opérations génétiques aux individus de la population existante. Les éléments clés des AG comprennent la représentation des individus, la sélection, le croisement, la mutation et le calcul de la fonction de compatibilité (en anglais fitness function) (Katoch *et al.*, 2021).

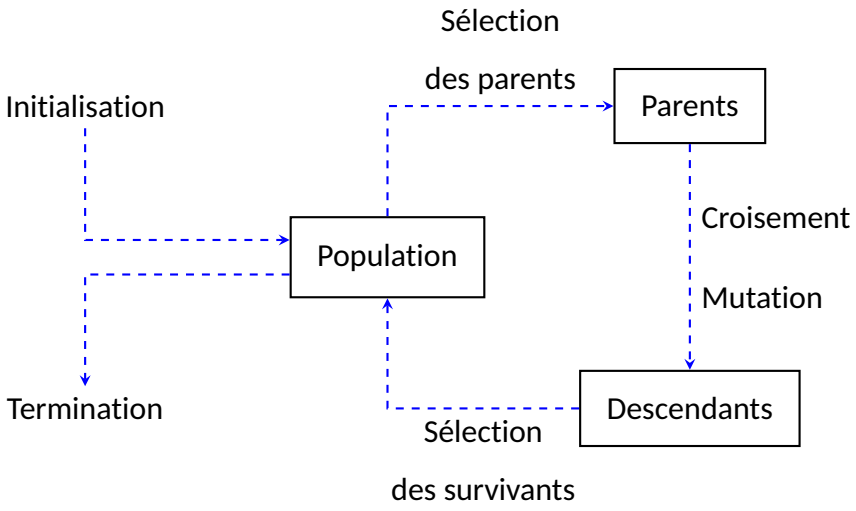

Figure 5.1 Fonctionnement général d'un algorithme génétique

Le processus général d'un AG se déroule généralement comme suit (voir la figure 5.1):

1. **Initialisation de la population:** Une population initiale, composée de plusieurs individus, est générée. Chaque individu représente une solution potentielle au problème.

2. **Évaluation des individus:** Chaque individu de la population est évalué en utilisant une fonction de compatibilité. Cette fonction mesure la qualité de la solution représentée par l'individu par rapport aux objectifs du problème.

3. **Sélection des parents:** Les individus de la population sont sélectionnés pour être les parents des générations futures. Les individus ayant une meilleure compatibilité ont une probabilité plus élevée d'être choisis, mais même les individus moins performants ont une chance de l'être aussi. Cette diversité permet une exploration plus diversifiée de l'espace des solutions.

4. **Croisement:** Les parents sélectionnés subissent l'opération de croisement. Le croisement combine les caractéristiques de deux parents pour créer de nouveaux individus (appelés aussi descendants).

5. **Mutation:** Les descendants peuvent subir à leur tour une opération de mutation. La mutation introduit de petites modifications aléatoires dans les descendants, permettant ainsi un plus grand degré d'exploration.

6. **Remplacement:** Les descendants sont utilisés pour remplacer une partie de la population courante. Cela garantit que la taille de la population reste constante et en même temps un remplacement opportuniste des individus les moins performants par les descendants les plus adaptés.

7. **Critères d'arrêt:** L'AG continue de générer de nouvelles générations jusqu'à ce qu'un critère d'arrêt soit satisfait, tel qu'un nombre maximal d'itérations, un degré de convergence suffisant, ou un temps d'exécution maximal.

En termes d'analyse mathématique, les AG adaptent dynamiquement leur processus de recherche en ajustant les probabilités de croisement et de mutation, convergeant finalement vers des solutions sous-optimales. Les AG ont la capacité de modifier les gènes encodés et d'évaluer plusieurs individus, ce qui peut potentiellement conduire à plusieurs solutions. Cette caractéristique inhérente confère aux AG une capacité de recherche globale supérieure aux méthodes de recherche locale.

5.2.2 Description de l'AGOR

Cette section fournit une présentation détaillée de notre algorithme génétique pour l'ordonnancement des requêtes (AGOR). Notre approche repose sur des ajustements spécifiques apportés aux composantes des algorithmes génétiques présentées dans la section précédente afin de s'adapter au contexte particulier du problème d'ordonnancement étudié.

5.2.2.1 Représentation d'un individu

Chaque individu est représenté par une matrice à deux lignes où les indices des colonnes représentent les indices des requêtes. La première ligne indique l'emplacement d'exécution de la tâche de la requête, avec les valeurs suivantes :

- 1 : si l'algorithme opte pour une exécution en local (c'est l'entrée i qui est transmise),
- 2 : s'il choisit une exécution au niveau du MEC (c'est la sortie $o_{k,i}$ qui est transmise), ou

Tableau 5.1 Représentation d'un individu

	Requêtes			
Indice	1	2	3	4
Exécution	1	2	0	1
Intervalle de temps	2	1	-1	3

- 0 : s'il choisit de rejeter la requête.

La deuxième ligne représente l'instant auquel la requête a été planifiée.

Le tableau 5.1 représente un exemple d'individu dans un contexte où quatre requêtes doivent être ordonnancées. Chaque colonne du tableau correspond à une requête numérotée de 1 à 4. Dans cet exemple, pour chaque requête, l'individu prend différentes décisions, comme suit:

1. Requête 1 : L'individu ordonnance l'envoi de l'entrée de la requête 1 à l'intervalle de temps 2.
2. Requête 2 : L'individu exécute la tâche associée à la requête 2 et ordonnance l'envoi de la sortie pour l'intervalle de temps 1.
3. Requête 3 : La requête 3 est rejetée, ce qui signifie qu'elle ne sera pas traitée ou planifiée dans cet exemple particulier.
4. Requête 4 : L'individu planifie la requête 4 pour une exécution locale. Cela implique que l'envoi de l'entrée de la requête 4 est ordonnancé pour l'intervalle de temps 3.

5.2.2.2 Compatibilité de l'individu

Cette sous-section présente deux fonctions de compatibilité différentes pour évaluer la qualité des individus dans AGOR. Les deux fonctions sont:

1. **Compatibilité avec échéance stricte :** (en anglais hard deadline, HD) elle mesure la capacité de la solution à respecter rigoureusement les contraintes de délai pour chaque requête. Dans cette approche, une requête n'est prise en compte que si elle est traitée et achevée avant sa date d'échéance stricte. La fonction de compatibilité associée à cette approche favorise donc les individus qui garantissent un respect strict des échéances des requêtes, ce qui est aligné avec la fonction objectif du problème d'ordonnancement définie dans la section 4.2.

$$\text{score}_{\text{HD}}(\text{individu}) = \sum_{\alpha \in \mathcal{A}} \begin{cases} 1 & \text{si } (\text{temps d'envoi}(\alpha) + T_\alpha) \leq (\tau_\alpha + \delta_\alpha) \\ 0 & \text{sinon} \end{cases} \tag{5.1}$$

2. **Compatibilité proportionnelle à l'échéance:** elle se base sur le concept de retard proportionnel par rapport aux dates d'échéance des requêtes. Elle pénalise l'individu en fonction de l'ampleur du retard en utilisant plusieurs seuils prédéfinis. Chaque seuil est associé à une pénalité spécifique, et le choix de la pénalité à appliquer dépend du seuil dépassé par le retard. Par exemple, si le retard dépasse une unité de l'intervalle de temps T_s, une pénalité spécifique est attribuée, différente de celle appliquée si le retard dépasse deux fois T_s, et ainsi de suite. Mathématiquement, la fonction suivante constitue un exemple de la compatibilité proportionnelle à l'échéance:

$$\text{score}_{\text{PD}}(\text{individu}) = \sum_{\alpha \in \mathcal{A}} \begin{cases} 2 & \text{si tard}_\alpha \geq 4 \\ 1 & \text{si } 0 < \text{tard}_\alpha < 4 \\ 0 & \text{sinon} \end{cases} \tag{5.2}$$

avec $\text{tard}_\alpha = \max(0, ((\text{temps envoi}(\alpha) + T_\alpha) - (\tau_\alpha + \delta_\alpha))$.

Cette fonction attribue des scores différents en fonction de l'importance du retard par rapport à la date d'échéance.

