UNIVERSITÉ DU QUÉBEC À MONTRÉAL

ALLOCATION DES RESSOURCES ET DÉCHARGEMENT DES TÂCHES DANS LES RÉSEAUX VÉHICULAIRES AVEC TRAITEMENT PÉRIPHÉRIQUE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

ABABACAR GAYE

JANVIER 2024

REMERCIEMENTS

Au terme de ce travail, je tiens à exprimer ma profonde gratitude et mes sincères remerciements à :

Mon Directeur de recherche, Dr. Elmahdi Driouch, pour tout le temps qu'il m'a consacré, la qualité de son suivi, ses orientations et son soutien financier durant toute la période de ma formation ;

Tous les membres du jury pour leurs remarques et propositions visant à améliorer notre travail ;

Tous mes collègues au sein du laboratoire avec qui j'ai passé de très bons moments d'échange et d'entraide, en particulier Cheikh Ibrahima Mbaye et Ismael Bagayoko ;

M. Mouhamad DIEYE. Ses orientations ont été d'une grande importance ;

Tout le cadre professoral et administratif de l'UQAM ;

Ma bien-aimée exceptionnelle, Ramatoulaye Dianko, pour son soutien indéfectible, ses conseils précieux et son encouragement constant tout au long de l'élaboration de ce mémoire. Sa présence inspirante a grandement enrichi cette expérience et a contribué de manière significative à la réussite de ce projet ;

Mes parents et à toute ma famille, au sens africain du terme, merci pour votre soutien continu et votre amour inconditionnel ;

Toute personne qui a contribué de près ou de loin à la réalisation de ce travail.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ACRONYMES

ADD Approche aléatoire de déchargement

AGE Algorithme génétique évolutif

AHD Algorithme heuristique de déchargement

AGED Algorithme génétique évolutif de déchargement

AWGN Bruit blanc, gaussien et additif (en anglais Additive white Gaussian noise)

CCH Canal de contrôle (en anglais Control Channel)

CPU Processeur central (en anglais Central processing unit)

DQN Deep Q Network

DSRC Dedicated Short Range Communication

EC Informatique en périphérie (en anglais Edge Computing)

ESI European Telecommunications Standards Institute

FC Informatique de brouillard (en anglais Fog computing)

GPS Système mondial de localisation (en anglais Global positioning system)

ICN Réseau centré sur l'information (en anglais Information-centric networks)

IoT Internet des objets (en anglais Internet of things)

IoV Internet des véhicules (en anglais Internet of vehicles)

MEC Informatique mobile en périphérie (en anglais Mobile edge computing)

NFV Virtualisation de fonctions réseau (en anglais Network function virtualization)

OBU Unité embarqué (en anglais on-board unit)

QdS Qualité de service

RWS Sélection par roulette (en anglais Roulette Wheel Selection)

RSU Unité de bord de route (en anglais road side unit)

SDN Réseau logiciel (en anglais Software-defined networks)

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

STI Systèmes de transport intelligents

V2I Vehicule-à-infrastructure

V2V Vehicule-à-vehicle

VANET Réseau adhoc véhiculaire (en anglais Vehicular ad-hoc network)

VEC Informatique véhiculaire en périphérie (en anglais Vehicular edge computing)

WAVE Wireless Access for Vehicular Environment

RÉSUMÉ

Au cours des dernières années, l'Internet des véhicules, (en anglais *Internet of Vehicles* - IoV), a gagné en popularité de manière exponentielle grâce à la montée en puissance des véhicules connectés. L'IoV représente un nouveau concept au sein des systèmes de transport intelligents (STI), visant à substantiellement améliorer la sécurité et l'efficacité de la circulation routière en tirant parti des capacités de communication sans fil. Actuellement, la majorité des travaux de recherche se concentrent sur l'allocation des ressources énergétiques des dispositifs qui composent l'IoV tels que les unités de bord de route (en anglais *road side unit* - RSU) ainsi que sur les véhicules électriques.

Parallèlement, le concept novateur de l'informatique en périphérie (en anglais *Mobile edge computing* - MEC) émerge comme une stratégie complémentaire à l'infonuagique, déployant davantage de ressources de calcul et de stockage à proximité des utilisateurs. Néanmoins, le MEC se trouve confronté à des défis liés à la limitation des ressources informatiques comparativement à l'infonuagique. En conséquence, des domaines essentiels tels que la répartition de la charge et des tâches entre les unités de bord et les dispositifs des usagers demeurent en partie non explorés.

Dans ce mémoire, nous présentons une solution novatrice pour l'allocation des ressources et le déchargement de tâches au sein du réseau véhiculaire. Notre proposition repose sur un algorithme génétique, conçu pour maximiser l'efficacité du déchargement des tâches vers les serveurs des RSUs. Dans cette démarche, nous prenons en considération des critères cruciaux tels que la contrainte de délai des tâches ainsi que les ressources systémiques disponibles. Une comparaison est effectuée entre notre solution, un algorithme heuristique et une approche aléatoire. Les simulations menées démontrent les avantages de notre algorithme, qui se distingue par ses performances supérieures en termes d'utilisation prometteuse des ressources et de gestion des déchargements de tâches.

Mots clés : Internet des véhicules, systèmes de transport intelligents, véhicules connectés, unités de bord de route, calcul périphérique mobile, allocation des ressources, déchargement de tâches, algorithme génétique, heuristique.

CHAPITRE I

INTRODUCTION

1.1 Contexte

Dans le contexte contemporain, caractérisé par une profusion de capteurs sophistiqués, une prolifération de véhicules connectés et une expansion fulgurante des applications requérant d'importantes ressources, le domaine des transports intelligents se trouve à un tournant critique. Cependant, il est devenu manifeste que le modèle traditionnel de l'infonuagique (en anglais *cloud computing*) ne peut plus répondre de manière adéquate aux exigences fondamentales de faible latence des applications au sein des systèmes de transport intelligent (STI). Cette réalité est exacerbée par le flux de données massives échangées dans les réseaux sans fil en général, atteignant désormais l'échelle des pétaoctetsNing *et al.* (2019).

La diversité des sources de données est un élément essentiel de ce panorama, incluant les applications de géolocalisation GPS, une multitude de capteurs, des dispositifs de surveillance vidéo avancés, et même les plateformes de médias sociaux. L'Internet des Véhicules (*en anglais Internet of Vehicles* - IoV) est apparu comme une réponse à cette problématique, dotant les véhicules modernes d'ordinateurs embarqués et de technologies sans fil avancées. L'IoV facilite ainsi la collecte et le traitement d'un flux continu d'informations allant de la vitesse des véhicules aux comportements des conducteurs, en passant par les signes précurseurs de fatigue, les coordonnées GPS précises, l'état des routes, les incidents en temps réel et les itinéraires optimaux à suivre Sewalkar et Seitz (2019).

C'est dans ce contexte que l'IoV devient un catalyseur pour la production et le traitement de données massives et hétérogènes, convergentes vers les plateformes intelligentes des véhicules. Cette évolution a engendré une myriade de services en lien avec la sécurité routière, procurant avantages et commodités aux conducteurs et passagers. Toutefois, l'acheminement de ces données vers le nuage informatique crée des goulots d'étranglement majeurs, se traduisant par une congestion du réseau, une utilisation excessive de la bande passante et des retards de traitement inacceptables, en décalage avec les besoins des usagers et les exigences de certaines applications.

C'est dans cette perspective que le concept de l'informatique en périphérie mobile (en anglais *Mobile Edge Computing* - MEC) a vu le jour. Le MEC s'étend au-delà de l'infonuagique en permettant le déploiement de ressources de calcul, de stockage et de mise en réseau, à proximité immédiate des utilisateurs par le biais de serveurs MEC intégrés par exemples aux stations de base cellulaires et aux points d'accès WiFi. Les opérations de calcul générées par les applications des utilisateurs peuvent être éffectuées localement au sein du réseau périphérique, soulageant ainsi considérablement la pression sur les infrastructures du nuage Xie *et al.* (2019).

1.2 Motivation et problématique

Le concept du MEC a émergé comme une solution prometteuse pour relever les défis posés par les applications mobiles gourmandes en ressources et exigeant une très faible latence. Dans le contexte particulier de l'IoV, des serveurs MEC peuvent être mis en place au niveau des unités de bord de route (en anglais *road side units* - RSU) le long des voies de circulation. Ces RSUs peuvent ainsi agir en tant qu'agents de diffusion d'informations en mettant en cache les données et en

les diffusant aux véhicules en mouvement. Cette approche réduit la dépendance à l'égard du nuage informatique distant, conduisant à une réduction de la congestion réseau, des délais de traitement plus courts, une meilleure efficacité énergétique et une amélioration globale de la qualité de service (QdS) pour les utilisateurs Ning *et al.* (2019).

Cependant, la mise en œuvre du MEC ne se fait pas sans défis. Les ressources limitées des dispositifs, notamment la mémoire, le CPU et l'énergie, ainsi que les coûts élevés de déploiement et de maintenance des RSUs, constituent des obstacles significatifs. Dans ce contexte, se pose une question centrale : comment concevoir une stratégie d'allocation des ressources et d'optimisation du déchargement des tâches (en anglais *offloading*) qui peuvent répondre aux besoins des applications et des dispositifs mobiles au sein de l'environnement véhiculaire propre au MEC ? Cette recherche vise à explorer ces problématiques et à proposer des solutions algorithmiques novatrices pour les surmonter.

Si de nombreuses études se sont déjà penchées sur l'allocation des ressources au niveau de l'infonuagique et du MEC, l'application de ce dernier aux réseaux véhiculaires demeure un terrain de recherche peu exploré. Les contraintes telles que la durée de vie réduite de la batterie, les capacités de calcul et de stockage limitées des véhicules ainsi que leur mobilité, constituent des problèmes majeurs pour les applications nécessitant des temps de réponse courts et des ressources importantes. Malgré les avantages offerts par le MEC, la gestion efficiente des ressources limitées des serveurs MEC est impérative pour répondre aux exigences de QdS des utilisateurs et pour minimiser la consommation d'énergie des dispositifs mobiles Ranadheera *et al.* (2017). En examinant les enjeux et les opportunités du MEC au sein des réseaux véhiculaires, cette étude vise à explorer ces problématiques et à proposer des solutions pour favoriser des applications véhicluaires performantes.

1.3 Objectifs

L'essence de cette recherche réside dans le développement d'un nouvelle stratégie pour l'allocation des ressources et la gestion efficiente du déchargement des tâches dans le contexte du MEC appliqué aux réseaux véhiculaires. L'objectif central est de résoudre les défis complexes liés à la distribution de charges de calcul générées par des applications mobiles exigeantes en ressources tout en considérant les contraintes propres aux dispositifs mobiles et aux RSUs. Dans cette perspective, cette étude s'articule autour de plusieurs objectifs clés.

Tout d'abord, elle s'attache à réaliser une analyse des enjeux spécifiques auxquels le MEC est confronté dans les réseaux véhiculaires. Ceci englobe la compréhension des contraintes liées aux ressources limitées, aux impératifs de faible latence et aux charges de déploiement des RSUs. À partir de cette analyse, l'objectif consiste à concevoir des algorithmes d'optimisation pour le déchargement efficace des tâches des dispositifs mobiles vers les RSUs. Cette conception doit prendre en compte la diversité des besoins des utilisateurs, tout en tenant compte des limitations des ressources.

Une part essentielle de cette recherche réside dans l'évaluation des performances des stratégies développées. Pour ce faire, plusieurs simulation seront réalisées pour évaluer l'efficacité des approches d'optimisation en termes de latence, de capacité du réseau et de QdS.

1.4 Contributions

Ce mémoire présente une série de contributions visant à résoudre les défis du déchargement des tâches dans le contexte du MEC appliqué aux réseaux véhiculaires. Les principales contributions sont les suivantes :

Tout d'abord, ce mémoire offre une description détaillée de l'architecture du réseau étudié, ainsi que du modèle du système considéré. Cette description sert de base pour la formulation ultérieure du problème de déchargement des tâches en tant qu'un problème d'optimisation.

Une deuxième contribution consiste en la formulation d'un problème d'optimisation dont l'objectif est de maximiser le nombre total de tâches déchargées sur les serveurs des RSUs, tout en garantissant que ces tâches satisfont aux exigences strictes de faible latence.

Dans le but de résoudre ce problème complexe, nous proposons une heuristique basée sur l'algorithme génétique évolutif (AGE). Cette heuristique résout le problème formulé, capitalisant sur les concepts d'évolution et de sélection pour trouver des solutions optimales ou quasi-optimales dans des délais raisonnables.

En outre, pour évaluer l'efficacité de l'heuristique génétique, une évaluation des performances est réalisée. Cette évaluation comparative met en avant les avantages de l'approche génétique par rapport à une heuristique simple et à une approche aléatoire. Les performances sont évaluées en termes de qualité des solutions obtenues, de capacité de traitement du réseau, de latence des tâches et d'utilisation optimale des ressources.

En résumé, les contributions de ce mémoire résident dans la formulation mathématique précise du problème de déchargement des tâches dans le cadre du MEC, la proposition et l'évaluation d'une heuristique basée sur l'algorithme génétique, ainsi que l'analyse comparative démontrant l'efficacité de cette heuristique par rapport à d'autres approches. Ces contributions s'inscrivent dans une perspective d'amélioration significative des performances des applications mobiles dans les réseaux véhiculaires.

1.5 Plan du mémoire

Le reste de ce mémoire est structuré de la manière suivante :

Chapitre 2 : Généralités et état de l'art
Dans ce chapitre introductif, nous présenterons une vue d'ensemble du MEC, en mettant en évidence son architecture, ses caractéristiques et son rôle dans les réseaux véhiculaires. Une revue de la littérature y sera effectuée pour examiner les travaux antérieurs liés au déchargement des tâches dans le contexte du MEC ainsi qu'à l'allocation des ressources. Cette revue permettra de situer notre recherche dans le contexte scientifique actuel.

Chapitre 3 : Modèle de système et formulation du problème
Ce chapitre sera dédié à la description détaillée de l'architecture du réseau étudié dans le cadre de cette recherche. Nous exposerons également le modèle de système que nous considérons pour le problème du déchargement des tâches et la formulation mathématique du problème.

Chapitre 4 : Proposition de la solution pour le déchargement des tâches
Nous aborderons dans ce chapitre la solution que nous proposons pour résoudre le problème complexe de déchargement des tâches dans le MEC. Nous décrirons en détail l'algorithme génétique évolutif que nous avons conçu, ainsi que la solution heuristique simple et l'approche aléatoire que nous utiliserons pour des comparaisons ultérieures.

Chapitre 5 : Évaluation des performances
Dans ce chapitre, nous évaluerons les performances de notre solution proposée ainsi que des autres approches dans des scénarios de simulation réalistes. Nous comparerons les résultats obtenus en termes de qualité des solutions, de latence des tâches, d'utilisation des ressources et de capacité de traitement. Cette évaluation

comparative mettra en lumière les avantages et les inconvénients des différentes méthodes.

Cahpitre 6 : Conclusion générale

Cette dernière partie résumera les principales contributions de ce mémoire, soulignera les résultats obtenus et discutera brièvement des perspectives futures pour les travaux de recherche dans le domaine du MEC et du déchargement des tâches dans les réseaux véhiculaires.

CHAPITRE II

GÉNÉRALITÉS ET ÉTAT DE L'ART

2.1 Généralités

Dans ce premier chapitre, nous allons aborder du MEC tout en explorant son architecture distinctive, ses caractéristiques innovantes et son rôle fondamental au sein des réseaux véhiculaires. Le MEC incarne une avancée majeure en matière de traitement décentralisé en mettant à disposition des ressources informatiques distribuées au plus près des utilisateurs, offrant ainsi des possibilités novatrices pour déploiement de services plus performants dans un monde de plus en plus connecté.

Au-delà de cette présentation du MEC, nous entreprendrons également une revue des recherches antérieures qui se sont penchées sur le déchargement des tâches au sein du MEC et sur l'allocation efficace des ressources. Cette revue approfondie de la littérature nous permettra de tracer les évolutions scientifiques et technologiques ayant pavé la voie à notre propre démarche de recherche. En contextualisant notre étude au sein de cet écosystème en constante évolution, nous esquisserons les contours de notre contribution à la compréhension et à l'optimisation de ces concepts cruciaux.