5.2.2.3 Initialisation

La manière dont nous générons cette population initiale peut avoir un impact sur la convergence et la diversité de notre algorithme génétique. L'initialisation de la population, décrite ci-dessous, est présentée dans l'algorithme 2 :

1. Pour chaque individu, nous générons aléatoirement un ordonnancement en prenant en compte les contraintes de temps et de ressources. Plus précisément, pour chaque requête de l'individu, nous générons aléatoirement l'emplacement où la requête sera exécutée. Ensuite, nous choisissons aléatoirement l'intervalle de temps d'ordonnancement t de manière à ce que t soit supérieur à τ_a et inférieur à $\delta_a - T_\alpha$. Cela garantit que le temps d'envoi est choisi de manière à respecter les contraintes de délai.
2. Avec une certaine probabilité, nous pouvons rejeter une requête dans le but d'augmenter la portion de l'espace de recherche à explorer.
3. Une fois la population initiale générée, la compatibilité de chaque individu est calculée en utilisant l'une de nos fonctions d'évaluation.

Cette initialisation aléatoire et diversifiée permet à notre algorithme génétique d'explorer un large espace de recherche dès le début, ce qui peut conduire à des solutions de haute qualité. Il est important de noter que les individus générés initialement peuvent ne pas respecter certaines contraintes du problème, telles que des conflits d'exécution entre certaines tâches. Dans le but de construire une population initiale composée d'individus représentant des solutions viables, une phase de correction est mise en œuvre après la génération initiale. Cette phase vise à ajuster les individus afin de résoudre les conflits et de respecter les contraintes.

Algorithm 2 Initialisation de la population

Entrée: Taille de la population N, Taille de l'individu L

Sortie: Population initiale P

1: **pour** $i \leftarrow 1$ to N **faire**
2: Créer un individu aléatoire C_i de taille L
3: Chaque gêne (colonne) de l'individu a l'emplacement et le temps d'envoi
4: Ajouter C_i à la population P
5: **fin pour**
6: $P \leftarrow$ Correction Population(P)
7: **return** P

5.2.3 Sélection des parents

La sélection des parents est une étape cruciale dans les algorithmes évolutifs, car elle détermine les individus choisis pour produire les descendants qui seront potentiellement inclus dans les générations futures. Pour garantir à la fois l'exploitation des solutions de haute qualité et l'exploration de l'espace des solutions, nous utilisons deux méthodes de sélection que nous comparerons dans le chapitre suivant, soient la sélection par roulette (en anglais roulette wheel selection) et la sélection des élites (Katoch *et al.*, 2021).

1. **Sélection par roulette (SR) :** Cette méthode de sélection se fonde sur une logique probabiliste. Chaque individu a une probabilité de sélection proportionnelle à son score de compatibilité. La règle est simple, plus un individu affiche un score de compatibilité élevé, plus ses chances d'être choisi en tant que parent sont grandes. Cette approche présente l'avantage de favoriser la préservation des individus de haute qualité. Cependant, elle conserve également une composante d'exploration en permettant aux individus moins performants d'avoir une opportunité de reproduction. Ceci est essentiel pour éviter de restreindre la diversité génétique, car de nouvelles solutions potentielles pourraient émerger.

2. **Sélection des élites (SE) :** En plus de la SR, nous considérons la sélection des élites. Cette méthode consiste à sélectionner directement les meilleurs individus de la population actuelle en fonction de leurs scores de compatibilité. Les individus d'élite sont ceux qui présentent

les performances les plus élevées. Cette approche garantit que les solutions de haute qualité sont préservées d'une génération à l'autre et contribuent à l'exploitation des zones de solution prometteuses. L'algorithme 3 présente le pseudo-code décrivant le processus de sélection des élites.

Algorithm 3 Sélection : Sélection des élites

Entrée: Population P, Taille de la nouvelle population N, Fonction de fitness f

Sortie: Nouveaux individus sélectionnés P'

1: Trier la population P en fonction de f (du meilleur au moins bon)

2: Sélectionner les N premiers individus de la population triée pour P'

3: **return** P'

5.2.4 Croisement

Les opérations de croisement sont des mécanismes utilisés pour créer de nouveaux individus en combinant les informations génétiques de deux parents ou plus. Les opérateurs de croisements considérés et évalués dans le cadre de cette étude sont les suivants (Soon *et al.*, 2013):

1. Dans le croisement à un point (CR1), un point de croisement aléatoire est choisi. Les informations génétiques des deux parents au-delà de ce point sont échangées entre eux. La figure 5.2 illustre un exemple de croisement CR1.

2. En cas de croisement à deux points (CR2) et de croisement à k points, deux points de croisement ou plus sont sélectionnés aléatoirement, et les informations génétiques des parents sont échangées en fonction des segments créés.

3. Dans le cas du croisement uniforme (CRU), nous décidons aléatoirement si nous devons échanger le gène avec la même position d'un autre individu. La figure 5.3 représente l'échange d'individus lors de l'opération de croisement uniforme.

Le pseudo-code de l'opération de croisement en un point est présenté dans l'algorithme 4.

Algorithm 4 Croisement : croisement en un point

Entrée: Deux parents P_1 et P_2, Taux de croisement p_c

Sortie: Enfant $C1, C2$

1: Choisir un point de croisement aléatoire k entre 1 et la taille du chromosome

2: Diviser P_1 et P_2 en deux parties au point k pour obtenir A_1 et B_1, A_2 et B_2

3: Créer l'enfant $C1$ en combinant A_1 avec B_2

4: Créer l'enfant $C2$ en combinant A_2 avec B_1

5: **return** $C1, C2$

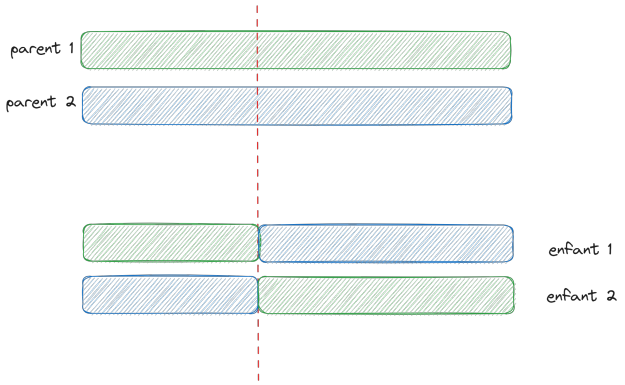

Figure 5.2 Illustration du croisement en un point (CR1)

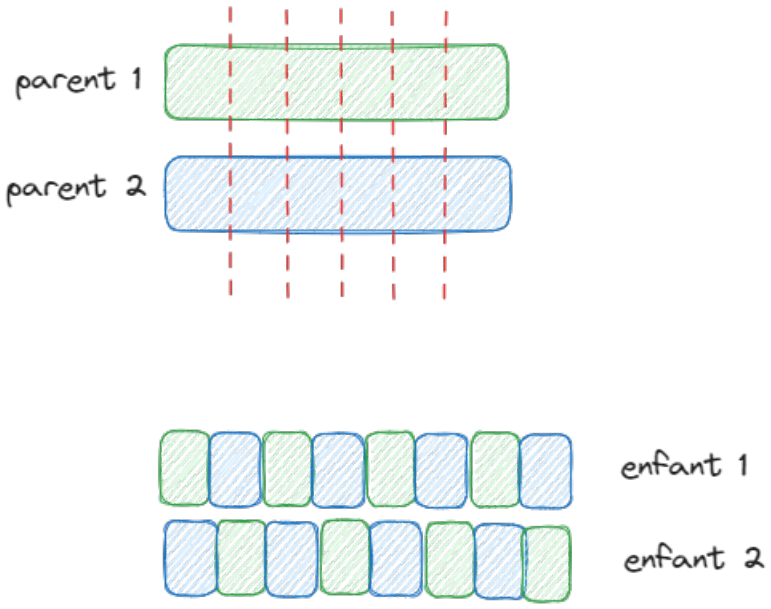

Figure 5.3 Illustration du Croisement uniforme (CRU)

5.2.5 Mutation

Une fois que les enfants ont été créés par croisement, l'opération de mutation est appliquée. La mutation permet d'introduire de petites modifications aléatoires dans les enfants. L'algorithme 5 décrite la mutation utilisée dans AGOR. Cela ajoute une exploration aléatoire dans l'espace des solutions. Les types de mutations dans AGOR concernent le changement de l'emplacement d'exécution d'une requête ou l'intervalle de temps d'exécution d'une requête dans un individu . La probabilité de mutation détermine la fréquence à laquelle ces modifications sont appliquées.

Algorithm 5 Mutation

Entrée: Individu C, Taux de mutation p_m

Sortie: Individu muté C'

1: **pour** Chaque gêne g dans C **faire**
2: **si** Aléatoire $< p_m$ **alors**
3: récupérer une valeur aléatoire entre $a_{n,k}$ et $a_{n,k} + d_{n,a}$
4: choisir aléatoirement ou exécuter la requête.
5: modifier la valeur de g dans C'
6: **fin si**
7: **fin pour**
8: **return** C'

5.2.6 Correction des individus

Avant de passer à la génération suivante, les individus peuvent subir une correction pour garantir que les solutions respectent les contraintes du problème. Par exemple, si deux requêtes du même groupe d'envoi ne peuvent pas être exécutées en même temps, la correction ajuste les emplacements ou les temps d'exécution pour résoudre ce conflit en suivant les règles établies dans le tableau 5.2 en utilisant les notations suivantes:

- $C1$: Tâche de la première requête.
- $C2$: Tache de la deuxième requête.

- $I1$: Entrée de la première requête.
- $I2$: Entrée de la deuxième requête.
- $L1$: Emplacement de la première requête (*peut être LOCAL ou SERVER*).
- $L2$: Emplacement de la deuxième requête (*peut être LOCAL ou SERVER*).

Possibilité	Action à prendre
$C1 = C2, I1 = I2, L1 \neq L2$	Ajuster $L1$ ou $L2$ pour avoir $L1 = L2$ (*LOCAL ou SERVER*)
$C1 = C2, I1 \neq I2, L1 \neq L2$	Rejeter aléatoirement une des deux requêtes
$C1 \neq C2, I1 \neq I2, L1 \neq L2$	Rejeter aléatoirement une des deux requêtes
$C1 = C2, I1 \neq I2, L1 = L2$	Rejeter aléatoirement une des deux requêtes
$C1 \neq C2, I1 \neq I2, L1 = L2$	Rejeter aléatoirement une des deux requêtes
$C1 \neq C2, I1 = I2, L1 = L2$	Si $L1 = L2 = LOCAL$, aucune action requise. Sinon, ajuster $L1$, $L2$ à *LOCAL*
$C1 \neq C2, I1 = I2, L1 \neq L2$	Ajuster $L1$, $L2$ à *LOCAL*
$C1 \neq C2, I1 \neq I2, L1 = L2$	Rejeter aléatoirement une des deux requêtes
$C1 = C2, I1 = I2, L1 = L2$	Aucune action requise, faisable

Tableau 5.2 Actions à entreprendre en cas de conflits

L'algorithme 6 présente le pseudo-code décrivant le processus de la correction de la population.

Algorithm 6 Correction Population

Entrée: Population P

1: **pour** Individu $\in P$ **faire**
2: **Appliquer le tableau 5.2**
3: **fin pour**
4: **return** P

5.2.7 Formation de la nouvelle population

Les parents sélectionnés, ainsi que les enfants issus du croisement et de la mutation, forment la nouvelle population pour la génération suivante. Cette population est ensuite évaluée à nouveau, et le processus de sélection, croisement, mutation et correction est répété pour chaque nouvelle génération.

5.2.8 Terminaison

Au fil des générations, l'algorithme fait évolué la population vers des solutions de plus en plus optimales. Les meilleurs individus, ceux avec les scores de compatibilité les plus élevés, représentent les ordonnancements les plus performants en termes d'utilisation des ressources et de respect des contraintes de temps. Nous fixons le nombre de générations à produire dans notre travail.

5.3 Conclusion

En conclusion, ce chapitre a exposé deux solutions distinctes pour aborder le problème de prise de décision de déchargement et d' ordonnancement des requêtes dans un environnement de réseau MEC. L'heuristique gloutonne offre une approche rapide et relativement simple pour générer des solutions acceptables, bien qu'elle puisse ne pas toujours aboutir à une optimisation globale. D'un autre côté, l'algorithme génétique, grâce à son processus évolutif, permet d'explorer un espace de recherche plus large, visant potentiellement des solutions plus performantes.

Chacune de ces approches présente des avantages et des limitations, et le choix entre les deux dépendra des spécificités du problème à résoudre et des contraintes temporelles. L'analyse de la complexité fournit des idées sur les performances théoriques de ces algorithmes dans différents contextes.

Dans le chapitre suivant, nous évaluerons empiriquement ces solutions à l'aide de simulations pour mieux comprendre leur comportement dans des scénarios réalistes.

CHAPITRE 6
ÉVALUATION DES PERFORMANCES ET RÉSULTATS

Ce chapitre présente une évaluation des performances des deux algorithmes proposés, soient l'heuristique gloutonne AHG et l'algorithme génétique AGOR, en moyen de simulations. Il commence par une présentation globale des paramètres généraux du système simulé ainsi qu'une description des choix des éléments réglables propres à l'algorithme génétique. Le chapitre expose par la suite une comparaison des performances des deux algorithmes.

6.1 Paramètres de simulation

6.1.1 Paramètres généraux

Les simulations effectuées dans le cadre de cette évaluation de performances considèrent un réseau composé d'une BS et d'un nombre variable d'UE. Les positions des UE sont générées aléatoirement selon une distribution uniforme dans une zone géographique de rayon, $R = 500$ mètres. La BS est positionnée au centre de cette zone, avec un accès exclusif aux ressources de calcul d'un serveur MEC. La fréquence du CPU de ce dernier est $F = 5$ GHz, tandis que les fréquences des CPU des UE sont considérées similaires et égales toutes à $f = 100$ MHz. Le réseau utilise une bande de fréquence de largeur $B = 20$ MHz. La durée d'un intervalle de temps définit dans le chapitre 4 est 1 s.

Chaque simulation s'étend sur un minimum de 40 intervalles de temps. Au début de chaque intervalle, chaque UE génère une requête avec une probabilité de 0,05. Rappelons qu'une requête α est définie par le tuple $(n, \tau, \kappa, \iota, \delta)$, avec

- n désigne l'UE ayant généré la requête.
- τ représentant l'intervalle de temps d'arrivée de la requête,
- κ désignant la tâche, choisie parmi un ensemble de cinq éléments,

- ι désigne les données d'entrées avec une taille comprise entre 1 kB et 1 MB.
- δ est l' échéance générée aléatoirement dans l'intervalle [3, 5], sauf indication contraire.

Enfin, la taille de la sortie d'une tâche, est générée aléatoirement entre 1,5 et 2 fois la taille de l'entrée. Chaque tâche a une exigence de calcul entre 100 MHz et 1 GHz. Le tableau 6.1 regroupe les paramètres généraux utilisés.

Tableau 6.1 Notation et valeurs des paramètres généraux

Notation	**Valeur**	Description
R	500m	Rayon de la couverture de la BS
N	[10, 100]	Nombre variable de UE
K	5	Nombre de tâches
$\|\mathcal{I}\|$	5	Nombre de données d'entrée
I_i	[1kb, 1Mb]	Taille de l'entrée i
β_k	[1,5, 2]	Rapport tailles de sortie/entrée
ψ_k	[100MHz, 1GHz]	Ressources de calcul requises pour la tâche k
F	5GHz	Fréquence du CPU du serveur MEC
f	100MHz	Fréquence du CPU d'un UE
B	20 MHz	Largeur de la bande passante
T_s	1s	La durée d'un intervalle de temps

6.1.2 Paramétrage de AGOR

Cette section détaille les choix effectués concernant les opérateurs et paramètres de AGOR, incluant le croisement, la sélection, la fonction de compatibilité, le nombre de générations, et la taille de la population. Les taux de mutation et de croisement demeurent constants et sont fixés à 0,1 et 0,7, respectivement.