2.1.1 Edge Computing : Vers une Approche Multi-accès en Périphérie

Le paradigme de l'infonuagique, dans sa forme traditionnelle, répond aux besoins en stockage de données et puissance de calcul des utilisateurs selon une approche

à la demande. Cependant, la centralisation des services dans le nuage informatique peut engendrer des problèmes majeurs. Les informations convergent vers un unique point central et sont ensuite distribuées en fonction des besoins, ce qui peut entraîner des goulots d'étranglement et des retards lorsque les demandes affluent simultanément Vhora et Gandhi (2020). Pour pallier ces défis, l'informatique en périphérie, également appelée *Edge Computing* (EC), a émergé en étendant les capacités de l'infonuagique jusqu'aux confins du réseau. Cette approche place les ressources de calcul, de stockage et de mise en réseau à proximité des sources de données Satyanarayanan (2017).

L'*Edge Computing* tire ses origines du concept de *cloudlets* Muniswamaiah *et al.* (2021). En effet, l'idée essentielle de rapprocher les capacités de calcul et de stockage des dispositifs utilisateurs dans le contexte de l'informatique en périphérie a été introduite pour la première fois en 2009, à travers le concept de *cloudlet* Satyanarayanan *et al.* (2009). Le *cloudlet* se réfère à la disposition stratégique d'ordinateurs dotés d'une puissance de calcul substantielle à des emplacements spécifiques, dans le but de fournir à la fois des capacités de traitement et de stockage aux dispositifs utilisateurs à proximité.

Le MEC est un concept qui désigne l'informatique déployée à la périphérie du réseau, proposant une alternative au modèle centralisé en offrant une latence minimale, une fiabilité accrue et une évolutivité optimale. Initialement associée au contexte des réseaux mobiles, la notion de périphérie s'est étendue. En 2017, l'ETSI (*European Telecommunications Standards Institute*) a étendu la portée du terme MEC pour englober non seulement les réseaux cellulaires, mais aussi d'autres types d'accès tels que le Wi-Fi et les points d'accès Qiu *et al.* (2020). Cette évolution a conduit à l'adoption de l'appellation *Multi-Access Edge Computing*, soulignant ainsi que la périphérie ne se limite pas aux réseaux mobiles. Ainsi, l'acronyme MEC peut désigner tantôt l'*Multi-Access Edge Computing*, tantôt le

Mobile Edge Computing, et ces termes sont interchangeables Vhora et Gandhi (2020).

L'implémentation du MEC repose sur une plateforme virtualisée qui tire parti des récents progrès en virtualisation des fonctions réseau *(network functions virtualization (NFV))*, en réseaux centrés sur l'information *(information-centric networks (ICN))* et en réseaux définis par logiciel *(software-defined networks (SDN))* Mao *et al.* (2017). De manière spécifique, le SDN permet aux administrateurs du réseau MEC de gérer les services via une abstraction des fonctions, ce qui favorise une évolutivité et une dynamique accrues Chang *et al.* (2016). Dans le même ordre d'idées, la NFV permet à un seul dispositif en périphérie de fournir des services informatiques à plusieurs appareils mobiles en créant plusieurs machines virtuelles, permettant ainsi l'exécution simultanée de différentes tâches ou fonctions réseau Hu *et al.* (2015). Par ailleurs, l'ICN introduit un nouveau paradigme de reconnaissance des services de bout en bout pour le MEC, en déplaçant l'approche centrée sur l'hôte vers une approche centrée sur l'information Mao *et al.* (2017). Un élément crucial dans la recherche en MEC réside dans l'évolution de ces technologies réseau générales afin qu'elles puissent être mises en œuvre au niveau des extrémités du réseau.

Une analyse complète des différentes conceptions architecturales proposées ainsi que de leur interconnexion avec les diverses générations de réseaux de communication a été fournie par Filali *et al.* (2020). Une enquête approfondie sur le MEC, mettant en lumière ses défis, outils et applications, a été réalisée par Vhora et Gandhi (2020). Par ailleurs, ils ont éclairé les distinctions entre le *Fog Computing* (FC), le *Multi-Access Edge Computing* (MEC) et le cloudlet (voir Figure 2.1) en termes de configuration des nœuds, d'implantation des nœuds, d'architecture logicielle, de capacités de collecte d'informations, de liens avec d'autres nœuds et de méthodes de connectivité disponibles.

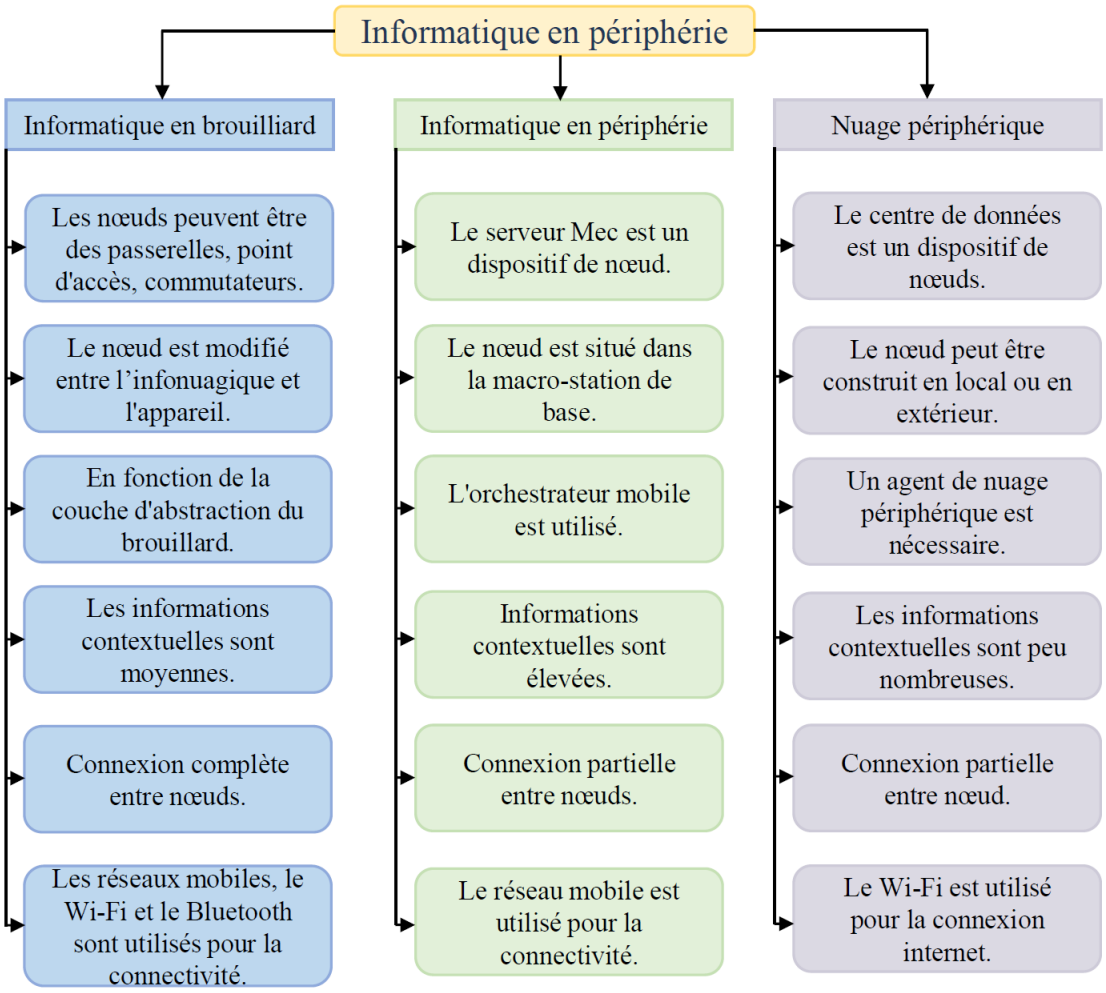

Figure 2.1. Disparités des approches en périphérie

Source Vhora et Gandhi (2020)

2.1.2 Architecture du Multi-Access Edge Computing

Dans le domaine de l'informatique en périphérie, différentes approches émergent en fonction des cas d'utilisation étudiés. Par exemple, dans l'étude menée par Marjanović *et al.* (2018), une architecture fonctionnelle a été proposée pour le MEC, une sous-catégorie spécifique de l'informatique en périphérie. Cette architecture se compose de quatre couches interconnectées, comme illustré dans la Figure 2.2.

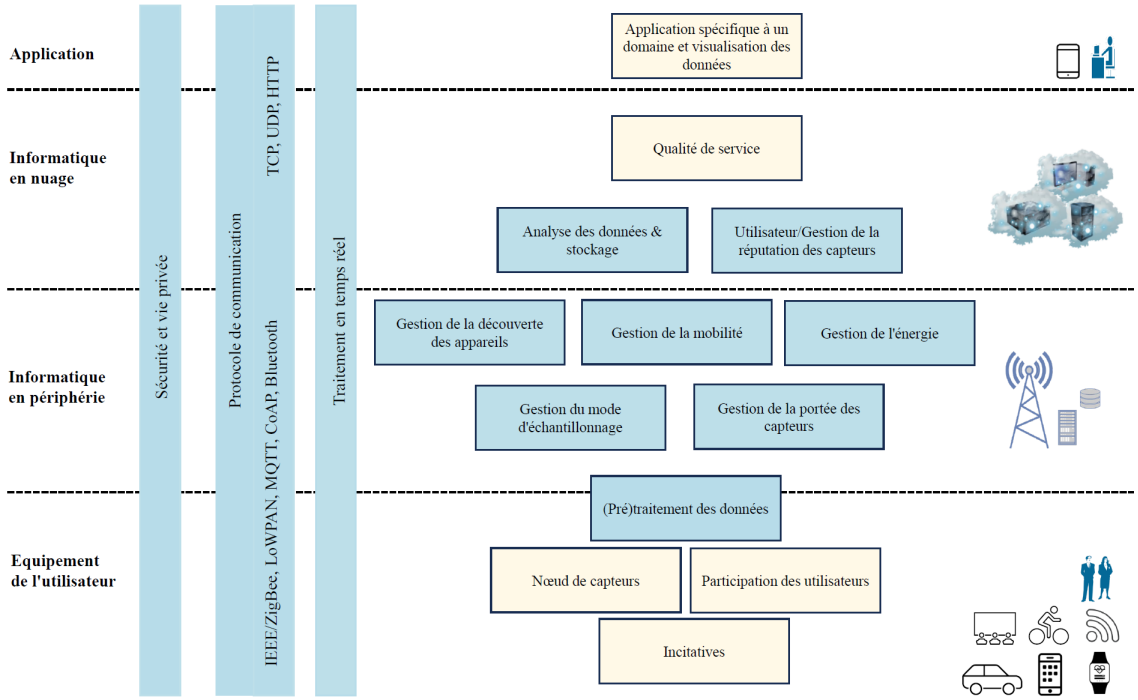

Figure 2.2. Architecture en quatre couches de l'informatique en périphérie

Source Marjanović *et al.* (2018)

Cette architecture est optimisée en tenant compte des attributs clés du MEC ainsi que des besoins spécifiques de l'infonuagique. Chaque couche occupe une position centrale dans l'ensemble du processus de traitement et de communication. Cette structure favorise une interaction active entre les utilisateurs et les ressources de calcul distribuées, assurant ainsi une mise en œuvre efficace des fonctionnalités de l'informatique en périphérie. Les auteurs Marjanović *et al.* (2018) ont détaillé les couches de la manière suivante :

— Applications

Créées dans les services du nuage informatique, les applications autorisent les utilisateurs à effectuer des tâches spécifiques. Elles offrent différentes interfaces

pour afficher les données et échanger des informations, accessibles via des interfaces Web et mobiles, incluant des notifications en temps réel.

— Infonuagique

Cette couche offre les ressources nécessaires pour effectuer des analyses de données complexes et pour le stockage à long terme. De plus, elle simplifie l'interaction entre les services promouvant ainsi la collaboration et le partage de données.

— Edge Computing

Positionnée à proximité des utilisateurs, cette couche surveille activement les utilisateurs, collecte leurs données, effectue des traitements et agrège les informations essentielles. Les serveurs en périphérie stockent des données sur les utilisateurs finaux, permettant ainsi leur réutilisation en cas de besoin. De plus, ces serveurs sont en mesure de détecter de nouvelles ressources et d'envoyer des réponses avec une faible latence.

— User Equipement

Au cœur de cette couche se trouvent une variété d'appareils dont des capteurs portables et de téléphones intelligents avec des capacités de détection diverses. Les données des capteurs peuvent être pré-traitées pour économiser l'énergie et la bande passante, avant d'être transmises aux serveurs en périphérie.

2.1.3 Caractéristiques et avantages

Plusieurs caractéristiques de l'informatique mobile en périphérie sont mises en avant dans les recherches menées par Liu *et al.* (2017), Muniswamaiah *et al.* (2021), Mao *et al.* (2017), Abbas *et al.* (2017) et Mach et Becvar (2017). Ces caractéristiques incluent notamment :

— la distribution possible des ressources, applications et services MEC à différents niveaux et emplacements d'un réseau sans fil, tels que les points d'accès, les routeurs, les stations de base et les appareils mobiles ;

— l'utilisation du nuage informatique pour gérer des tâches peu sensibles aux délais et requérant d'importantes capacités de calcul et de stockage ;

— une latence extrêmement réduite, de l'ordre de quelques millisecondes, pour soutenir des applications sensibles au délai ;

— une hétérogénéité des nœuds en termes de ressources de traitement, de stockage et de connectivité réseau ;

— la gestion de la mobilité, qui permet aux dispositifs mobiles de se connecter aux services de calcul en périphérie de réseau à travers des connexions sans fil, même lorsqu'ils traversent fréquemment différents points d'accès au réseau ;

— la capacité d'un serveur MEC à exploiter la proximité des dispositifs périphériques vis-à-vis des utilisateurs finaux, afin de surveiller en temps réel des données comme les comportements, les localisations et les contextes environnementaux.

En accord avec toutes les caractéristiques énoncées précédemment, le MEC offre les avantages suivants Xie *et al.* (2019), Marjanović *et al.* (2018), Mao *et al.* (2017) :

— Réduction de la latence : Le MEC rapproche le traitement des données des utilisateurs finaux, ce qui se traduit par une exécution plus rapide des tâches et une expérience utilisateur améliorée.

— Traitement localisé : Le MEC permet aux dispositifs de périphérie d'exécuter localement certaines tâches, optimisant ainsi l'utilisation des ressources.

— Sécurité améliorée : En gardant les données sensibles sur des serveurs locaux, le MEC réduit les vulnérabilités liées aux transferts de données sur des réseaux publics.

— Extensibilité et évolutivité : Le MEC permet un déploiement souple des ressources, équilibrant la charge de travail de manière plus efficace.

— Support pour les applications en temps réel : Les applications nécessitant une réponse rapide bénéficient d'une latence réduite grâce au traitement en périphérie.

— Réduction des coûts de bande passante : Le MEC minimise le besoin de transférer des données vers le nuage central, réduisant ainsi les coûts liés à la bande passante.

— Autonomie améliorée des appareils : Certaines tâches déchargées vers les périphéries contribuent à économiser la puissance de calcul et à prolonger la durée de vie des capteurs et des dispositifs mobiles.

— Adaptation aux exigences locales : Le MEC permet d'adapter les services en fonction des besoins spécifiques à chaque emplacement.

En résumé, le MEC réduit de manière significative les limitations des systèmes centralisés du l'infonuagique en offrant une approche distribuée mieux adaptée aux besoins des utilisateurs.

2.1.4 Internet des véhicules ou Internet of Vehicles (IoV)

Chaque année, un grand nombre de vies sont tragiquement perdues sur les routes en raison des accidents de la circulation. De plus, une part considérable de notre temps quotidien est gaspillée en raison des embouteillages importants. Cette situa-

tion entraîne également un gaspillage significatif de carburant et une augmentation constante de la pollution atmosphérique. Pour faire face à l'ensemble de ces défis, le concept de système de transport intelligent connaît un intérêt grandissant tant sur le niveau académique qu'industriel Darwish et Bakar (2018).