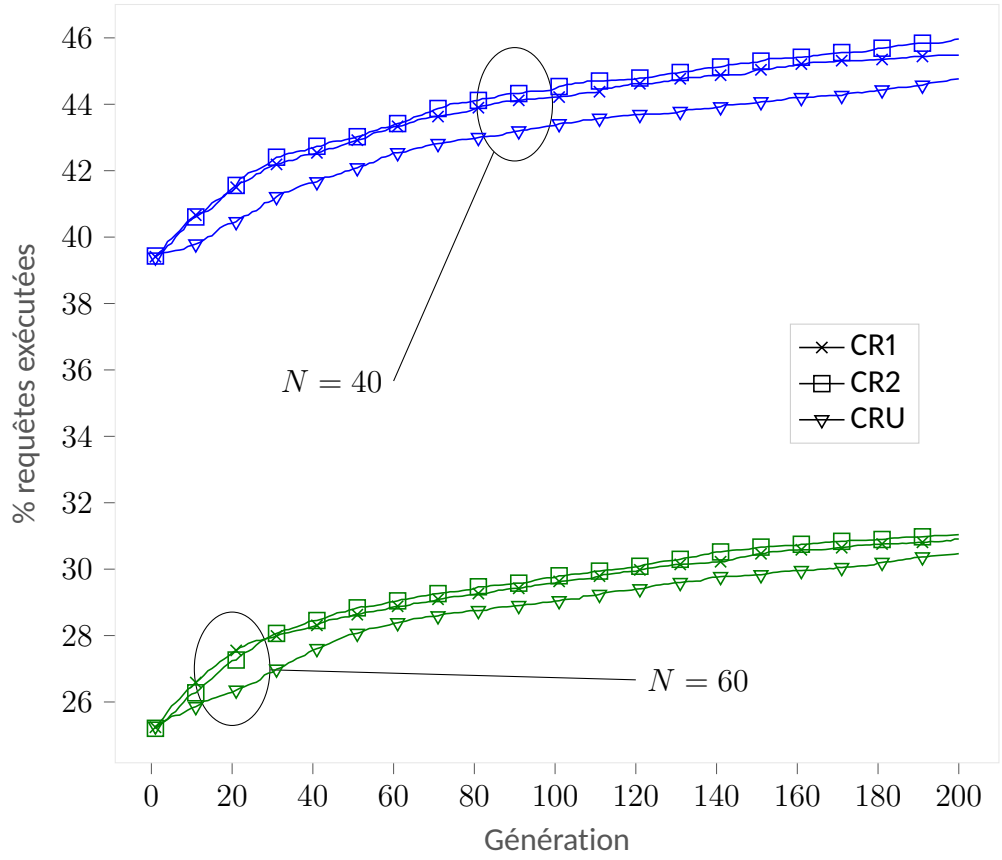

Figure 6.1 Pourcentage des requêtes exécutées en fonction du nombre de générations pour différents opérateurs de croisement.

La figure 6.1 montre l'évolution du pourcentage des requêtes exécutées par rapport à celles générées par les UE en fonction de l'indice de la génération. La fonction de compatibilité est maintenue à HD , tandis que l'opérateur de sélection reste constant avec SE. On compare trois opérateurs de croisement qui sont CR1 (à un point), CR2 (à deux points) et CRU (uniforme), pour deux valeurs différentes du nombre d'UE N. La figure montre une supériorité claire de l'opérateur CR2, et ce pour les deux valeurs de N considérés.

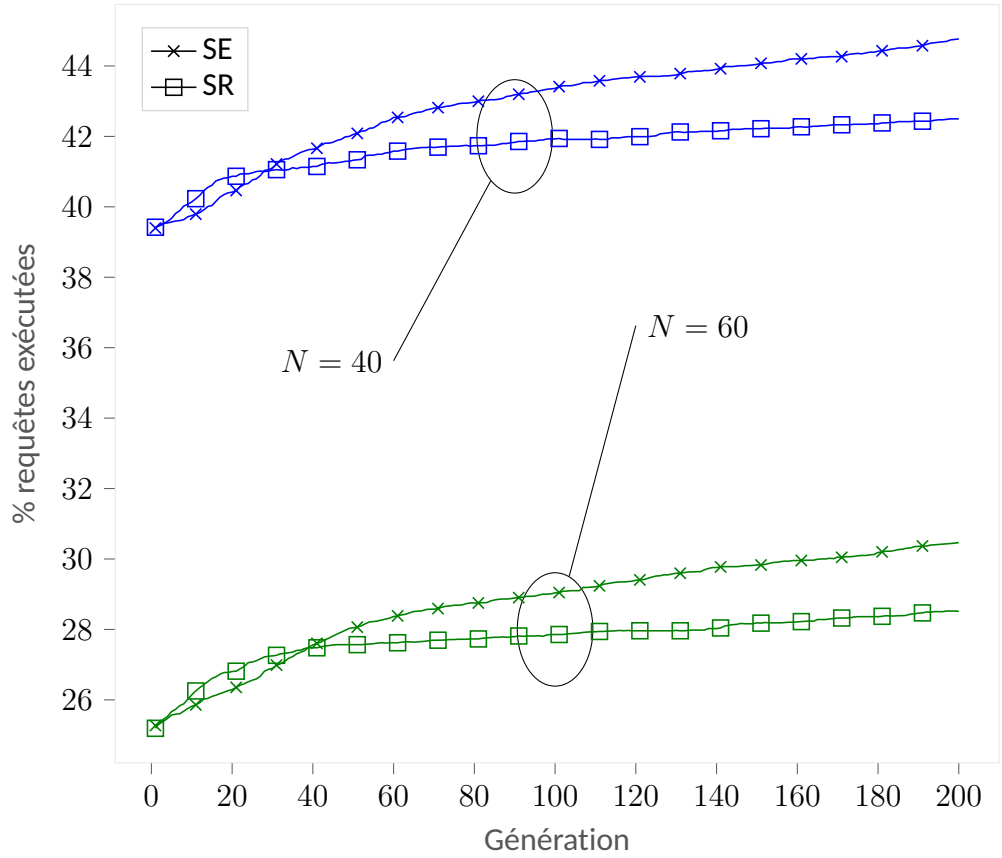

Figure 6.2 Pourcentage des requêtes exécutées en fonction du nombre de générations pour différents opérateurs de sélection.

D'une façon similaire, la figure 6.2 présente le pourcentage des requêtes exécutées en considérant les deux méthodes de sélection présentées dans le chapitre précédent, soit SR (par roulette) et SE (par élitisme). La compatibilité HD et le croisement CRU sont utilisés. On remarque que malgré une amélioration rapide au début de l'exécution avec SR, les performances de cette méthode peinent à s'améliorer et se voient saturées dès l'atteinte de quelques dizaines de générations. Le comportement de la sélection par élitisme est différent et permet d'atteindre des performances largement supérieures, surtout si l'on peut se permettre un nombre de générations plus élevé.

Cette constatation suggère une corrélation potentielle avec le compromis exploration/exploitation. On peut conjecturer que SR privilégie l'exploration, bénéfique initialement, mais limitant les possibilités d'amélioration à long terme en n'exploitant pas suffisamment le voisinage des solutions

prometteuses. À l'inverse, SE semble adoptée une approche plus équilibrée, ce qui pourrait expliquer ses performances supérieures.

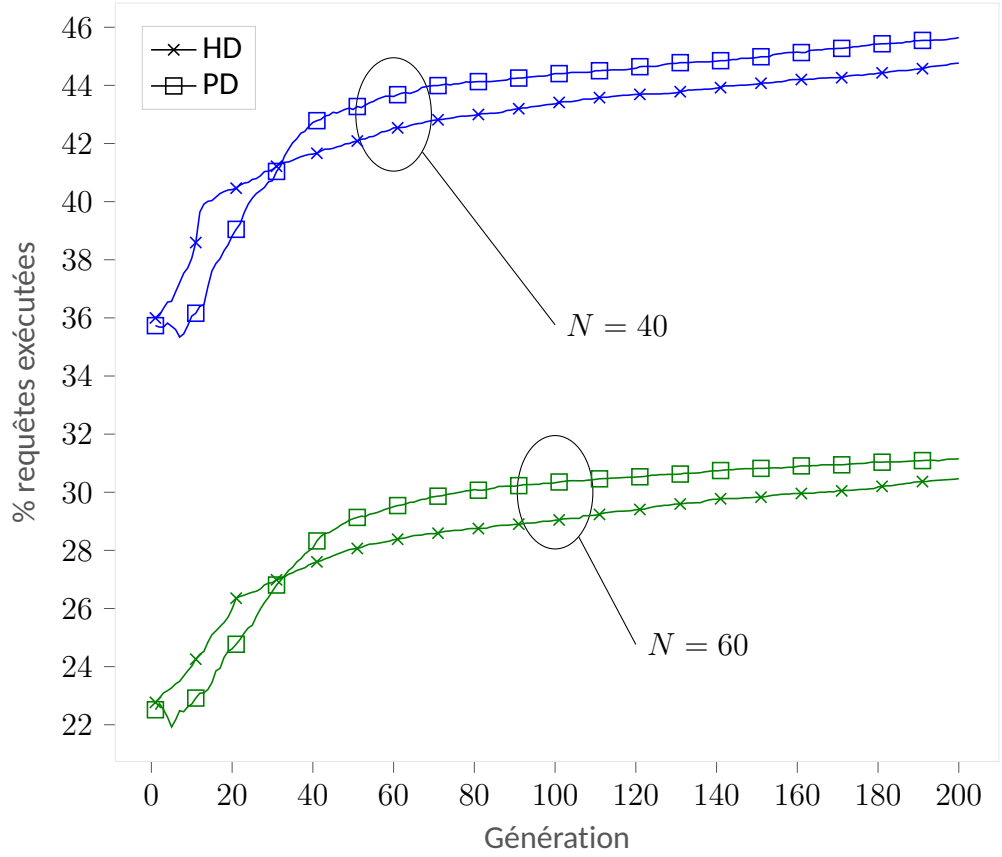

Figure 6.3 Pourcentage des requêtes exécutées en fonction du nombre de générations pour différentes fonctions de compatibilité