Le réseau ad hoc véhiculaire (en anglais *Vehicular Ad-Hoc Network*, VANET) est devenu un domaine de recherche florissant. Dans le cadre du VANET, les véhicules communiquent entre eux pour former un réseau. Cependant, la performance du VANET est sensiblement impactée par la mobilité des véhicules, dépendant de la densité du trafic et d'autres facteurs. Parallèlement, avec la prolifération des capteurs et de l'Internet des Objets (en anglais *Internet of Things*, IoT), ainsi qu'une augmentation du nombre de véhicules connectés à Internet, les réseaux ad hoc véhiculaires ont évolué vers le concept plus large de l'Internet des Véhicules (IoV) Alouache *et al.* (2018) Yang *et al.* (2014).

L'IoV offre des avantages significatifs en termes de communication et de coordination entre les véhicules, ouvrant la voie à des solutions potentielles pour améliorer la sécurité routière, la gestion du trafic et l'efficacité énergétique. Selon Darwish et Bakar (2018), l'Internet des véhicules est considéré comme une plateforme facilitant les interactions et les échanges d'informations (tels que l'état du trafic, la géolocalisation, la vitesse, etc.) entre les véhicules, les objets connectés, les individus et l'environnement. Les véhicules peuvent communiquer avec d'autres véhicules ou avec des unités routières (en anglais *road side unit*, RSU) via une unité embarquée (en anglais *on-board unit*, OBU) Xie *et al.* (2019).

L'émergence de la réalité augmentée, ainsi que le développement de la conduite autonome et d'autres applications novatrices, a considérablement enrichi les capacités des réseaux de véhicules Lu *et al.* (2014). Cependant, ces nouvelles applications sont généralement gourmandes en termes de capacités de calculs et requièrent la

gestion de grandes quantités de données, ce qui exerce une pression considérable sur les infrastructures classiques des réseaux de véhicules.

Pour relever ce défi, l'utilisation de l'informatique mobile en périphérie s'avère prometteuse. Elle vise à étendre les ressources de calcul et de stockage vers les périphéries du réseau. L'intégration de MEC permet ainsi d'accroître davantage les capacités de traitement et de stockage du réseau de véhicules, offrant une solution plus adaptée aux exigences croissantes des applications modernes Lu *et al.* (2014). La collaboration entre les réseaux MEC et les réseaux véhiculaires a émergé comme une tendance fondamentale pour répondre aux impératifs de faible latence et de bande passante exigés par les applications dans l'environnement du système de transport intelligent.

2.2 État de l'art

Dans cette section, nous entreprenons une revue de la littérature afin de présenter les travaux préexistants qui sont en corrélation avec la problématique étudiée dans ce travail. Notre examen se focalisera sur les recherches portant sur le déchargement des tâches et l'allocation de ressources dans le contexte du MEC au sein de l'environnement véhiculaire.

Nous procéderons à une exploration détaillée des publications scientifiques afin d'identifier les avancées, les défis et les solutions innovantes qui ont été abordés dans ce domaine. Cette revue de la littérature permettra de situer notre travail dans le contexte plus large de la recherche existante, tout en identifiant les lacunes et les opportunités d'amélioration qui guideront nos contributions et notre approche méthodologique.

2.2.1 Déchargement de tâches et allocation des ressources dans le MEC

En raison de leur capacité de calcul limitée, les appareils mobiles pourraient ne pas être en mesure d'exécuter toutes les tâches en temps voulu. De plus, l'exécution de ces calculs pourrait entraîner une consommation d'énergie très élevée, épuisant rapidement leur batterie Mao *et al.* (2017). En réponse à ces défis, une approche couramment adoptée est le déchargement de tâches. Cette méthode vise à surmonter les limitations de calcul des appareils mobiles en transférant les charges de travail intensives vers des serveurs ou d'autres dispositifs à la périphérie du réseau Liu *et al.* (2017). Une fois que ces tâches sont exécutées avec succès, les résultats sont renvoyés à l'appareil initial.

Le déchargement de tâches améliore les capacités des appareils mobiles tout en réduisant leur consommation d'énergie Djigal *et al.* (2022). Bien que les applications mobiles aient la possibilité de décharger des tâches de calcul vers le nuage informatique, le déchargement au niveau du MEC, sans avoir recours à Internet pour le transfert des tâches, contribue grandement à diminuer la latence, à économiser la bande passante, ainsi qu'à apporter d'autres avantages cités plus haut.

Dans le contexte de l'IoV, le calcul en périphérie des véhicules, également connu sous le nom de *vehicular edge computing* (VEC), tire parti des capacités de calcul des véhicules et des RSU équipées de serveurs VEC pour améliorer les capacités de traitement des tâches Lee et Na (2022). La gestion des ressources joue un rôle essentiel dans l'amélioration des performances du système VEC. Le processus d'allocation des ressources consiste à distribuer des ressources disponibles parmi un ensemble de dispositifs physiques à divers utilisateurs, en respectant les contraintes spécifiques liées à leur utilisation Wang *et al.* (2021). Du point de vue de la périphérie, cela englobe l'assignation des ressources de calcul, de stockage et de communication aux demandes de services spécifiques des utilisateurs. Une

allocation de ressources adéquate doit assurer des niveaux élevés de qualité d'expérience (QdE) et QdS, ainsi qu'une meilleure répartition des charges en fonction des ressources disponibles Djigal *et al.* (2022), Rodriguez et Buyya (2017).

Parmi les ressources étudiées dans le cadre de l'optimisation des performances du edge computing, les ressources de calcul et de communication ont été les plus explorées Filali *et al.* (2020). Un autre aspect important concerne le stockage, qui a fait l'objet d'études approfondies en relation avec la mise en cache en périphérie Liu *et al.* (2016).

L'optimisation du déchargement des tâches et de l'allocation des ressources constitue le cœur des problématiques du VEC. Dans l'IoV, la plupart des véhicules sont en mouvement, ce qui rend très complexe l'optimisation du système dans une topologie variant au fil du temps Ning *et al.* (2019). Il existe très peu de travaux de recherche qui étudient le déchargement des tâches des véhicules ainsi que l'allocation conjointe des ressources en tenant compte de la grande mobilité des véhicules. De plus, la majorité des travaux se concentre uniquement sur le déchargement des tâches de véhicule à véhicule, également appelé *vehicle-to-vehicle* (V2V), et sont très limités en ce qui concerne le déchargement des tâches par coopération entre les véhicules et les RSU, également appelé *vehicle-to-infrastructure* (V2I).

Dans Xiong *et al.* (2020), les auteurs ont privilégié la sélection de véhicules disposant de ressources abondantes en tant que véhicules de service. Ils ont élaboré une méthode d'apprentissage fédéré par renforcement Q visant à minimiser les coûts de communication et de calcul, ainsi que les probabilités d'échec de déchargement. Dans ce contexte, la croissance du nombre de véhicules pourrait potentiellement engendrer une surcharge des véhicules désignés en tant que véhicules de service, lorsque le nombre de véhicules sollicitant des services augmente.

L'optimisation conjointe du déchargement des tâches et de l'allocation des res-

sources a été abordée dans Wang *et al.* (2018), mais uniquement avec une seule station de base et plusieurs véhicules. Une approche de déchargement et de répartition des tâches de manière fédérée a été présentée, où une tâche est subdivisée en trois parties pour un traitement fédéré visant à optimiser le temps de calcul du véhicule. Une classification des véhicules en deux catégories : véhicules clients et véhicules de brouillard est effectuée par Zhou *et al.* (2019). Les auteurs ont mis en œuvre des techniques d'apprentissage automatique pour résoudre les problèmes liés au déchargement de données sensibles à la latence au sein des réseaux de véhicules.

Dans l'article Fan *et al.* (2023), les auteurs ont abordé l'optimisation conjointe du déchargement de tâches et de l'allocation des ressources pour le calcul en périphérie des véhicules. Ils ont axé leur étude sur les modes V2I et V2V, tout en considérant une unique RSU dans leur scénario. Leurs travaux ont mis en lumière des éléments que nous avons également explorés dans notre recherche, notamment la flexibilité accordée aux véhicules pour opter soit pour le traitement local de leurs tâches, soit pour leur déchargement vers la RSU par l'intermédiaire du mode V2I.

Cependant, il est important de noter que leur approche d'optimisation a été contrainte par des facteurs tels que la tolérance aux délais des tâches et la fenêtre temporelle durant laquelle les véhicules peuvent maintenir leurs connexions avant de quitter la portée de la RSU. Dans le contexte de notre propre étude, la durée pendant laquelle un véhicule demeure dans la zone couverte par la RSU ne constitue pas une restriction majeure. En effet, dans l'éventualité où un véhicule sort de la zone de couverture, le résultat de sa tâche est acheminé vers la prochaine RSU située sur son trajet. Cette approche garantit une transmission infaillible des résultats des tâches, indépendamment de la vélocité à laquelle le véhicule évolue au sein du réseau véhiculaire.

La contribution de Xu *et al.* (2019) s'est focalisée sur le déchargement de tâches impliquant des nœuds en périphérie ainsi que des véhicules dotés de capacité de calcul. Les chercheurs ont élaboré un schéma de routage des tâches, suivi d'une stratégie équilibrée de déchargement des dites tâches. Cependant, ils ont omis d'intégrer les capacités de calcul intrinsèques aux véhicules, préférant décharger l'intégralité des tâches vers les nœuds en périphérie.

Contrastant avec cette approche, notre propre recherche se distingue par l'inclusion minutieuse des ressources de calcul inhérentes aux véhicules. Notre méthodologie offre la possibilité de traiter les tâches soit via les RSUs, soit directement par les véhicules, pourvu que ces derniers disposent des ressources de calcul suffisantes pour l'exécution des tâches. Cette souplesse dans le traitement des tâches garantit une conformité aux contraintes de délai des tâches, tout en maximisant l'utilisation efficace des ressources disponibles.

L'étude menée par Cui et al. dans Cui *et al.* (2020) se concentre sur les problèmes d'optimisation du déchargement de tâches et de l'allocation de ressources. L'objectif principal de leur recherche est de minimiser le coût total du délai pour l'ensemble des véhicules au sein du système. Le travail repose sur l'hypothèse que lorsque plusieurs véhicules choisissent de décharger leurs tâches simultanément, la bande passante sans fil est attribuée de manière uniforme aux différentes tâches déchargées.

Cependant, il est important de noter que cette stratégie de répartition uniforme ne garantit pas une exploitation optimale des ressources disponibles. Une fois qu'un véhicule a achevé le déchargement de sa tâche, la bande passante doit être réallouée à une autre tâche qui est en cours de transfert vers la RSU afin d'éviter une sous-utilisation de la bande passante. Cette approche vise à réduire de manière efficace le temps de transmission des tâches vers le RSU, tout en encourageant

une allocation dynamique de la bande passante. Dans notre travail, nous prenons en compte ces considérations pour développer une stratégie d'optimisation plus aboutie du déchargement des tâches et de l'allocation des ressources.

Dans Hong *et al.* (2019), les auteurs ont étudié l'allocation optimale des ressources dans les systèmes MEC pour véhicules, dans le but de minimiser les délais de traitement des tâches. Ils ont proposé une approche visant à réduire la latence totale de traitement des tâches parmi un ensemble de véhicules dans une plage horaire en utilisant une perspective d'intégration de correspondance de ressources. Cette approche prend en compte la capacité de calcul propre à chaque unité de bord de route ainsi que la latence cumulée du déchargement des tâches, dans le but de formuler un plan d'allocation des ressources qui soit optimal. Les résultats obtenus mettent en évidence que cette stratégie permet de réduire efficacement la latence du réseau et d'apporter une amélioration significative aux performances globales du système.

Dans Li *et al.* (2020a) et Li *et al.* (2020b), l'allocation de ressources et le déchargement de calcul sont abordés dans le contexte d'un système VEC en adoptant l'hypothèse réaliste d'un canal de propagation variant dans le temps. Les auteurs ont formulé un problème visant à optimiser l'utilisation des ressources radio et de calcul tout en respectant les contraintes de délai des tâches. L'objectif principal est de maximiser l'utilité, définie comme étant le nombre de tâches déchargées vers le serveur VEC.

Initialement, les auteurs ont considéré le scénario où l'efficacité spectrale est fixe. Cela signifie que le taux de transmission des données reste invariable sur toute la durée de communication, négligeant ainsi la variation de la qualité du canal. Pour résoudre ce problème, ils ont proposé l'algorithme *Branch and Bound*, ainsi qu'un algorithme d'arrondi au nombre entier le plus proche, connu sous le nom

de *closest rounding integer* (CRI), afin de simplifier la complexité du LBB.

En s'appuyant sur les stratégies d'allocation de ressources basées sur l'efficacité spectrale fixe, les auteurs ont introduit les algorithmes LBBCO (*LBB based computation offloading*) et CRICO (*CRI based computation offloading*) pour résoudre le problème d'efficacité spectrale variant dans le temps. Ces approches ont également été étendues aux scénarios impliquant plusieurs véhicules et plusieurs tâches. De plus, les auteurs ont étudié l'impact de l'évanouissement (en anglais *fading*) à petite échelle sur les schémas de déchargement proposés. Les simulations ont confirmé les performances prometteuses des algorithmes proposés.

Dans l'article, Li *et al.* (2021), les mêmes auteurs mettent en avant l'importance du VEC pour améliorer la sécurité et la QdS dans les systèmes de transport intelligents. L'étude se concentre sur le problème conjoint de sélection de RSU et d'allocation de ressources, visant à minimiser le délai total de déchargement des tâches, en respectant les contraintes de bande passante et de ressources de calcul. Les chercheurs proposent des algorithmes pour les scénarios de RSU non collaboratives et collaboratives. L'approche consiste à reformuler le problème non convexe en un problème convexe et à utiliser des méthodes distribuées et de linéarisation pour résoudre les problèmes. Les résultats des simulations démontrent l'efficacité des algorithmes proposés par rapport aux méthodes de référence.

Dans Liu *et al.* (2023), les auteurs explorent des stratégies de déchargement de tâches et d'allocation de ressources dans un contexte IoV où les véhicules sont capables de récolter de l'énergie. Ils se sont penchés sur les scénarios d'heures creuses et d'heures de pointe, proposant des approches pour réduire le retard moyen d'exécution des tâches. Les solutions proposées incluent la sélection du mode d'exécution, le routage des données, la répartition des canaux, la mise en cache et la gestion de la mémoire cache. L'algorithme génétique et l'algorithme

Deep Q Network (DQN), sont utilisées pour résoudre ces problèmes d'optimisation complexes. Les simulations confirment l'efficacité de ces approches et soulignent la pertinence d'intégrer la technologie blockchain pour améliorer le déchargement des tâches et la sécurité des transmissions de données au sein de l'IoV.

Une approche collaborative pour le déchargement de calcul dans les réseaux véhiculaires en combinant le MEC et l'infonuagique est présentée dans Zhao *et al.* (2019). Les auteurs ont formulé un problème d'optimisation pour décider conjointement du déchargement de calcul et de l'allocation de ressources afin de maximiser l'utilité du système. Pour résoudre ce problème, les auteurs ont proposé un solution appelé CCORAO (*collaborative computation offloading and resource allocation optimization*), qui comprend une stratégie de déchargement de calcul et d'allocation de ressources. Ils ont également développé l'algorithme DCORA (*A distributed computation offloading and resource allocation*) pour le CCORAO, réduisant la complexité tout en maintenant les performances. Les résultats de simulation démontrent l'efficacité de leur approche, surtout lorsque les ressources MEC sont limitées.

Du *et al.* (2018) aborde les défis posés par les applications gourmandes en ressources des véhicules intélligents et propose une solution basée sur le MEC et l'utilisation des bandes de fréquence *TV white space* (TVWS) pour le déchargement de calcul. L'approche aborde une optimisation bilatérale qui vise à minimiser les coûts à la fois pour les véhicules intelligents et les serveurs MEC, tout en préservant la stabilité du réseau.