Afin de choisir la fonction de compatibilité la mieux adaptée au problème étudié, nous présentons dans la figure 6.3 les performances des deux fonctions de compatibilité, à échéance proportionnelle (PD) et fixe (HD). Ces simulations sont réalisées en fixant tous les autres paramètres et opérations, tels que la méthode de croisement à CRU, la sélection à SE, et la taille de la population à 100, pour deux valeurs de N. La compatibilité à échéance proportionnelle s'avère être la plus performante, surtout lorsque l'algorithme a la possibilité de s'exécuter plus longtemps, c.-à-d. un nombre de générations plus élevé. Ces performances sont dues à la nature de la compatibilité proportionnelle, qui permet d'inclure dans l'exploration de l'espace de recherche des individus représentant des solutions non faisables avec une pénalité. Cette propriété, absente dans le cas

de la compatibilité fixe, permet de s'éloigner plus efficacement des optima locaux. La figure 6.4

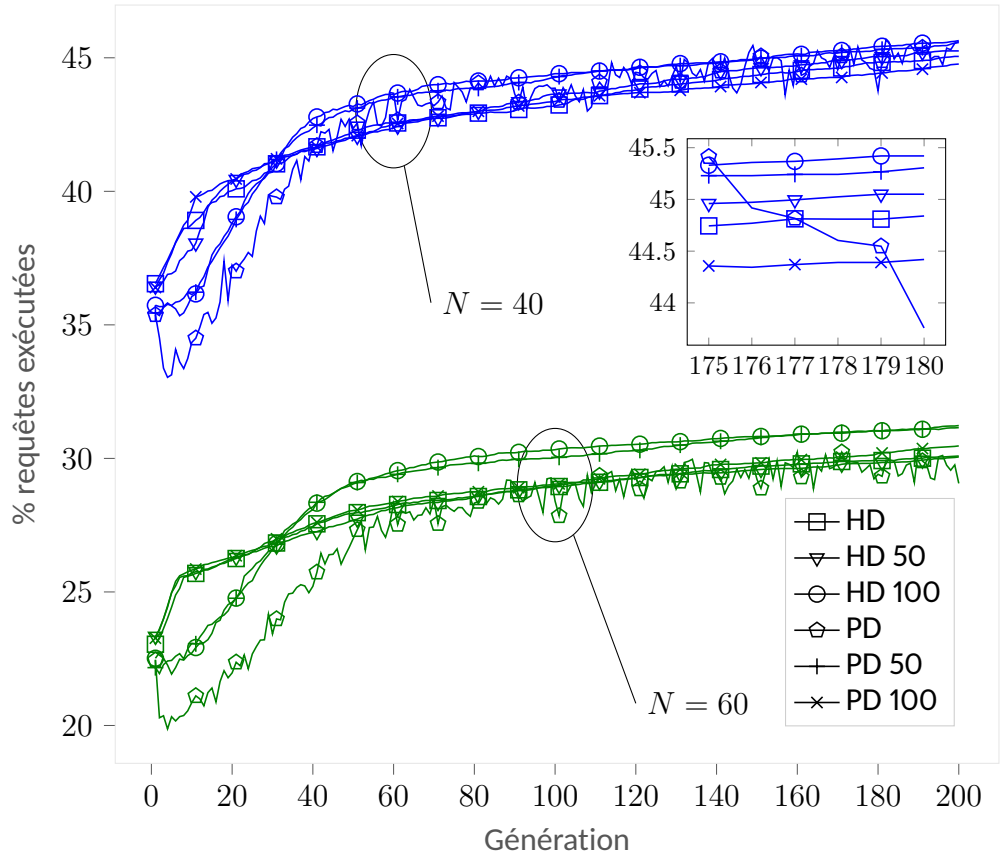

Figure 6.4 Pourcentage des requêtes exécutées en fonction du nombre de générations pour différentes niveaux de correction.

présente le pourcentage de requêtes exécutées par AGOR en fonction du nombre de générations lorsque différents niveaux de correction sont appliqués aux individus de chaque génération. Ces niveaux sont: sans correction (HD ou PD), correction de la moitié des individus (HD 50 ou PD 50) et correction de tous les individus de la population (HD 100 ou PD 100). Ces niveaux de correction sont appliqués aux deux fonctions de compatibilité pour deux valeurs de N avec les opérations génétiques SE et CRU. La compatibilité proportionnelle sans correction est la plus performante, même si elle est rattrapée par les deux autres types de correction, à savoir une correction de la moitié des individus de chaque génération (PD 50) et la correction de tous les individus (PD 100). La figure 6.5 présente l'évolution du pourcentage de requêtes exécutées en fonction de la taille de la population, en utilisant l'opérateur de croisement CR2, la sélection SE et la fonction de compatibilité PD pour différentes valeurs de N (nombre de UE) : 20, 40, 60. L'observation principale

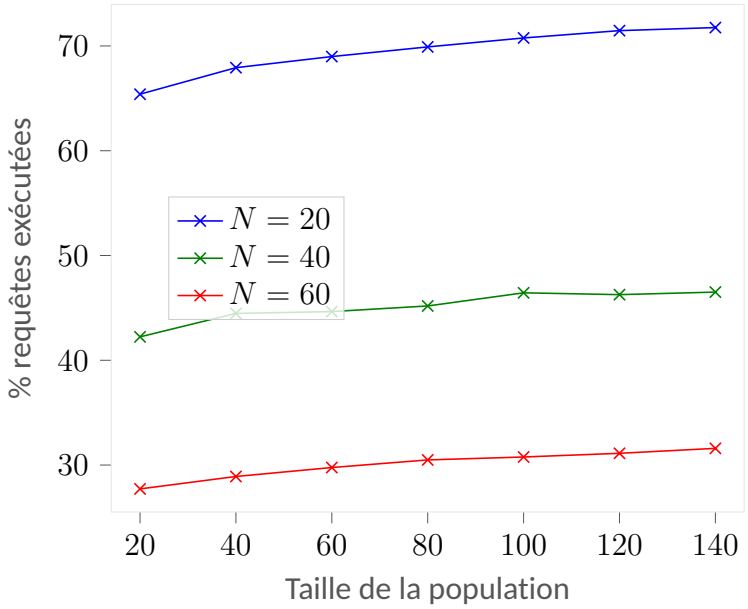

Figure 6.5 Pourcentage des requêtes exécutées en fonction de la taille de la population pour différentes Nombre de UE

réside dans la progression graduelle des performances en relation avec l'accroissement de la taille de la population, atteignant un plateau autour de 100 pour chaque valeur de N. Cette stabilisation suggère que l'augmentation au-delà de 100 n'apporte pas d'amélioration significative des performances. Par conséquent, nous décidons de maintenir une taille de population constante à 100 pour la suite de nos expériences.

Tableau 6.2 Paramètres et opérations sélectionnées pour AGOR

Paramètres et opérations	**Choisis**
Méthode de sélection	SE
Méthode de croisement	CR2
Nombre de générations	150
Taille de la population	100
Compatibilité	Échéance proportionnele
Application de la correction	sans correction

En se basant sur les résultats des figures précédentes, les simulations de la section suivante utiliseront les choix présentés dans le tableau 6.2.

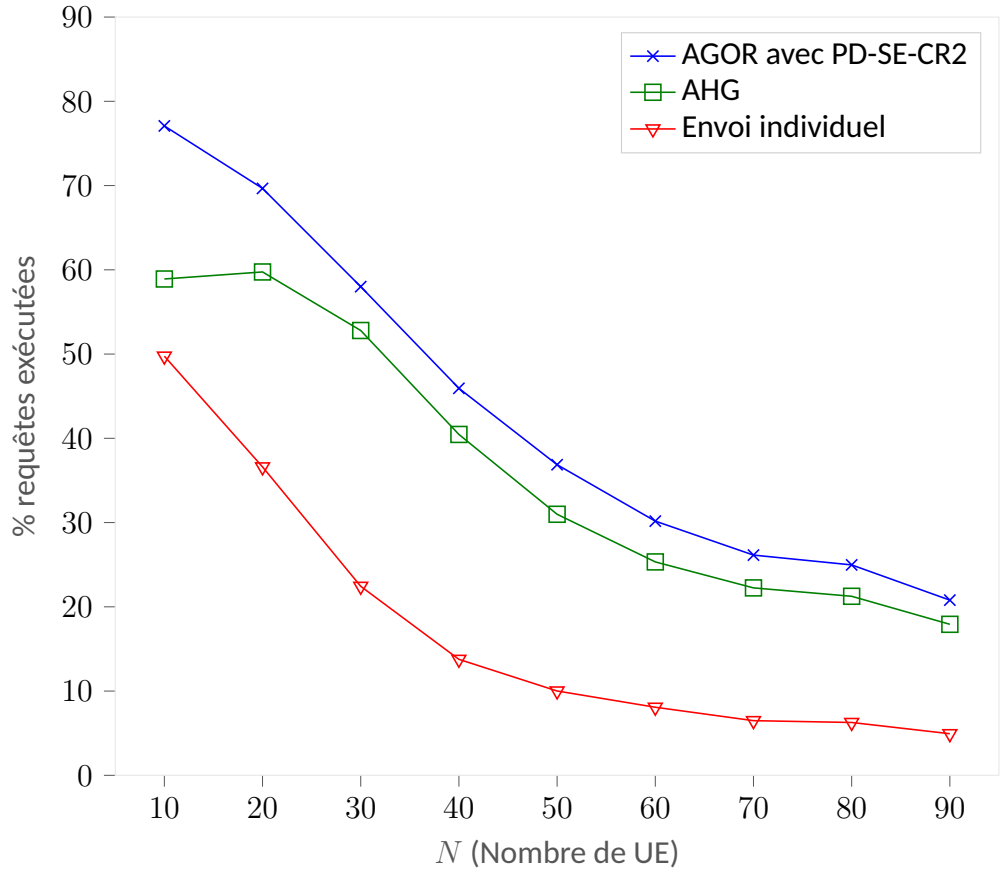

Figure 6.6 Comparaison du pourcentage des requêtes exécutées en variant le nombre de UE.

La figure 6.6 présente le pourcentage des requêtes exécutées en fonction du nombre d'utilisateurs. Nous comparons AHG et AGOR à un algorithme qui effectue exclusivement des envois individuels. Ce dernier priorise les requêtes dont l'échéance est la plus proche et la sert par un envoi individuel, soit de l'entrée si l'exécution locale est possible dans le délai imparti, sinon du traitement au niveau du MEC, suivi de l'envoi de la sortie. Il est à noter que, vu que la probabilité de générer une requête est inchangée dans le temps, le fait d'avoir un réseau plus dense implique la génération d'un nombre plus important de requêtes. Par conséquent, il est normal de constater que l'augmentation du nombre de UE diminue nécessairement le taux d'exécution à cause de la quantité de ressources qui reste inchangée face à une augmentation de la demande. Toutefois, malgré cette tendance de diminution partagée par les trois algorithmes, on remarque que les performances de AGOR surpassent clairement celles de AHG et de l'envoi individuel. Cette supéri-

orité demeure constante, indépendamment de la densité du réseau. La supériorité s'explique par le nombre de multidiffusions effectuées par AGOR par rapport à l'heuristique AHG.

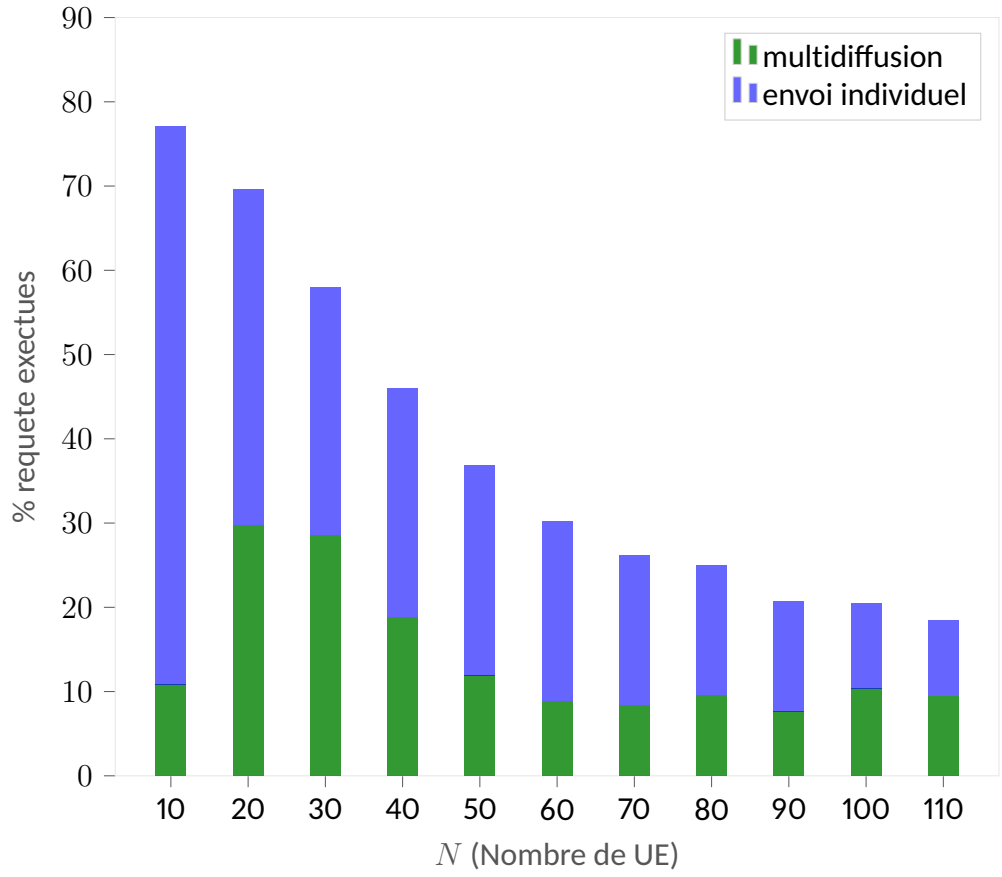

Figure 6.7 Répartition des requêtes en envoi individuel et multidiffusion en fonction du nombre d'utilisateurs : AGOR.

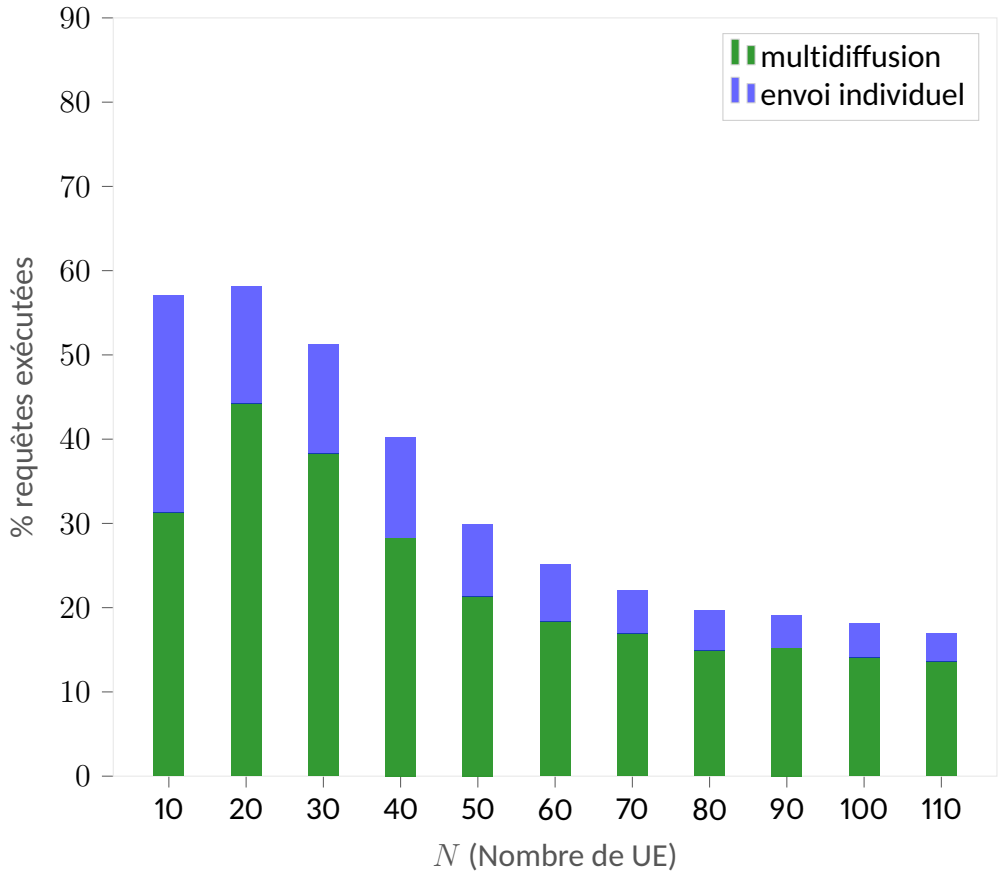

Figure 6.8 Répartition des requêtes en envoi individuel et multidiffusion en fonction du nombre d'utilisateurs : Heuristique.