Pour résoudre ce problème, les auteurs ont présenté un algorithme basé sur l'optimisation de Lyapunov. Cet algorithme a pour objectif d'optimiser plusieurs aspects tels que le déchargement, la fréquence du CPU local, l'allocation des ressources radio et la provision du serveur MEC. Les résultats des simulations ont

démontré la convergence rapide de l'algorithme proposé ainsi que sa capacité à équilibrer efficacement les coûts et les délais. En comparaison avec d'autres approches, l'algorithme a montré une meilleure réduction des coûts.

2.2.2 Approche de solution génétique

Li *et al.* (2022) aborde le déchargement de tâches dans les systèmes de calcul en périphérie véhiculaire, en particulier pour les véhicules électriques. L'objectif est d'économiser l'énergie en déchargeant les tâches lorsque la batterie est limitée. Pour résoudre ce problème d'optimisation, les auteurs proposent l'utilisation d'un algorithme génétique. Les résultats expérimentaux montrent des économies d'énergie significatives par rapport à l'exécution locale des tâches.

Les auteurs de Su *et al.* (2023) ont également proposé une solution basée sur l'algorithme génétique NSGA-II. Leur objectif est de réduire le délai de traitement et la consommation de ressources en déchargeant les tâches de calcul vers les RSUs. À la différence de la plupart des approches existantes, leur solution prend en compte la coordination de plusieurs RSU ainsi que les exigences individuelles de QdS des différentes applications.

Pour résoudre les défis liés à la surcharge des serveurs du VEC causée par les applications qui deviennent de plus en plus gourmandes, une approche innovante a été proposée par Zeng *et al.* (2020). Ils ont introduit un modèle de VEC assisté par des véhicules "bénévoles". Ces derniers sont encouragés à aider les serveurs VEC surchargés en échange de récompenses. Dans leur étude, Zeng et al. ont utilisé un jeu de Stackelberg pour analyser les interactions entre les véhicules demandeurs de service et les serveurs VEC. Ils ont appliqué un algorithme génétique pour obtenir les meilleures stratégies pour les deux parties. Cette approche a démontré sa capacité à réduire les coûts de déchargement pour les véhicules et à améliorer

l'utilité des serveurs VEC, comme confirmé par les simulations réalisées.

Finalement, Song *et al.* (2022) ont étudié l'optimisation conjointe du déploiement des serveurs de calcul et des associations de déchargement d'utilisateurs dans les réseaux sans fil à l'informatique en périphérie. Leur approche repose sur un algorithme génétique personnalisé pour minimiser le délai de service moyen tout en respectant les exigences de délai individuelles des utilisateurs. Les résultats de simulation ont démontré que leur algorithme surpassait la recherche aléatoire et les heuristiques classiques en termes d'efficacité du délai de service.

2.2.3 Discussion de l'état de l'Art

L'état de l'art que nous venons de passer en revue présente un panorama de travaux axées sur le déchargement de tâches et l'allocation des ressources dans le contexte du VEC. Cette section se propose de discuter des différentes approches présentées dans cette littérature, en soulignant les problématiques qui n'étaient pas suffisamment étudiées.

— Mobilité

La mobilité des véhicules est un aspect souvent négligé. Les RSUs et les véhicules ont des trajectoires en constante évolution, ce qui peut entraîner des déconnexions ou des changements de connectivité. Cette limitation, bien que courante dans les scénarios réels, est rarement abordée dans les recherches existantes.

— Maximisation du nombre de tâches déchargées sur les serveurs VEC

Contrairement à certaines approches dans la littérature qui visent à minimiser la décharge de tâches sur les serveurs VEC, notre objectif est de maximiser cette décharge en optimisant l'utilisation des ressources de calcul en périphérie. En prenant en compte les avantages du déchargement de tâches, tels que la réduction de la latence et la préservation de l'autonomie des appareils mobiles, notre travail se concentre sur l'élaboration d'une stratégie robuste pour envoyer le maximum de tâches possibles vers les serveurs VEC.

Dans un contexte où les véhicules modernes sont équipés de capacités de calcul de plus en plus avancées, il devient essentiel d'exploiter ces ressources pour améliorer l'efficacité des systèmes VEC. Notre approche repose sur la notion que certaines tâches, bien que réalisables localement sur les véhicules, peuvent être traitées plus efficacement sur les serveurs VEC, en tirant parti de leur puissance de calcul supérieure. En maximisant le nombre de tâches déchargées, nous visons à optimiser l'utilisation des ressources en périphérie et à offrir des avantages significatifs en termes de latence, de bande passante et de qualité de service.

— Allocation de ressources collaborative

Une caractéristique des environnements VEC réside dans la présence de multiples véhicules et RSUs interagissant de manière dynamique. Cependant, on remarque que la plupart des articles antérieurs négligent souvent la complexité engendrée

par ce scénario de déchargement multi-véhicules et multi-RSU. Dans ce scénario, les défis d'allocation de ressources et de déchargement de tâches sont nettement plus complexes.

La collaboration entre les différentes entités joue un rôle clé pour le déchargement efficace des tâches. Notre approche se démarque des méthodes traditionnelles en explorant les opportunités de collaboration entre les véhicules et les RSUs, plutôt que de se focaliser sur des contextes multi-véhicules et mono-RSU. Nous reconnaissons que cette collaboration peut optimiser la répartition des charges, réduire la latence et exploiter au mieux les ressources en périphérie.

— Optimisation de la bande passante : Éviter la sous-utilisation

L'allocation de la bande passante sans fil dans les environnements du VEC est une dimension très importante, souvent négligée. Dans de nombreuses études précédentes, la bande passante était généralement répartie uniformément entre les tâches de déchargement, ce qui conduisait à une sous-utilisation des ressources sans fil.

Certaines tâches n'utilisant pas leur allocation pleinement tandis que d'autres restent en attente faute de ressources sans fil disponibles. Cependant, notre recherche reconnaît l'importance de l'allocation dynamique de la bande passante pour optimiser l'efficacité du système.

2.3 Conclusion

Dans ce chapitre, nous avons exploré l'état de l'art en matière de déchargement de tâches et d'allocation de ressources dans le contexte du VEC. Les travaux examinés révèlent des approches diverses, chacune abordant des aspects spécifiques de ce domaine en pleine expansion.

Notre revue de littérature a mis en lumière des défis cruciaux tels que l'allocation des ressources et la maximisation de plusieurs fonctions d'utilité. Cependant, des lacunes subsistent, notamment dans la considération approfondie de la collaboration entre entités multiples, la dynamique de la mobilité, et la maximisation du nombre de tâches déchargées sur les serveurs de périphérie.

Dans le chapitre suivant, nous passerons de l'examen des travaux existants à la présentation de notre propre modèle de système. Nous aborderons également la formulation précise du problème que nous chercherons à résoudre, à savoir la maximisation du nombre de tâches déchargées aux serveurs de périphérie.

CHAPITRE III

MODÈLE DE SYSTÈME ET FORMULATION DU PROBLÈME

Dans ce chapitre, nous commençons par présenter la modélisation du système considéré dans le cadre de cette étude. Ensuite, nous formulons le problème de la maximisation du nombre de tâches déchargées aux serveurs de périphérie.

3.1 Modèle du sytème

3.1.1 Architecture IoV Hybride

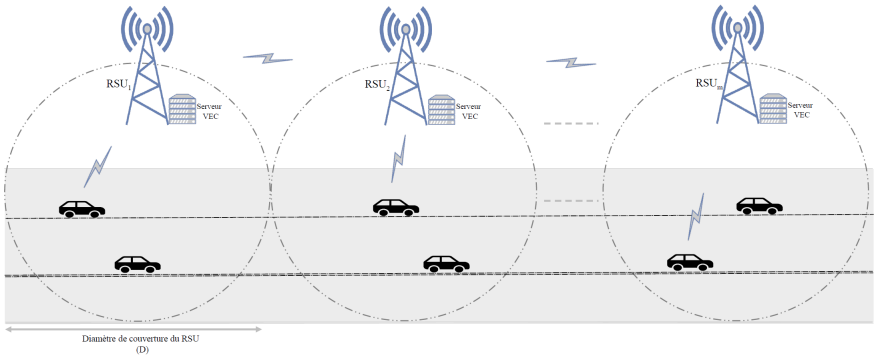

Figure 3.1. Architecture réseau véhiculaire

Notre travail se concentre sur le déchargement V2I (véhicule à infrastructure), comme le montre la Figure 3.1. Le scénario considéré se déroule sur une route rectiligne, telle qu'une autoroute, avec des véhicules circulant sur la même direction. Des unités de bord de route ou *Roadside Unit* (RSU) sont déployées le long de la route. Chaque RSU est équipé d'un serveur VEC (*vehicle edge computing*) pour traiter les requêtes des véhicules et d'un module émetteur/récepteur lui permet-

tant de communiquer avec les véhicules se trouvant dans sa couverture et avec les autres RSU. Les RSUs ont chacune une couverture et les véhicules circulant dans la portée de transmission ne peuvent accéder qu'au RSU correspondant. Lorsqu'un véhicule désire traiter une tâche, il le signale au RSU qui existe dans sa portée en envoyant un message *beacon*. En conséquence, la RSU recueille les données pertinentes de tous les véhicules ayant une tâche et se trouvant dans sa portée telles que les informations sur les tâches, l'emplacement, la vitesse, etc....

Chaque véhicule est équipé d'un OBU (*On-Board Unit*) qui lui permet de communiquer notamment avec la RSU. La suite de protocoles WAVE (*Wireless Access for Vehicular Environment*), et plus particulièrement la norme DSRC (*Dedicated Short Range Communication*) peut être utilisé pour la communication entre les véhicules et les RSUs. Le spectre de communication WAVE permet aux nœuds de diffuser régulièrement des messages appelé *beacon* sur le canal de contrôle ou *Control Channel* (CCH) pour déclarer les services qu'ils offrent (dans le cas d'un RSU) tels que la vitesse, la position, les demandes de traitement, etc Sorkhoh *et al.* (2019). Étant donné que les véhicules disposent d'un temps de séjour limité dans la zone de communication d'une RSU, il est crucial de gérer efficacement les ressources VEC de la RSU de manière à respecter les délais des tâches intolérantes aux retards.

Le serveur VEC doit céduler le traitement de ces tâches en respectant leurs échéances tout en minimisant le nombre de tâches refusées. Au début de chaque période d'ordonnancement, la RSU équipée de capacités VEC recueille toute l' information relative à l'ensemble des véhicules à portée et à leurs tâches de calcul souhaitées. Ensuite, la RSU effectue un filtrage préliminaire et rejette les tâches qui ne peuvent pas être traitées pour les raisons suivantes :

— les ressources requises pour exécuter la tâche est supérieure aux ressources

de la RSU ;

— le temps nécessaire pour traiter la tâche est supérieur à l'échéance de la tâche.

Toutefois, si la RSU à laquelle un véhicule soumet sa tâche est dans l'incapacité de la traiter, en raison de contraintes de ressources, elle peut l'acheminer vers une autre RSU disposant des capacités requises. On suppose que le réseau contient un contrôleur central qui dispose désormais d'une image entièrement du système, lui permettant de planifier le traitement des tâches de calcul au niveau des RSUs. Le contrôleur central décide ensuite des tâches à traiter et les véhicules à l'origine de ces requêtes sont instantanément avisés de les décharger. Le résultat d'une tâche est envoyé au véhicule correspondant dès qu'elle est terminée. Si le véhicule sort de la portée de communication de la RSU avant d'avoir reçu le résultat de sa tâche, la RSU envoie le résultat à la RSU où se trouve le véhicule, garantissant ainsi une transmission fiable des résultats de chaque tâche, quel que soit le degré de mobilité du véhicule dans le réseau véhiculaire.

3.1.2 Description du modèle

Le réseau véhiculaire considéré est composé de M RSU. Soit $\mathcal{M} = \{1, ..., M\}$ l'ensemble des RSUs qui est aussi considéré comme étant l'ensemble des serveurs VEC. Nous désignons par $\mathcal{I} = \{1, ..., I\}$ l'ensemble des véhicules, qui est aussi considéré comme l'ensemble des tâches. Chaque véhicule a une vitesse constante V et la période durant laquelle le véhicule traverse la couverture du RSU m est D/V, où D représente le diamètre de la couverture de la RSU. Nous définissons la tâche i par le triplet $\{c_i, s_i, t_i^{max}\}$, où c_i est le nombre de cycles CPU requis pour exécuter la tâche i, s_i est la taille du fichier de données de la tâche i et t_i^{max} est le délai maximal toléré pour accomplir la tâche i.

Chaque tâche peut soit être exécutée par un serveur VEC sélectionné ou localement par le véhicule. Nous désignons par δ_{im} la variable de décision pour le calcul de la tâche i et par β_{im} la variable de décision pour le déchargement de la tâche i. Lorsque la tâche i est envoyée à la RSU intermédiaire et exécutée à la RSU de calcul m, les conditions suivantes doivent être satisfaites : $\beta_{im} = 1$ et $\delta_{im} = 1$. Si $\delta_{im} = 1$, alors la tâche i est traitée au niveau du serveur VEC sélectionné m. Si $\delta_{i0} = 1$, la tâche i est exécuté en local et par conséquent $\sum_{m=1}^{M} \delta_{im} = 0$. Dans le cas contraire, c'est à dire si $\delta_{i0} = 0$, alors $\sum_{m=1}^{M} \delta_{im} = 1$.

Dans le scénario considéré, chaque véhicule à la possibilité d'envoyer sa tâche soit :

— à la RSU actuelle (c'est-à-dire la RSU où se trouve le véhicule) ;
— soit à une RSU intermédiaire ;
— soit de se déplacer jusqu'à la portée de la RSU de calcul m pour lui transmettre sa tâche i.

La RSU intermédiaire est alors chargée d'envoyer la tâche à la RSU de calcul m pour son traitement. Lorsque la tâche i est exécutée sur le serveur VEC m, le temps total de déchargement de la tâche peut être exprimé par :

$$T_i^{off} = T_i^{depl} + T_i^{envoi} + T_{i,rsu}^{trans} + T_i^{cal} + T_i^{resultat} \tag{3.1}$$

$$T_i^{off} \leq t_i^{max} \tag{3.2}$$

Les différents éléments de l'équation seront expliqués dans ce qui suit.

$\mathcal{M}$	ensemble des RSUs également ensemble des serveurs VEC.
$\mathcal{I}$	ensemble des véhicules également des tâches.
t_i^{max}	délai maximal toléré.
c_i	capacité requis pour exécuter la tache i
s_i	taille du fichier de données de la tache i
i	indice d'une tâche
m	indice d'une RSU
F_m^{max}	capacité de calcul maximale du serveur VEC
β_{im}	variable de décision pour le déchargement i
δ_{im}	variable de décision pour le calcul de i
$\hat{\gamma}_i$	indice de la RSU initial
$\hat{\beta}_i$	indice de la RSU intermédiaire
$\hat{\delta}_i$	indice de la RSU de calcul
T_i^{off}	temps total de déchargement de la tâche i
T_i^{depl}	temps de déplacement du véhicule
T_i^{envoi}	temps d'envoi de la tâche i du véhicule à la RSU
T_i^{trans}	temps de transmission de la tâche entre les RSUs
T_i^{cal}	temps de calcul de la tâche
T_i^{envoi}	temps d'envoi des résultats de calcul de la tâche i de la RSU au véhicule
D	diamètre de couverture d'une RSU
V	vitesse de chaque véhicule
M	nombre de RSU.
I	nombre de tâches
W	bande passante totale du sytème.
$\mathcal{B}$	ensemble de bande de fréquence.

Tableau 3.1. Tableau de notations

B	nombre de bande de fréquence.
b	indice d'une bande.
d_{im}	distance de communication entre véhicule et RSU
$r_i(t)$	débit de transmission à un instant t entre un véhicule et la RSU.
$h_{im}(t)$	gain du canal vers la RSU.
p_i	puissance de transmission du véhicule.
θ	exposant de l'affaiblissement de trajet.
N_i	puissance du bruit gaussien additif.
R_{rsu}	débit de transmission de données entre deux RSUs adjacentes.
$f_{i,0}$	nombre de cycles CPU allouées à la tâche i sur le véhicule.
$f_{i,m}$	nombre de cycles CPU allouées à la tâche i sur la RSU.
β	vecteur de sélection d'une RSU .
B	vecteur d'allocation de la bande passante.
F	vecteur d'allocation fréquence de calcul.
f_p	pénalité sur la fréquence.
t_p	pénalité sur le temps.
b_p	pénalité sur la bande passante.
F_h	fonction de compatibilité.
F_{ph}	probabilité de compatibilité.
C_{ph}	probabilité de cumulative.
μ_c	taux de croisement.
μ_m	taux de mutation.
$pop.size$	taille de la population.

Tableau 3.2. Tableau de notations suite

Les tableaux 3.1 et 3.2 donne un résumé plus clair de toutes les variables que nous utilisons dans ce travail.

Lorsque la tâche du véhicule i s'exécute sur le serveur VEC de la RSU m, le temps de déchargement peut être divisé en cinq parties :

a) Temps de déplacements du véhicule au RSU intermédiaire

C'est le temps que met le véhicule pour se rendre de la RSU où la tâche i est générée à la zone de couverture de la RSU m. Nous désignons par $\hat{\gamma}_i$ l'indice de la RSU initiale (RSU où se trouve le véhicle lorsque la tâche est générée) et $\hat{\beta}_i$ l'indice de la RSU intermédiaire (RSU de transmission, où la tâche va être transmise). Le temps de déplacement est alors donné par :

$$T_i^{depl} = (\hat{\gamma}_i - \hat{\beta}_i)\frac{D}{V} \tag{3.3}$$

b) Temps de transmission entre le véhicule et la RSU intermédiaire

La bande passante totale de ce système est W et elle est divisée en un ensemble $\mathcal{B}$ de B bandes de fréquence afin d'effectuer un accès multiple par répartition de fréquence. Chaque station de base peut attribuer une ou plusieurs bandes à un ensemble $\mathcal{K} \subseteq \mathcal{I}$ de K véhicules. Chaque véhicule $i \in \mathcal{K}$ se voit attribuer un ensemble $\mathcal{B}_{im} \subseteq \mathcal{B}$ de B_{im} bandes par la RSU m. À chaque RSU, une bande $b \in \mathcal{B}$ spécifique est alloué à au plus un véhicule $i \in \mathcal{K}$.

— Le débit :

Les temps déchargement de la tâche i et de téléchargement des résultats de calcul dépendent du canal sans fil entre le véhicule et la RSU. Lorsque les tâches sont déchargées, l'état du canal, qui comprend principalement l'affaiblissement du tra-

jet et l'évanouissement, ne peut être considéré comme constant. Les techniques de modulation et de codage adaptatifs sont couramment utilisées dans les systèmes sans fil à large bande pour ajuster le débit de données en fonction du rapport signal sur bruit (en anglais *signal-to-noise-ratio* ou SNR) reçu. En outre, la qualité du canal sans fil entre la RSU m et le véhicule i en mouvement est fortement influencée par l'affaiblissement sur le trajet, qui est fonction de la distance de communication d_{im}. L'affaiblissement sur le trajet augmente lorsque la distance de communication augmente Xing *et al.* (2015). Le débit de transmission à un instant t entre un véhicule i et la RSU m qui le sert est donné par :

$$r_i(t) = B_{im} \frac{W}{B} \log_2(1 + \mathrm{SNR}_{im}(t)) \tag{3.4}$$

Où $\mathrm{SNR}_{im}(t)$ est le SNR au niveau du RSU à l'instant t. Étant donné que les véhicules se déplacent à grande vitesse, nous supposons que le débit atteignable varie en fonction du temps, dans la couverture de la RSU. Le $\mathrm{SNR}_{im}(t)$ est donné par :

$$\mathrm{SNR}_{im}(t) = \frac{P_i h_{im}(t)}{N_i} \tag{3.5}$$

Où $h_{im}(t)$ est le gain du canal vers la RSU, qui dépend de la distance entre le véhicule i et la RSU, $h_{im}(t) = (d_{im}(t))^{-\sigma}$ avec σ est l'exposant de l'affaiblissement de trajet, p_i est la puissance de transmission du véhicule et N_i est la puissance du bruit blanc, gaussien et additif (en anglais *additive white Gaussian noise* ou AWGN) sur la totalité de la bande allouée au véhicule i. Il est clair que les temps de déchargement et de téléchargement des résultats de calcul dépendent de l'emplacement instantané des véhicules dans la couverture de la RSU. Plus le véhicule est proche de la RSU, plus le débit est élevé et plus il est éloigné de la RSU, plus

le débit de transmission est faible.

— Le temps de déchargement :

Le temps de transmission est calculé par :

$$T_i^{envoi} = \frac{s_i}{B_{im} r_i(t)} \tag{3.6}$$

où

$$r_i(t) = \int_{t_0}^{T_i^{envoi}+t_0} r_i(t)dt \tag{3.7}$$

Avec t_0 est le moment où un véhicule commence à décharger sa tâche i. Le débit $r_i(t)$ renferme le terme $d_{im}(t)$, où $d_{im}(t) = [(x_m - t \times V)^2 + y_m^2]$. Ici, V est la vitesse du véhicule, et (x_m, y_m) sont les coordonnées bidimensionnelles de la RSU, avec (0,0) désignant le point d'entrée dans la zone de couverture de la RSU Sorkhoh *et al.* (2019). La résolution de cette équation peut être évitée en divisant la couverture totale de la RSU en plusieurs zones, comme illustré à la figure 3.2. Nous supposons que le débit de transmission de chaque zone est constant et calculé à l'aide de $r_i(t)$ où la distance utilisée est la distance entre la RSU et le centre de la zone.

La figure 3.2, présente une représentation des débits de transmission potentiels pour différentes zones de distances entre le véhicule et la RSU, illustrant ainsi la division de la zone de couverture en segments distincts, chacun avec un débit spécifique en bits par seconde. Cette disposition reflète la réalité des réseaux sans fil, où une meilleure qualité de connexion est associée à une proximité accrue de la source du signal, en l'occurrence, la RSU. À mesure que la distance entre le véhicule et la RSU augmente, les débits de transmission diminuent progressivement,

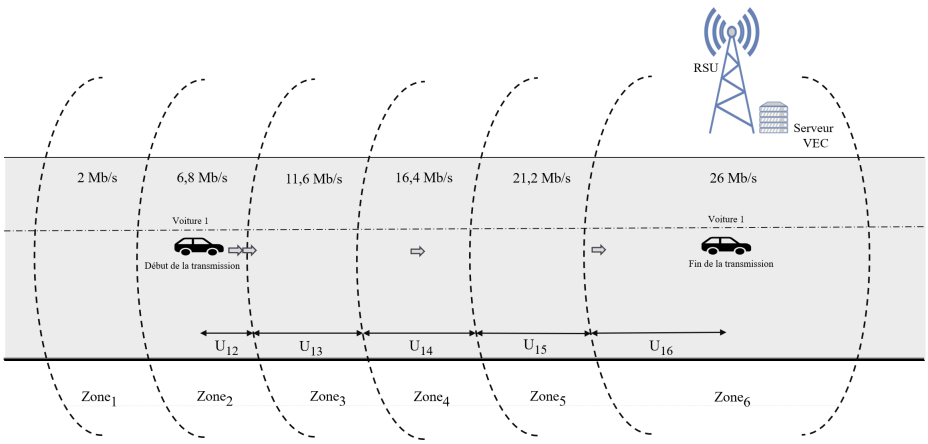

Figure 3.2. véhicule 1 décharge sa tâche à partir de la zone 2 jusqu'à la zone 6

Source Sorkhoh *et al.* (2019)

ce qui est conforme au comportement typique des réseaux sans fil. Ces valeurs de débit par zone sont essentielles pour une planification et une gestion efficace des réseaux de communication sans fil, permettant une modélisation précise de la capacité de transmission en fonction de la distance par rapport à la RSU.

c) Transmission entre RSU intermédiaire et RSU de calcul

Lorsqu'un véhicule envoie la tâche i à une RSU intermédiaire, ce dernier doit la transmettre à la RSU de calcul m. On désigne par $\hat{\delta}_i$ l'indice du RSU de calcul m, et rappelons que $\hat{\beta}_i$ est l'indice de la RSU intermédiaire. Le temps de transmission entre les RSUs peut être exprimé comme suit :

$$T_{i,rsu}^{trans} = (\hat{\delta}_i - \hat{\beta}_i)\frac{s_i}{R_{rsu}} \tag{3.8}$$

Où R_{rsu} est le débit de transmission de données entre deux RSUs adjacentes.

La relation entre $\hat{\delta}_i$ et δ_{im} peut être exprimée comme suit : $\hat{\delta}_i = \sum_{m=1}^{M} \delta_{im}$ et la

relation entre $\hat{\beta}_i$ et β_{im} peut être écrite comme suit : $\hat{\beta}_i = \sum_{m=1}^{M} \beta_{im}$.

Lorsque la tâche i est envoyée à la RSU intermédiaire et exécutée à la RSU de calcul m, les conditions suivantes doivent être satisfaites : $\beta_{im} = 1$ et $\delta_{im} = 1$. Une tâche peut être traitée par une RSU située dans une direction opposée à celle du véhicule. Par conséquent, nous pouvons réécrire l'équation (3.8) afin de calculer le temps de transmission entre la RSU intermédiaire et la RSU de calcul en fonction de l'emplacement des RSUs et de la taille de la tâche comme suit :

$$T_{i,rsu}^{trans} = \left| \left(\sum_{m=1}^{M} \delta_{im} \times m \quad - \quad \sum_{m=1}^{M} \beta_{im} \times m \right) \frac{s_i}{R_{rsu}} \right|. \tag{3.9}$$

d) Temps de calcul de la tâche

Lorsque le véhicule exécute sa tâche en local, le temps de calcul peut être exprimé comme suit :

$$T_{i,0} = \frac{c_i}{f_{i,0}} \tag{3.10}$$

Lorsque le véhicule exécute la tâche i sur le serveur VEC m, le temps de calcul peut être exprimé comme suit :

$$T_i^{cal} = \frac{c_i}{f_{i,m}} \tag{3.11}$$

Où $f_{i,0}$ et $f_{i,m}$ sont respectivement le nombre de cycles CPU allouées à la tâche i sur le véhicule et sur le serveur VEC m.

e) Temps de transmission du résultat de l'exécution

Comme les résultats des calculs sont très faibles, le temps d'envoi des résultats peut être ignorée Yang *et al.* (2019), Li *et al.* (2018).

3.2 Formulation du problème

Dans cette partie, nous formulons le problème de sélection conjointe des RSUs et d'allocation des ressources dans le but de maximiser le nombre total de tâches déchargées aux serveurs VEC. Nous allons commencer par décrire les contraintes du problème auxquelles le système doit satisfaire avant de terminer par la fonction objectif.

3.2.1 Les contraintes du problème

Les contraintes du problème se présentent comme suit :

— La contrainte d'allocation des ressources de calcul

Les ressources de calcul du serveur VEC étant limitées, cette première contrainte oblige de respecter ces limites. Elle peut être exprimée comme suit :

$$\sum_{i=1}^{I} \delta_{im} f_{i,m} \leq F_m^{max}, \forall m \in \mathcal{M} \tag{3.12}$$

Où F_m^{max} représente la capacité de calcul maximale du serveur relié à la RSU m.

— La contrainte de respect d'échéance de la tâche

Elle exprime que le temps de déchargement de la tâche i doit être inférieur ou égal à son délai maximal t_i^{max}, afin de garantir que l'exigence de délai de déchargement

soit respectée.

$$T_i^{off} \leq t_i^{max}, \forall i \in \mathcal{I} \tag{3.13}$$

— La contrainte de la bande passante maximale de la RSU

Elle assure que le nombre de bandes alloués par une RSU ne dépasse pas le nombre total de bandes dont il dispose.

$$\sum_{i=1}^{I} B_{im} \leq B, \forall b_{im} \in \mathcal{B} \tag{3.14}$$

Il est pertinent d'envisager l'ajout de contraintes pour tenir compte des besoins individuels des véhicules, notamment l'utilisation de variables de poids pour chaque tâche, permettant ainsi aux véhicules ayant des poids plus élevés d'obtenir une allocation de bande passante plus importante.

— La contrainte de transfert unique de tâches entre les RSUs intermédiaires

Elle indique que chaque tâche i peut être transférée vers au plus une RSU intermédiaire à la fois. Si $\beta_{im} = 1$ pour une RSU intermédiaire m, cela signifie que la tâche i est transférée du véhicule vers le RSU avant d'être envoyée à la RSU de calcul δ_{im}. Sinon, si $\beta_{im} = 0$, la tâche i est exécutée localement par le véhicule.

$$\sum_{m=1}^{M} \beta_{im} \leq 1, \forall i \in \mathcal{I} \tag{3.15}$$

$$\beta_{im} \in \{0, 1\}, \forall i \in \mathcal{I}, m \in \mathcal{M} \tag{3.16}$$

— La contrainte d'exécution unique des tâches

Elle garantit que chaque tâche i est exécutée au plus une fois. Elle peut être interprétée comme suit : si $\delta_{im} = 1$, alors la tâche i est exécutée par le serveur VEC m et ne peut être exécutée par aucun autre serveur. Sinon, si $\delta_{im} = 0$ pour tout m, alors la tâche i est soit exécutée localement par le véhicule, soit elle n'est pas encore attribuée à un serveur VEC.

$$\sum_{m=1}^{M} \delta_{im} \leq 1, \forall i \in \mathcal{I} \tag{3.17}$$

$$\delta_{im} \in \{0, 1\}, \forall i \in \mathcal{I}, m \in \mathcal{M} \tag{3.18}$$

— La contrainte de correspondance entre les variables de décision de déchargement de tâches et d'exécution de tâches

L'équation (3.19) lie les variables de décision β_{im} et δ_{im} pour s'assurer que si la tâche i est déchargée vers un serveur VEC sélectionné m ($\delta_{im} = 1$), alors elle doit être transférée depuis le véhicule vers le serveur VEC en passant par une RSU intermédiaire ($\beta_{im} \leq 1$).

$$\sum_{m=1}^{M} \beta_{im} \leq \sum_{m=1}^{M} \delta_{im}, \forall i \in \mathcal{I}, \tag{3.19}$$

En cas contraire, si la tâche i est exécutée localement par le véhicule ($\delta_{i0} = 1$), alors la contrainte est automatiquement satisfaite car $\sum_{m=1}^{M} \delta_{im} = 0$.

Ainsi, l'équation (3.19) garantit que les tâches sont transférées depuis les véhicules vers les serveurs VEC en passant par des RSUs intermédiaires lorsque cela est nécessaire.

— Les contraintes de sélection des RSUs intermédiaires et de calcul

Nous ajoutons les contraintes suivantes :

$$\beta_{im} \leq \sum_{j=1}^{m} \delta_{ij}, \forall i \in \mathcal{I}, m \in \mathcal{M} \tag{3.20}$$

$$\beta_{im} \leq \sum_{j=m}^{M} \delta_{ij}, \forall i \in \mathcal{I}, m \in \mathcal{M} \tag{3.21}$$

$$\sum_{m=1}^{M} \delta_{im} = 1, \forall i \in \mathcal{I} \tag{3.22}$$

Les contraintes (3.20) et (3.21) garantissent que les RSUs intermédiaires sont choisis parmi les RSUs situés devant le véhicule et qu'ils ne sont pas choisis parmi les RSUs que le véhicule a déjà dépassés.

La nouvelle contrainte (3.22) garantit que chaque tâche i est affectée à exactement une RSU de calcul parmi toutes les RSUs disponibles.