Les figures 6.7 et 6.8 présente le pourcentage de requêtes exécutées en fonction du nombre d'utilisateurs, ainsi que la proportion attribuée à la multidiffusion par rapport à l'envoi individuel de AGOR et de AHG. AGOR performe mieux, non seulement à cause du nombre d'envois en multidiffusion, mais aussi pour les envois individuels. Les envois en multidiffusion de AGOR sont plus intelligents et sont constitués de groupes de taille plus importante que ceux de AHG. L'AGOR exécute davantage de requêtes que AHG, même si ce dernier effectue plus de multidiffusion dans certains cas. AGOR parvient à trouver un équilibre, tandis que AHG se concentre principalement sur la réalisation de multidiffusions.

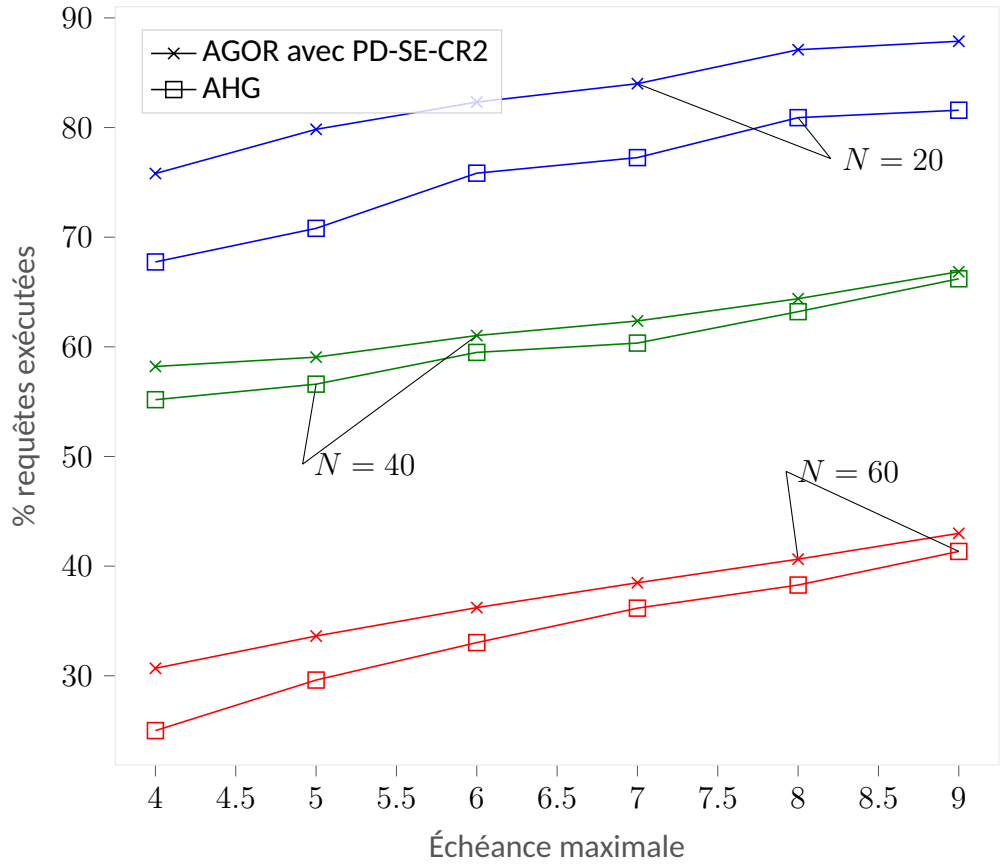

Figure 6.9 Pourcentage d'exécution des requêtes par rapport à l'échéance maximale.

La figure 6.9 présente les performances en termes de pourcentage des requêtes exécutées en faisant varier l'échéance maximale des requêtes pour deux valeurs de N. La performance de tous les algorithmes augmente avec l'échéance maximale. Lorsque les requêtes ont une échéance plus grande, le système dispose de plus de temps pour ordonnancer les requêtes. Davantage de requêtes peuvent ainsi être exécutées, et davantage de multidiffusions ont lieu. Si l'échéance continue d'augmenter, toutes les requêtes pourront être exécutées avec des multidiffusions.

6.3 Conclusion

Ce chapitre a été dédié à l'évaluation des performances des deux algorithmes proposés, à savoir AHG et AGOR, à travers des simulations approfondies. L'analyse a débuté en détaillant les paramètres généraux du système simulé. Nous avons examiné de près le paramétrage spécifique de l'algorithme

AGOR, discutant des opérateurs de croisement, de sélection, de la fonction de compatibilité, ainsi que du nombre de générations et de la taille de la population.

Les résultats des simulations ont été présentés à travers diverses figures, mettant en lumière la supériorité de certains choix opérationnels, tels que l'opérateur de croisement CR2, la sélection SE, et la fonction de compatibilité PD. Ces choix ont démontré des avantages significatifs en termes de pourcentage des requêtes exécutées.

L'analyse comparative des performances d'AGOR par rapport à AHG et à un algorithme de référence basé sur des envois individuels a révélé une nette supériorité des algorithmes proposés. Elle est principalement attribuée à une utilisation plus intelligente de la multidiffusion et à une gestion plus efficace des envois individuels. Enfin, nous avons exploré l'impact de l'échéance maximale des requêtes sur les performances, soulignant une amélioration globale avec l'augmentation de cette échéance.

CHAPITRE 7
CONCLUSION

Ce mémoire a été le fruit d'une exploration approfondie des défis et des opportunités inhérents à l'intégration de la multidiffusion dans un réseau MEC. Notre démarche a été structurée de manière à couvrir divers aspects, de l'examen de l'état de la recherche existante à la proposition de solutions visant à améliorer les performances des réseaux de communication, en particulier pour des applications exigeantes en calcul.

Nous avons formulé un problème complexe axé sur la prise de décision de déchargement de tâches au niveau du MEC pour plusieurs utilisateurs, conscient de la multidiffusion, dans le but de maximiser l'efficacité des réponses aux requêtes. Pour résoudre ce défi, nous avons introduit une solution heuristique gloutonne d'ordonnancement de tâches, qui, malgré sa moindre complexité, offre des performances encourageantes.

L'élément clé de notre contribution réside dans la proposition d'une solution d'ordonnancement de tâches basée sur un algorithme génétique. Nous avons développé plusieurs composants fondamentaux, notamment des fonctions de compatibilité, de croisement, de sélection, ainsi que des mécanismes de modification des individus au sein de la population. Les simulations détaillées ont permis de définir les combinaisons optimales d'opérateurs et de paramètres pour notre algorithme génétique, mettant en évidence son efficacité particulière à notre problème.

Les résultats obtenus ont été comparés avec notre solution heuristique gloutonne et avec des scénarios où la BS transmet exclusivement en envoi individuel. Ces comparaisons ont révélé la supériorité de notre approche génétique, soulignant l'importance d'un équilibre judicieux entre les envois en multidiffusion et individuels pour obtenir des performances optimales.

Plusieurs perspectives de recherche demeurent ouvertes. La gestion dynamique de la mise en cache des sorties de tâches, adaptée aux besoins changeants des utilisateurs, constitue un aspect essentiel pour réduire le temps de traitement, en tenant compte, par exemple, de la popularité des

tâches. De même, la coordination efficace entre plusieurs serveurs associés à différentes stations de base constitue également un défi de taille. Enfin, l'intégration de la mobilité des utilisateurs, pour refléter de manière plus réaliste les scénarios d'utilisation, est également un axe de recherche prometteur.

RÉFÉRENCES

Abbas, A., Raza, A., Aadil, F. et Maqsood, M. (2021). Meta-heuristic-based offloading task optimization in mobile edge computing. *International Journal of Distributed Sensor Networks*, *17*(6), 15501477211023021.