3.2.2 Fonction objectif

Dans ce travail, notre objectif vise à maximiser le nombre de tâches déchargées des véhicules vers les RSUs de calcul via les RSUs intermédiaires. La fonction objectif est formulé mathématiquement comme suit :

$$\text{Fonction objectif} = \sum_{i}^{I} \sum_{m=1}^{M} \beta_{im}. \tag{3.23}$$

Par conséquent, le problème étudié dans ce mémoire peut être écrit sous la forme :

$$\max_{\beta,B,F} \quad \sum_{i}^{I} \sum_{m=1}^{M} \beta_{im}$$
$$\text{s.c.} \quad (3.12) - (3.22) \tag{P1}$$

Le problème de déchargement des tâches de calculs est souvent difficile à résoudre en un temps raisonnable, car le nombre de combinaisons possibles de déchargement des tâches augmente rapidement avec le nombre de tâches à décharger. C'est pourquoi des approches métaheuristique, telles que l'algorithme génétique évolutif, sont souvent utilisées pour résoudre ce type de problème. Les algorithmes génétiques sont des algorithmes de recherche inspirés de la théorie de l'évolution qui utilisent des opérateurs de sélection, de croisement et de mutation pour trouver des solutions quasi-optimales dans un temps polynomial.

En utilisant l'algorithme génétique évolutif (AGE), il nous est possible de trouver une solution sous optimale pour le problème formulé dans un temps raisonnable et qui satisfait les contraintes. Toutefois, il est important de noter que la qualité de la solution trouvée dépendra fortement de la qualité de notre fonction d'évaluation et des paramètres de l'algorithme génétique.

3.3 Conclusion

Dans ce chapitre, nous avons examiné le problème de déchargement de tâches de calcul dans un contexte de réseau véhiculaire avec des RSU intermédiaires et des RSU de calcul. Nous avons formulé mathématiquement le problème en définissant les contraintes, ainsi que la fonction objectif qui traduit notre objectif principal.

L'introduction des variables de décision telles que β_{im} et δ_{im} permet de modéliser les choix de déchargement des tâches et d'affectation aux RSUs intermédiaires et

aux RSUs de calcul. Les contraintes introduites assurent la faisabilité du problème, prenant en compte des aspects tels que la capacité de calcul des serveurs VEC, le respect des échéances des tâches, la bande passante des RSUs, et la sélection appropriée des RSUs intermédiaires et de calcul.

Dans le prochain chapitre, nous mettrons en œuvre l'algorithme génétique évolutif pour résoudre le problème formulé.

CHAPITRE IV

ALGORITHME DE DÉCHARGEMENT DE TÂCHES ET D'ALLOCATION DE RESSOURCES BASÉ SUR AGE

Dans ce chapitre, nous commençons par présenter notre approche de résolution du problème formulé précédemment, basée sur l'algorithme génétique évolutif. Ensuite, nous décrivons une heuristique gloutonne simple ainsi qu'une approche aléatoire qui seront utilisées pour comparer les performances de l'algorithme génétique.

4.1 Algorithme génétique évolutif

4.1.1 Introduction

Un algorithme génétique évolutif (AGE) représente une méthode de recherche utilisée en optimisation mathématique afin de trouver des solutions quasi-optimales pour des problèmes ayant des espaces de recherche énormes Tabassum *et al.* (2014). Les algorithmes génétiques font partie de la famille des algorithmes évolutionnaires. Ils sont conçus pour trouver la solution la plus proche de l'optimal global à un problème donné en se basant sur des opérations tels que le croisement, la mutation, la sélection, etc... Varghese et Raj (2016). Inspirée de l'idée d'évolution naturelle de Charles Darwin, l'AGE est une technique itérative qui considère simultanément de nombreuses solutions dans l'espace de recherche. Une solution dans l'espace de recherche correspond dans le problème étudié dans ce mémoire à une solution de déchargement et d'allocation de ressources pour un ensemble de

tâche précis.

4.1.2 Description de l'algorithme

Un algorithme génétique définit les quatre notions suivantes :

1. Une population est un sous-ensemble de solutions de déchargement qui se met à jour à chaque nouvelle itération.
2. La taille de la population est le nombre d'éléments dans ce sous-ensemble.
3. Une génération désigne la population à chaque itération. Au fil des générations, la taille de la population reste inchangée.
4. L'élite représente l'ensemble des meilleures solutions de la génération actuelle qui sont directement transférées à la génération suivante sans modification.

Chaque solution de déchargement constitue un individu de la population et est représentée par un triplet d'affectation de tâches (i, β_i, δ_i) où i désigne l'indice de la tâche, β_i est le RSU intermédiaire vers la tâche est transmise et γ_i représente le RSU de calcul responsable du traitement de la tâche. Chaque triplet est appelé un gène, tandis que la séquence de gènes est appelée un chromosome. La longueur du chromosome h_c est égale au nombre de tâches I. Un exemple de chromosome représentant une solution de déchargement pour $I = 6$ tâches est présenté dans la figure 4.1, où la tâche 1 est déchargée sur la RSU intermédiaire 1 et exécutée sur la RSU de calcul 1. L'identifiant de la RSU intermédiaire et celui de la RSU de calcul étant les mêmes, alors la tâche est déchargée et traitée sur le même serveur. La tâche 4 n'est pas déchargée sur les serveurs, ce qui entraîne une valeur de 0 pour la RSU intermédiaires ainsi que pour la RSU de calcul. Dans ce cas, son exécution en local sur le véhicule est envisagée.

AGE utilise la sélection naturelle des solutions de déchargement les plus adaptées,

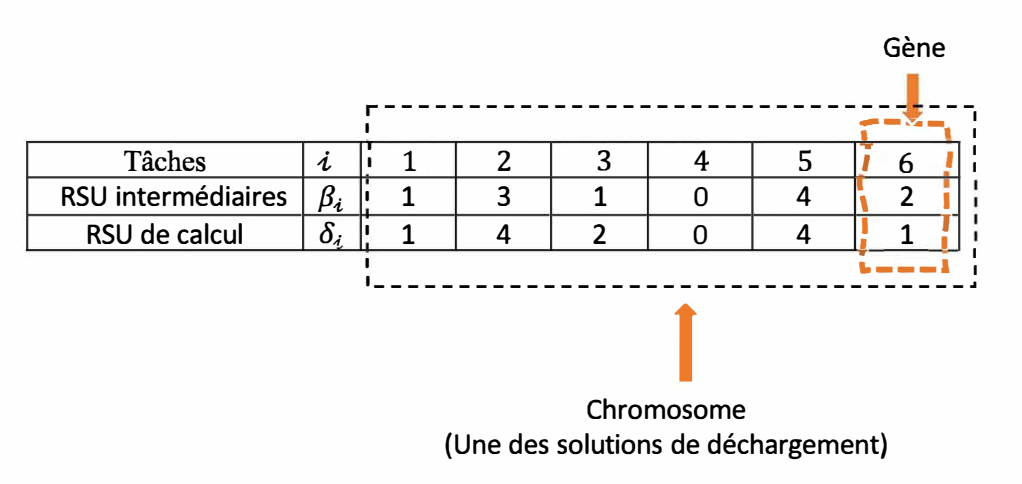

Figure 4.1. Exemple d'un chromosome qui représente une des solutions possibles de déchargement.

c'est-à-dire celles qui se rapprochent de la solution optimale, ont plus de chances à être sélectionnées pour faire partie de la génération suivante. La fonction objectif 3.23 est utilisée pour évaluer le degré de compatibilité (en anglais *fitness*) des solutions de déchargement. Afin de prendre en compte les contraintes du problème (3.12)-(3.22), le calcul de ce degré de compatibilité considère l'ajout d'une pénalité selon le niveau de violation de ces contraintes.

L'algorithme proposé comprend six étapes :

- initialisation d'un sous-ensemble de solutions de déchargement ;
- évaluation des solutions du sous-ensemble ;
- sélection des solutions les plus prometteuses ;
- croisement pour générer des solutions de descendance ;
- mutation des allocations de serveurs ;
- terminaison de l'algorithme.

Nous décrivons chaque étape en détail dans les sous-sections suivantes.

4.1.3 Initialisation

Au cours de cette phase, un sous-ensemble initial de solutions de déchargement est généré de façon aléatoire, c'est-à-dire que chaque tâche est affecté aléatoirement à une RSU intermédiaire β_i pour transmission et une RSU de calcul δ_i pour exécution. Le paramètre *pop.size* désigne le nombre total de solutions dans le sous-ensemble. Notons qu'une petite valeur de *pop.size* peut entraîner une convergence rapide, mais la solution peut être piégée dans des optima locaux, tandis qu'un nombre élevé élargit l'espace exploré avec une convergence plus lente vers l'optimal global.

L'algorithme 1 présente le pseudocode pour l'initialisation d'un sous-ensemble de solutions de déchargement.

Algorithme 1 : Initialisation

Data : *pop.size*, I, l'ensemble $\mathcal{M}$ RSUs

Result : Sous-ensemble initial de solutions de déchargement

```
1 for h = 1 à pop.size do
2     for i = 1 à I do
3         tâche ← i
4         intermédiaire ← β_i = selectRandom(M)
5         calcul ← δ_i = selectRandom(M)
6     end
7     chromosome(h) ← triplet (tâches, intermédiaire, calcul)
8 end
```

4.1.4 Évaluation des solutions de déchargement initiales

À cette étape de l'algorithme, la qualité de chaque solution de l'ensemble initial de solutions de déchargement est évaluée en utilisant une fonction de compatibilité dérivée de la fonction objectif 3.23. Pour gérer les contraintes d'optimisation définies par les équations (3.12) à (3.22), un mécanisme de pénalité est utilisé pour décourager les solutions qui ne respectent pas ces contraintes. Par conséquent, la fonction compatibilité qui sera utilisée consiste en la somme du nombre de tâches exécutées et une valeur de pénalité et est représentée comme suit :

$$F_h = \sum_{i=1}^{I} \sum_{m=1}^{M} \beta_{im} + \text{pénalité} \tag{4.1}$$

Plusieurs exemples de fonctions de pénalité sont couramment utilisés dans la littérature. Ils incluent :

1. les fonctions de pénalité statiques, qui attribuent des pénalités fixes à toutes les solutions qui ne respectent pas les contraintes ;
2. les fonctions de pénalité dynamiques, qui ajustent la valeur de la pénalité en fonction de la qualité des solutions ;
3. les fonctions de pénalité coévolutives, qui permettent aux pénalités de coévoluer avec les solutions ;
4. les fonctions de pénalité adaptatives Thakur *et al.* (2014).

.

Les fonctions de pénalité statiques sont utilisées dans notre travail. Elles sont statiques car elles sont définies à l'avance et ne changent pas au cours de l'exécution de l'algorithme. Nous considérons ainsi les fonctions suivantes :

$$f_p = \begin{cases} -1 & \text{si } \sum_{i=1}^{I} f_{im} \geq F_m^{max} \\ 0 & \text{sinon} \end{cases} \tag{4.2}$$

Où f_p désigne la pénalité sur la fréquence. Si la fréquence totale alloué est plus grande que la fréquence disponible alors on pénalise la solution.

$$t_p = \begin{cases} -1 & \text{si } \sum_{i=1}^{I} T_i^{off} \geq t_i^{max} \\ 0 & \text{sinon} \end{cases} \tag{4.3}$$

Où t_p est la pénalité sur la temps de déchargement. Si le temps de déchargement totale est plus grand que le délai de déchargement maximale, la solution est pénalisée.

$$b_p = \begin{cases} -1 & \text{si } \sum_{i=1}^{I} b_{im} \geq B \\ 0 & \text{sinon} \end{cases} \tag{4.4}$$

Où b_p représente la pénalité sur la bande passante. Si la bande passante totale allouée excède la bande passante disponible alors la solution est pénalisée.

L'algorithme 2 présente le pseudocode pour évaluer le sous-ensemble des solutions.

Algorithme 2 : Évalutation

Data : sous-ensemble initial de solutions de déchargement

Result : Score de compatibilité de chaque solution de déchargement

```
for h = 1 à pop.size do
    F_h = Σ_i^I Σ_{m=1}^M β_im
    for i = 1 à I do
        Calculer f_p en utilisant l'équation 4.2
        Calculer t_p en utilisant l'équation 4.3
        Calculer b_p en utilisant l'équation 4.4
    end
    Calculer F_h en utilisant l'équation 4.1
end
```

4.1.5 Sélection des solutions de déchargement les plus performantes

L'étape de sélection permet de choisir les solutions qui ont la meilleure valeur de compatibilité parmi un ensemble de solutions candidates. L'algorithme proposé utilise la méthode de sélection par roulette (en anglais *Roulette Wheel Selection*) (ou RWS) car c'est la méthode la plus utilisée dans le contexte de l'ordonnancement des requêtes dans les systèmes informatiques Akbari *et al.* (2017), Materwala *et al.* (2022). RWS est une méthode de sélection où la probabilité de sélectionner une solution de déchargement est proportionnelle à son degré de compatibilité. Dans RWS, la probabilité de compatibilité et la probabilité cumulative sont calculées à l'aide des équations (4.5) et (4.6) respectivement.

$$F_{ph} = \frac{F_h}{\sum_{h=1}^{pop_{size}} F_h} \tag{4.5}$$

$$C_{ph} = \sum_{i=1}^{h} F_{ph} \tag{4.6}$$

Dans ce travail, d'autres méthodes de sélection sont implémentées afin d'évaluer l'intérêt de RWS. Ces méthodes sont la sélection par tournoi (en anglais *Tournament Selection*) et la sélection par élitisme (en anglais *Elitism Selection*). Une roulette est construite en utilisant les probabilités calculées de manière que la zone occupée par chaque solution soit proportionnelle à sa probabilité de compatibilité. Un nombre aléatoire est ensuite généré pour chaque solution et placé sous la zone de la roulette à laquelle elle appartient. Les nouvelles solutions sont sélectionnées en fonction des nombres aléatoires attribués à chaque solution et de la proportion de la zone de la roulette qu'elles occupent en raison de leur probabilité de compatibilité. Le pseudocode pour la sélection est présenté dans l'algorithme 3.

Algorithme 3 : Sélection

Data : *pop.size*, sous-ensemble initial de solutions de déchargement

Result : Sous-ensemble de solutions

```
1  for h = 1 à pop.size do
2  |  Calculer F_ph en utilisant l'équation 4.5
3  |  Calculer C_ph en utilisant l'équation 4.6
4  |  Générer un nombre aléatoire (0 ou 1) en utilisant la sélection à partir de
   |   l'ensemble A
5  end
6  for g = 1 à pop.size do
7  |  for h = 1 à pop.size do
8  |  |  Calculer C_ph en utilisant l'équation 4.6 pour la solution h if Générer
   |  |   un nombre aléatoire (0 ou 1) en utilisant la sélection à partir de
   |  |   l'ensemble A > C_ph then
9  |  |  |  SolutionSélectionnée(g) est basée sur la solution de la génération
   |  |  |   précédente (h − 1)
10 |  |  end
11 |  end
12 end
```

4.1.6 Croisement

Le croisement permet de générer des solutions de descendance, appelées également enfants. Cette phase tente d'améliorer la qualité des solutions de la future génération en explorant l'espace de recherche dans le voisinage des solutions les plus prometteuses, sélectionnées lors de la phase de sélection. Le croisement augmente le nombre de solutions de déchargement appropriées dans le sous-ensemble de solutions en échangeant les allocations de serveur entre deux solutions parentes.

Le nombre de solutions parentes sélectionnées pour l'opération de croisement dépend du paramètre μ_c, appelé le taux de croisement (en anglais *crossover rate*). Ce taux définit la probabilité qu'une paire de solutions parentes subissent une opération de croisement pour produire des solutions enfants. La longueur de la séquence de gènes échangés lors de l'opération de croisement sera désignée par $longueurCoupure$.

Dans la littérature, plusieurs types de croisement sont utilisés. Parmi lesquelles nous citons :

— Croisement à un point (single-point crossover) : Ce type de croisement crée de la diversité en échangeant des parties entières de chromosomes à un seul point de coupure.
— Croisement à deux points (two-point crossover) : Le croisement à deux points offre une variation supplémentaire par rapport au croisement à un point, en échangeant des parties de chromosomes à deux points de coupure.
— Croisement uniforme (uniform crossover) : Ce type de croisement mélange chaque bit du chromosome avec une probabilité spécifique, permettant un mélange plus uniforme des gènes.

Chaque type de croisement peut être approprié dans différentes situations et est souvent sélectionné en fonction des besoins spécifiques de l'application Umbarkar et Sheth (2015).

Dans notre travail, nous avons implémenté et comparé ces trois méthodes de croisement. Les paires de solutions parentes sont sélectionnées aléatoirement en fonction d'un taux de croisement μ_c. Ensuite, elles subissent un croisement pour générer deux solutions enfants. Les deux meilleures solutions parmi les parents et les enfants sont conservées pour la génération suivante, afin d'améliorer les solutions.

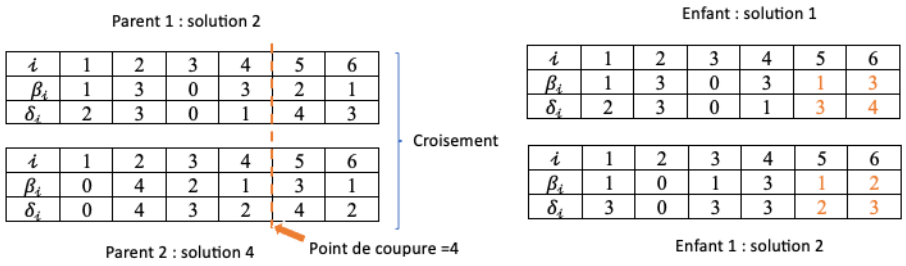

Figure 4.2. Croisement à un point sur les solutions de déchargement pour générer des solutions enfants.

La Figure 4.2 montre le croisement des solutions de déchargement, où un point de coupure aléatoire détermine l'échange des allocations de serveurs entre les parents. Les meilleures solutions parentes génèrent des solutions enfants, remplaçant les moins performantes. Ce processus est répété pour chaque paire, fournissant ainsi plus de choix à l'étape de sélection afin d'améliorer la qualité des solutions obtenues. Le pseudocode de l'opérateur de croisement à un point est présenté dans l'algorithme 4.

Algorithme 4 : Croisement

Data : $pop.size, \mu_c, longueurCoupure$

Result : Ensemble de solutions de déchargement

```
1  paires ← liste vide;
2  for h = 1 à pop.size do
3      prob(h) ← NombreAléatoire ∈ U[0, 1];
4      if prob(h) < μ_c then
5          paires.ajouter(h);
6      end
7  end
8  foreach paire ∈ paires do
9      parent_1, parent_2 ← SélectionnerParents(paire);
10     point_coupure ← NombreAléatoire ∈ U[1, longueurCoupure − 1];
11     for j = 1 à point_coupure do
12         descendant_1(j) ← parent_1(j);
13         descendant_2(j) ← parent_2(j);
14     end
15     for k = point_coupure + 1 à longueurCoupure do
16         descendant_1(k) ← parent_2(k);
17         descendant_2(k) ← parent_1(k);
18     end
19     solution ← Meilleur(parent_1, parent_2, descendant_1, descendant_2)
20 end
```

4.1.7 Mutation

La mutation est une étape essentielle dans les algorithmes génétiques. Elle sert principalement à introduire de la diversité au sein de la population de solutions, ce qui est important pour une meilleure exploration de l'espace de recherche, évitant ainsi les optimas locaux. Dans le contexte de notre problème, l'opérateur de mutation permet de créer de nouvelles solutions plus adaptées en modifiant certaines allocations de serveurs. Le taux de mutation μ_m est essentiel car il détermine la probabilité qu'une tâche soit sélectionnée pour mutation. Le nombre de tâches à muter (T_{mut}) est calculé en fonction du taux de mutation μ_m, déterminant ainsi combien de tâches seront modifiées dans la population.

À chaque itération, le processus de mutation commence par la sélection aléatoire d'une tâche parmi l'ensemble des tâches du problème. Cette sélection se fait de manière aléatoire pour garantir l'exploration de différentes possibilités.

Une fois que la tâche a été sélectionnée, nous calculons sa position actuelle dans la séquence de traitement en utilisant la longueur de la coupure spécifiée. Cette position nous permet d'identifier la solution à laquelle appartient la tâche, ainsi que la tâche précise qui doit être muter. Cette étape de localisation de la tâche est cruciale pour garantir que nous travaillons sur la bonne solution et la bonne tâche.

Après avoir identifié la tâche à muter, nous passons à l'étape de mutation proprement dite, qui consiste à choisir un serveur parmi l'ensemble des RSUs disponibles pour cette demande. Le choix du serveur peut dépendre de diverses stratégies, telles que la sélection du serveur le plus proche, le serveur ayant la capacité la plus appropriée, ou d'autres critères pertinents pour le problème.

En résumé, cette partie de l'algorithme effectue des mutations en sélectionnant

aléatoirement des tâches, en localisant la solution et la tâche précise à réallouer, puis en choisissant judicieusement un serveur parmi les RSUs disponibles pour la demande en cours de mutation. Cela permet d'explorer différentes allocations de tâches aux serveurs pour améliorer la qualité de la solution.

Algorithme 5 : Mutation

Data : $pop.size$, I, μ_m, $longueurCoupure$, l'ensemble $\mathcal{M}$ RSUs

Result : Sous-ensemble muté de solutions de déchargement

$T_{alloc} \leftarrow longueurCoupure \times pop.size$
$T_{mut} \leftarrow T_{alloc} \times \mu_m$
for $h \leftarrow$ *1 à* T_{mut} **do**
    $rnd(h) \leftarrow$ Générer un nombre aléatoire dans l'intervalle [1, T_{alloc}]
    $R_h \leftarrow$ reste de $rnd(h)$ divisé par $longueurCoupure$
    if R_h *égal à 0* **then**
        solution$_h \leftarrow \frac{rnd(h)-R_h}{longueurCoupure}$
        demande à réallouer$_h \leftarrow longueurCoupure$
    end
    else
        solution $\leftarrow \left(\frac{rnd(h)-R_h}{longueurCoupure}\right) + 1$
        demande à réallouer$_h \leftarrow R_h$
    end
    Serveur$_{rnd} \leftarrow$ Choisir parmi l'ensemble $\mathcal{M}$ RSUs
end

4.2 Algorithme heuristique de déchargement (AHD)

Cet algorithme constitue un benchmark pour comparer les performances de l'algorithme génétique. L'algorithme adopte une approche itérative pour attribuer les

tâches, en privilégiant d'abord les RSUs disponibles pour chaque tâche. L'exécution en local est envisagée en dernier recours. L'algorithme 6 présente le pseudo-code de AHD.

Initialement, l'algorithme commence par trier les tâches en fonction de leur date d'arrivée. Ensuite, à chaque itération de la boucle, l'algorithme examine les tâches de manière individuelle. Il commence par vérifier si une RSU est disponible pour l'exécution de la tâche en cours. Si tel est le cas, la tâche est alors assignée à ce serveur, ce qui entraîne une augmentation des compteurs de serveurs utilisés et des tâches exécutées.

Cependant, si aucun RSU n'est disponible pour la tâche en cours, l'algorithme vérifie si l'exécution locale est possible. Dans cette situation, la tâche est exécutée localement par le véhicule. Une nouvelle fois, cette action se traduit par une augmentation du compteur de tâches exécutées.

Ce processus itératif se répète pour toutes les tâches, garantissant que chaque tâche soit assignée à un RSU disponible, minimisant ainsi le recours aux ressources locales. À la fin de l'algorithme, les compteurs serveursUtilises et tachesExecutees fournissent les résultats finaux, indiquant respectivement le nombre total de serveurs utilisés et le nombre total de tâches exécutées.

Algorithme 6 : Algorithme heuristique de déchargement (AHD)

Data : Nombre de tâches I, Ensemble de RSUs $\mathcal{M}$

Result : Nombre de serveurs utilisés, Nombre total de tâches exécutées

```
1  serveursUtilises ← 0 tachesExecutees ← 0
2  Tri des tâches par ordre croissant de leurs instants
3  for i ← 1 à I do
4      foreach tache dans taches do
5          RSUDisponible ← Sélection du premier RSU disponible avec une
            capacité de calcul suffisante pour traiter tache avant son échéance
6          if RSUDisponible ≠ null then
7              Attribuer tâche à RSU (tache, RSUDisponible)
               serveursUtilises ← serveursUtilises + 1
               tachesExecutees ← tachesExecutees + 1
8          end
9          else if ExecutionLocalePossible(tache) then
10             Attribuer tâche au Véhicule (tache)
               tachesExecutees ← tachesExecutees + 1
11         end
12     end
13 end
14 return serveursUtilises, tachesExecutees
```

4.3 Approche aléatoire de déchargement (AAD)

L'algorithme ADD est une approche naïve où les tâches sont attribuées à des RSUs de manière aléatoire, sans tenir compte de la charge des RSUs ou des délais d'exécution des tâches. Les tâches sont d'abord triées par ordre d'arrivée, puis elles sont assignées de manière aléatoire à des RSUs disponibles. Si aucun RSU n'est

disponible, la tâche est envisagée en exécution locale. Enfin, les résultats finaux comprennent le nombre total de tâches exécutées et le nombre de RSUs utilisés, fournissant ainsi une répartition aléatoire des tâches sans stratégie de sélection avancée. Cette approche simplifiée peut être pertinente lorsque les contraintes de temps et de ressources ne sont pas critiques.

4.4 Conclusion

Ce chapitre a fourni une vue approfondie de la méthodologie d'optimisation adoptée, mettant en lumière les principes fondamentaux des algorithmes génétiques et leur application au problème spécifique du déchargement. La modélisation des objectifs d'optimisation et des contraintes a établi les bases d'une approche pour résoudre le problème complexe du déchargement.

L'accent a été mis sur les opérateurs génétiques, tels que le croisement et la mutation, et leur rôle crucial dans la génération de solutions diverses et de haute qualité. En parallèle, deux benchmarks ont été introduits : l'Algorithme Heuristique de Déchargement (AHD) et l'Approche Aléatoire de Déchargement (AAD), qui serviront de références pour évaluer les performances de l'algorithme génétique.

Au chapitre suivant, nous mettrons en œuvre ces concepts. L'analyse des résultats obtenus et l'ajustement éventuel des paramètres des algorithmes seront effectués. La comparaison des performances entre les différentes approches permettra d'évaluer l'efficacité de l'algorithme génétique dans le contexte du problème de déchargement.

CHAPITRE V

ÉVALUATION DES PERFORMANCES DE L'AGED

Dans ce chapitre, nous allons évaluer les performances de notre approche basée sur l'algorithme génétique évolutif (AGE) que nous appelons AGED (algorithme génétique évolutif de déchargement). À cette fin, nous allons le comparer à l'algorithme heuristique simple ainsi qu'à l'approche aléatoire présentées dans la section précédente.

5.1 L'environnement des simulations

Nous considérons un réseau véhiculaire où plusieurs RSUs sont positionnés le long d'une route unidirectionnelle. Sauf indication contraire, nous considérons la présence de quatre RSUs. Chaque RSU offre une couverture de 500 mètres. Les véhicules ont chacune une tâche et se déplacent à une vitesse de 100 km/h. La largeur de bande passante utilisée par chaque RSU est de 20 MHz. Les CPU des véhicules ont une fréquence maximale de 200 MHz, tandis que le serveur VEC atteint 10 GHz. Nous divisons la zone de chaque RSU en six régions et les ressources de chaque RSU en 100 bandes de fréquence (voir figure 3.2).

Les caractéristiques des tâches sont des variables aléatoires qui suivent des distributions gaussiennes. La taille des données des tâches est modélisée avec $s_i \sim \mathcal{N}(10, 1)$, mesurée en mégaoctets (Mo), tandis que les besoins en cycles CPU sont modélisés avec $c_i \sim \mathcal{N}(1, 0.1)$, mesurés en gigacycles. Chaque tâche a une échéance de 6 secondes après sa génération. L'exposant de l'affaiblissement du trajet θ a

une valeur de 3, la puissance du bruit σ^2 est égale à 10^{-13} W, et la puissance de transmission de chaque véhicule i est de 1 W.

Dans l'exécution de notre algorithme AGED, sauf indication contraire, nous avons défini le nombre de générations à 80, la taille de la population à 100 et la taille de l'élite est 20. Les taux d'opération de mutation et de croisement sont respectivement de 0,1 et 0,7. Par ailleurs, afin d'enrichir nos simulations, nous avons complété nos résultats en utilisant la méthode de Monte Carlo, avec une moyenne calculée sur 200 itérations.

Nous avons implémenté notre solution AGED, l'algorithme heuristique, et l'approche aléatoire en utilisant le langage de programmation Python.

5.2 Résultats

Afin d'évaluer la pertinence de notre solution, nous évaluons le pourcentage de tâches déchargées en fonction d'une multitude de critères.

La figure 5.1 présente la convergence de l'algorithme génétique proposé pour différentes valeurs du nombre de tâches. Pour les trois valeurs considérées, on remarque que l'algorithme réalise des gains de performance significatifs entre la première et la quarantième génération. L'amélioration devient de moins en moins importante au-delà de cette génération. Par conséquent, nous considérons pour les prochaines simulations un nombre de générations égal à 80 afin d'obtenir les performances les plus élevées possibles. Toutefois, il est à noter que ce paramètre étant réglable, il peut être choisit d'une façon arbitraire selon l'application et l'environnement d'opération considéré.

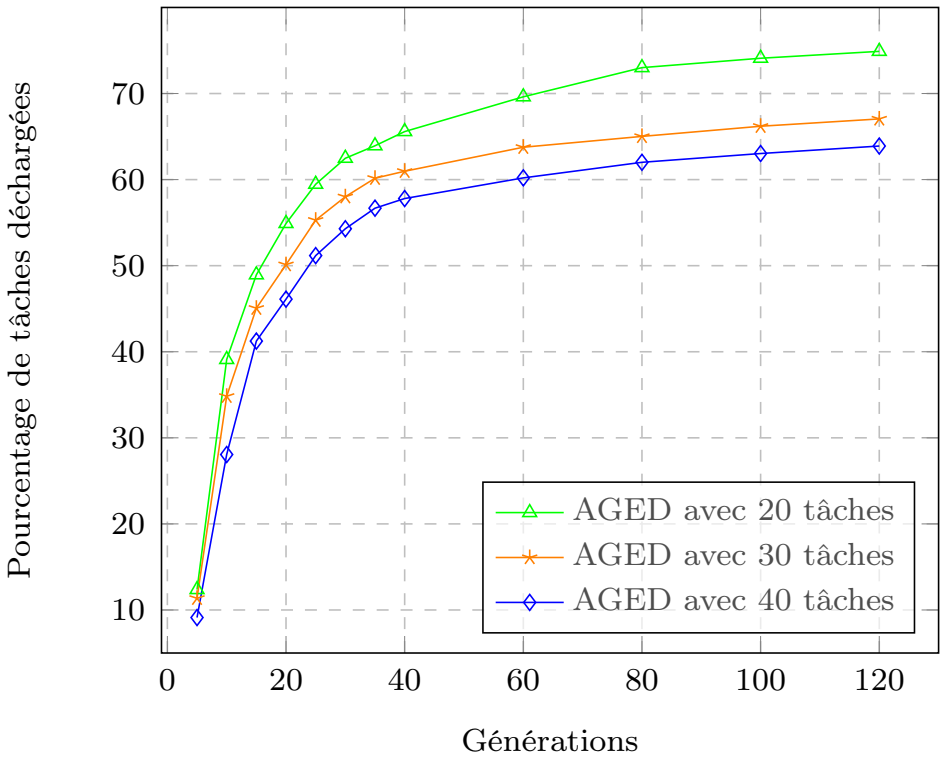

Figure 5.1. Pourcentage de tâches déchargées vs. le nombre de générations.