Afolabi, R. O., Dadlani, A. et Kim, K. (2012). Multicast scheduling and resource allocation algorithms for ofdma-based systems: A survey. *IEEE Communications Surveys & Tutorials*, *15*(1), 240–254.

Al-Habob, A. A., Dobre, O. A., Armada, A. G. et Muhaidat, S. (2020). Task scheduling for mobile edge computing using genetic algorithm and conflict graphs. *IEEE Transactions on Vehicular Technology*, *69*(8), 8805–8819.

Araniti, G., Condoluci, M., Scopelliti, P., Molinaro, A. et Iera, A. (2017). Multicasting over emerging 5g networks: Challenges and perspectives. *Ieee network*, *31*(2), 80–89.

Cao, K., Liu, Y., Meng, G. et Sun, Q. (2020). An overview on edge computing research. *IEEE access*, *8*, 85714–85728.

Chandra, R., Saravanaselvi, P. et Latha, P. (2013). Multicasting of video stream over wimax network. Dans *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 1–5. IEEE.

Chukhno, N., Chukhno, O., Moltchanov, D., Pizzi, S., Gaydamaka, A., Samuylov, A., Molinaro, A., Koucheryavy, Y., Iera, A. et Araniti, G. (2023). Models, methods, and solutions for multicasting in 5g/6g mmwave and sub-thz systems. *IEEE Communications Surveys & Tutorials*.

Du, C., Chen, Y., Li, Z. et Rudolph, G. (2019). Joint optimization of offloading and communication resources in mobile edge computing. Dans *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2729–2734. `http://dx.doi.org/10.1109/SSCI44817.2019.9003099`

Guo, C., Cui, Y. et Liu, Z. (2018). Optimal multicast of tiled 360 vr video. *IEEE Wireless Communications Letters*, *8*(1), 145–148.

Hao, H., Xu, C., Yang, S., Zhong, L. et Muntean, G.-M. (2021). Multicast-aware optimization for resource allocation with edge computing and caching. *Journal of Network and Computer Applications*, *193*, 103195.

Hao, H., Xu, C., Zhong, L. et Wu, D. O. (2020). Stochastic cooperative optimization for multicast scheduling in heterogeneous and green 5g networks. *IEEE Transactions on Green Communications and Networking*, *4*(3), 903–913.

He, W., Wu, S. et Sun, J. (2021). An effective metaheuristic for partial offloading and resource

allocation in multi-device mobile edge computing. Dans *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, 1419–1426. IEEE.

Huang, C., Zhang, J., Poor, H. V. et Cui, S. (2016). Delay-energy tradeoff in multicast scheduling for green cellular systems. *IEEE Journal on Selected Areas in Communications*, *34*(5), 1235–1249. `http://dx.doi.org/10.1109/JSAC.2016.2551559`

Huang, X., Zhao, Z. et Zhang, H. (2017). Cooperate caching with multicast for mobile edge computing in 5g networks. Dans *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, 1–6. IEEE.

Katoch, S., Chauhan, S. S. et Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, *80*, 8091–8126.

Kong, X., Wu, Y., Wang, H. et Xia, F. (2022). Edge computing for internet of everything: A survey. *IEEE Internet of Things Journal*, *9*(23), 23472–23485.

Lei, H., Yang, H., Park, K.-H., Ansari, I. S., Jiang, J. et Alouini, M.-S. (2023). Joint trajectory design and user scheduling for secure aerial underlay iot systems. *IEEE Internet of Things Journal*.

Li, H., Li, W. et Zhang, X. (2022a). A genetic algorithm for task offloading problem in vehicular edge computing. Dans *2022 China Automation Congress (CAC)*, 6242–6247. IEEE.

Li, H., Li, X., Zhang, M. et Ulziinyam, B. (2020). Multicast-oriented task offloading for vehicle edge computing. *IEEE Access*, *8*, 187373–187383.

Li, H., Sun, Y., Zhang, Y., Jin, B., Wang, Z., Wu, W. et Fang, C. (2021). Mobility-aware predictive computation offloading and task scheduling for mobile edge computing networks. Dans *2021 7th International Conference on Computer and Communications (ICCC)*, 1349–1354. IEEE.

Li, R., Huang, C., Zhang, H. et Jiang, S. (2022b). Multicast schedule for multi-message over multi-channel: A permutation-based wolpertinger deep reinforcement learning method. Dans *2022 IEEE/CIC International Conference on Communications in China (ICCC)*, 826–831. IEEE.

Mao, Y., You, C., Zhang, J., Huang, K. et Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, *19*(4), 2322–2358.

Mao, Y., Zhang, J. et Letaief, K. B. (2016). Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, *34*(12), 3590–3605.

Nath, S. et Wu, J. (2020). Deep reinforcement learning for dynamic computation offloading and

resource allocation in cache-assisted mobile edge computing systems. *Intelligent and Converged Networks*, *1*(2), 181–198.

Pirozmand, P., Hosseinabadi, A. A. R., Farrokhzad, M., Sadeghilalimi, M., Mirkamali, S. et Slowik, A. (2021). Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing. *Neural computing and applications*, *33*, 13075–13088.

Qadir, J., Baig, A., Ali, A. et Shafi, Q. (2014). Multicasting in cognitive radio networks: Algorithms, techniques and protocols. *Journal of Network and Computer Applications*, *45*, 44–61.

Qu, G., Wu, H., Li, R. et Jiao, P. (2021). Dmro: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing. *IEEE Transactions on Network and Service Management*, *18*(3), 3448–3459.

Sabella, D., Vaillant, A., Kuure, P., Rauschenbach, U. et Giust, F. (2016). Mobile-edge computing architecture: The role of mec in the internet of things. *IEEE Consumer Electronics Magazine*, *5*(4), 84–91.

Sinthiya, S. C., Shuvo, N. I., Mahmud, R. R. et Ahmed, J. (2022). Low-cost task offloading scheme for mobile edge cloud and internet cloud using genetic algorithm. Dans *2022 4th International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 1–6. IEEE.

Somuyiwa, S. O., György, A. et Gündüz, D. (2019). Multicast-aware proactive caching in wireless networks with deep reinforcement learning. Dans *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 1–5. http://dx.doi.org/10.1109/SPAWC.2019.8815489

Soon, G. K., Guan, T. T., On, C. K., Alfred, R. et Anthony, P. (2013). A comparison on the performance of crossover techniques in video game. Dans *2013 IEEE international conference on control system, computing and engineering*, 493–498. IEEE.

Sun, Y., Chen, Z., Tao, M. et Liu, H. (2020). Bandwidth gain from mobile edge computing and caching in wireless multicast systems. *IEEE Transactions on Wireless Communications*, *19*(6), 3992–4007. http://dx.doi.org/10.1109/TWC.2020.2979147

Vella, J.-M. et Zammit, S. (2012). A survey of multicasting over wireless access networks. *IEEE Communications Surveys & Tutorials*, *15*(2), 718–753.

Wang, D., Tian, X., Cui, H. et Liu, Z. (2020). Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware mec network. *China Communications*, *17*(8), 31–44.

Won, H., Cai, H., Guo, K., Netravali, A., Rhee, I., Sabnani, K. *et al.* (2009). Multicast scheduling in cellular data networks. *IEEE Transactions on Wireless Communications*, *8*(9), 4540–4549.

Yang, G., Hou, L., He, X., He, D., Chan, S. et Guizani, M. (2020). Offloading time optimization via markov decision process in mobile-edge computing. *IEEE internet of things journal*, *8*(4),

2483–2493.

Yuan, P., Li, M., Li, S., Liu, C. et Zhao, X. (2023). Maximizing the capacity of edge networks with multicasting. *Applied Sciences*, *13*(14), 8424.

Zalat, M. S., Darwish, S. M. et Madbouly, M. M. (2022). An adaptive offloading mechanism for mobile cloud computing: A niching genetic algorithm perspective. *IEEE Access*, *10*, 76752–76765.

Zhang, F., Ge, J., Wong, C., Li, C., Chen, X., Zhang, S., Luo, B., Zhang, H. et Chang, V. (2019). Online learning offloading framework for heterogeneous mobile edge computing system. *Journal of Parallel and Distributed Computing*, *128*, 167–183.

Zhou, Y., Yu, F. R., Chen, J. et Kuo, Y. (2018). Cache-aware multicast beamforming design for multicell multigroup multicast. *IEEE Transactions on Vehicular Technology*, *67*(12), 11681–11693.

Zhu, A. et Wen, Y. (2021). Computing offloading strategy using improved genetic algorithm in mobile edge computing system. *Journal of Grid Computing*, *19*(3), 38.