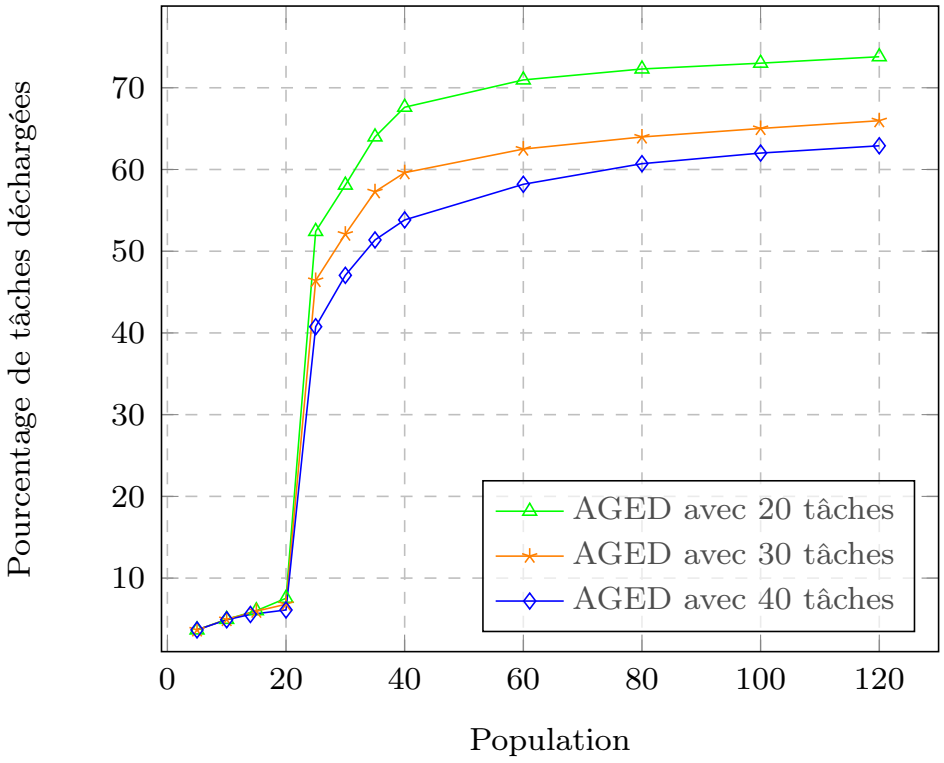

Figure 5.2. Pourcentage de tâches déchargées avec succès vs. taille de la population.

En examinant les courbes de la figure 5.2, on observe que plus la taille de la population est grande, plus le pourcentage de tâches déchargées avec succès tend à augmenter. Comme prévu, cette remarque suggère que l'augmentation de la population améliore les performances de notre algorithme génétique. Toutefois, on note une saturation des trois courbes de la figure. Pour le cas spécifique d'un nombre de tâches entre 20 et 40, l'amélioration de performance devient peu significative lorsque la taille de la population devient supérieure à 80. Par conséquent, on considère une taille de population de 100 chromosomes pour toutes les simulations suivantes.

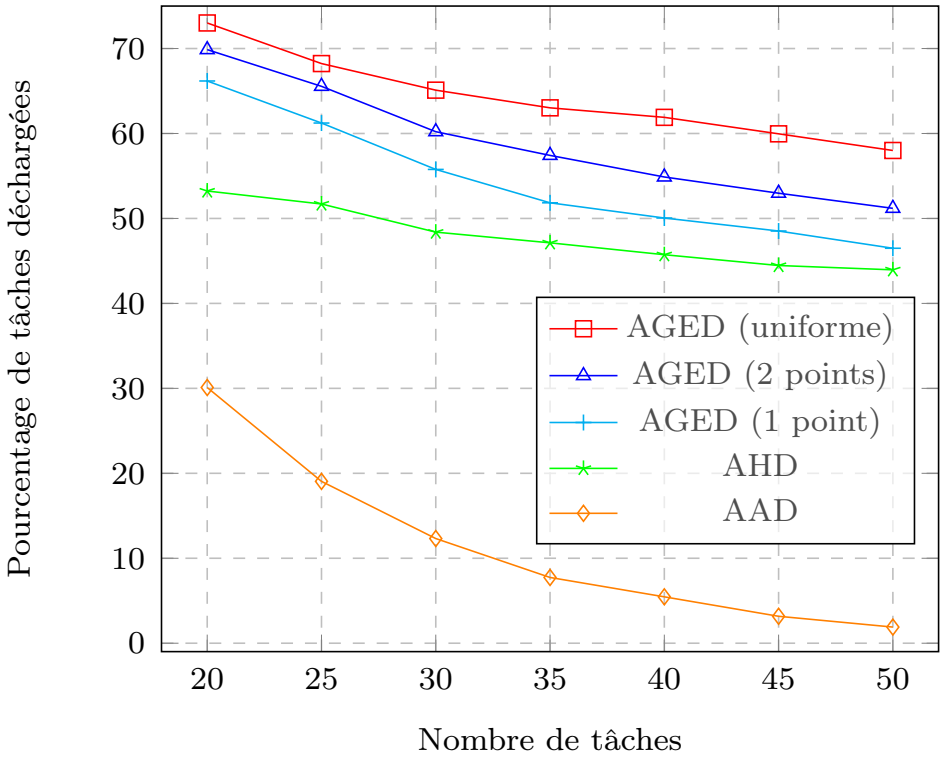

Figure 5.3. Pourcentage de tâches déchargées avec succès vs. le nombre total de tâches pour les trois méthodes de croisement.

Dans le cadre de cette recherche, nous avons implémenté trois opérateurs de croisement, à savoir le croisement à un point, le croisement à deux points et le croisement uniforme pour notre approche AGED. Dans la figure 5.3, nous pouvons observer que le croisement uniforme offre les meilleures performances. Lorsqu'il est utilisé, AGED parvient à exécuter entre 62% et 73% des tâches lorsque le nombre total de tâches passe de 20 à 50. En revanche, les deux autres opérateurs ont des performances clairement inférieures et atteignent respectivement entre 56% et 70% et entre 52% et 67% pour le même nombre de tâches. Il est important de noter que les performances du croisement uniforme viennent à un coût additionnel (même si limitée) en complexité des calculs. D'autre part, nous observons une nette supériorité de l'approche AGED par rapport aux deux algorithmes benchmarks, ADD

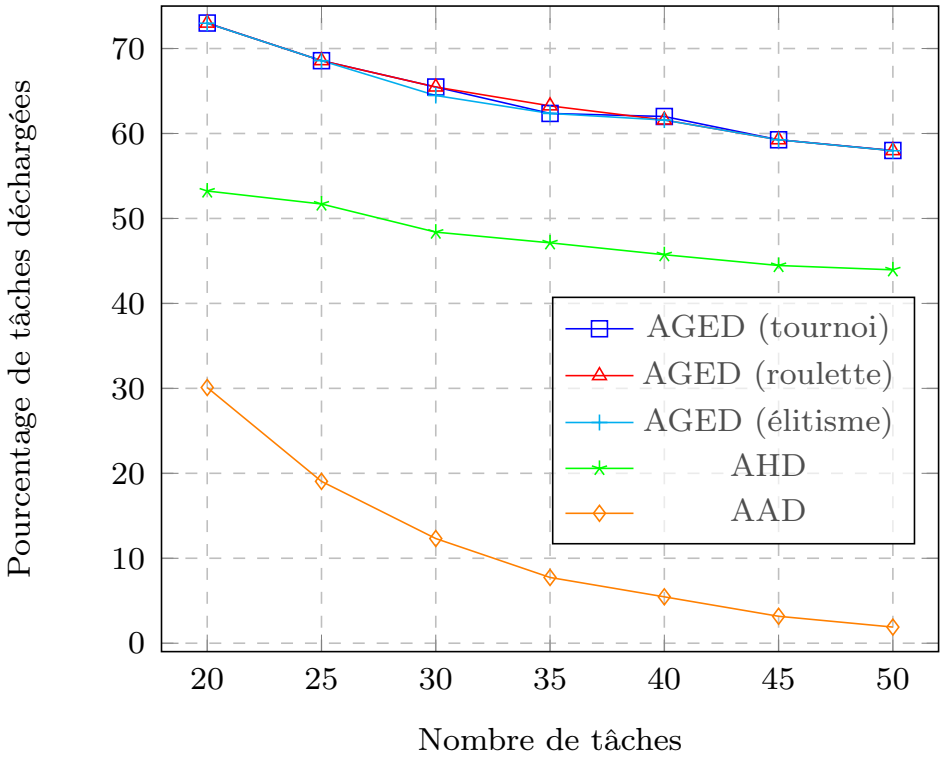

Figure 5.4. Pourcentage de tâches déchargées avec succès vs. le nombre total de tâches pour trois méthodes de sélection.

et AHD, et ce quel que soit la méthode de croisement utilisée.

D'une manière similaire à la comparaison des opérateurs de croisement dans la figure 5.3, nous comparons dans la figure 5.4 des opérateurs de sélection, soient la sélection par tournoi, la sélection par roulette et la sélection par élitisme. Nous remarquons que les trois opérateurs offrent presque les mêmes performances pour toutes les valeurs considérées pour le nombre total des tâches.

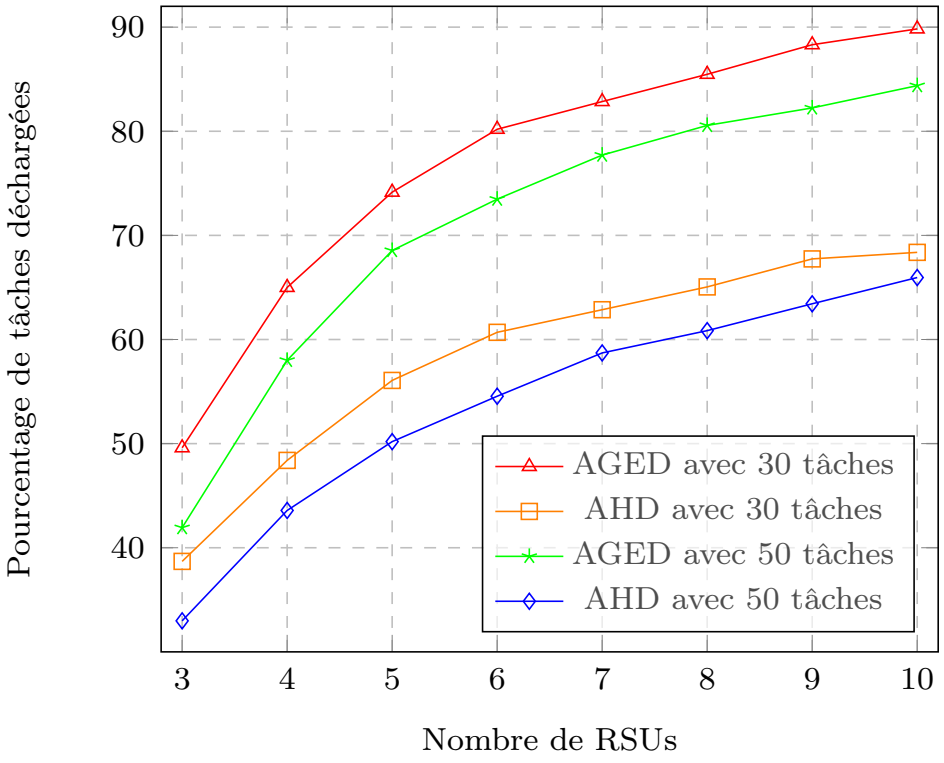

Figure 5.5. Pourcentage de tâches déchargées avec succès vs. le nombre de RSUs.

Dans la figure 5.5, nous comparons les performances de AGED dans deux scénarios différents, à savoir 30 tâches et 50 tâches, avec celles de l'algorithme heuristique dans les mêmes conditions. Dans les deux scénarios, le pourcentage de tâches déchargées augmente progressivement avec l'augmentation du nombre de RSUs déployés, à cause de la disponibilité grandissante des ressources de calcul dans le réseau. La figure démontre une fois de plus la nette supériorité de AGED par rapport à l'heuristique. En effet, grâce à une meilleure exploration de l'univers de solutions, AGED permet de tirer davantage profit de la présence de plus de ressources de calcul au niveau des RSUs.

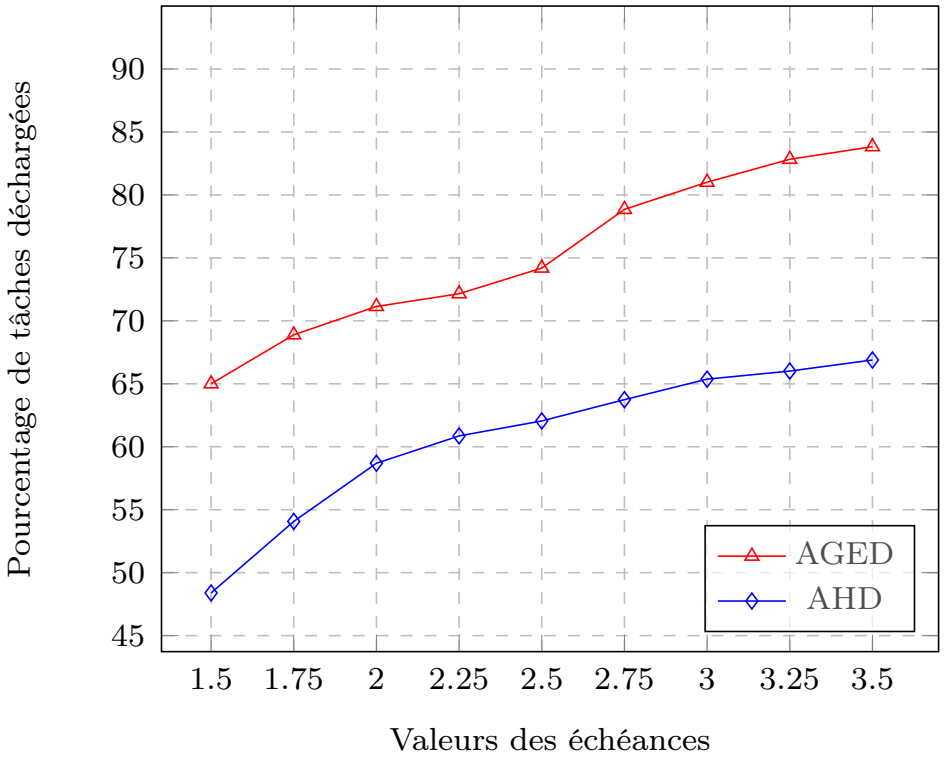

Figure 5.6. Pourcentage de tâches déchargées avec succès vs. le délai avant échéance.

Dans la figure 5.6, nous évaluons les performances de notre solution en faisant varier les valeurs des échéances. Notre approche AGED affiche des performances supérieures à celles de l'algorithme heuristique. À mesure que le délai maximal augmente, le pourcentage de tâches déchargées avec succès augmente pour les deux approches, atteignant respectivement 81% et 65% pour AGED et l'heuristique lorsque le délai est fixé à 3 secondes. L'évaluation a été effectuée avec 30 tâches en utilisant 4 RSUs.

CHAPITRE VI

CONCLUSION

L'exploration des défis complexes liés au déchargement des tâches dans le cadre du Mobile Edge Computing (MEC) appliqué aux réseaux véhiculaires a été le fil conducteur de ce mémoire. Notre démarche s'inscrit dans un contexte contemporain où le MEC, émergeant au-delà du modèle traditionnel de l'infonuagique, se révèle être une réponse stratégique aux exigences croissantes des systèmes de transport intelligents (STI). Ces derniers, caractérisés par une multitude de capteurs sophistiqués et une prolifération de véhicules connectés, imposent la nécessité de solutions innovantes pour relever les défis posés par la gestion efficace des ressources et l'optimisation du déchargement des tâches.

Dans ce paysage dynamique des STI, où la technologie joue un rôle central dans la transformation des transports, notre recherche vise à apporter une contribution significative en développant des stratégies pour la résolution des problèmes complexes liés au déchargement des tâches dans le contexte spécifique du MEC. Ce travail met l'accent sur une gestion efficace des ressources dans le but de maximiser les performances en termes du nombre de tâches exécutées sur le réseau.

Ce mémoire s'est attaquée aux défis complexes liés au déchargement des tâches dans le contexte du MEC appliqué aux réseaux véhiculaires. Notre objectif central était de concevoir des solutions algorithmiques pour l'allocation efficace des ressources radio et informatique et l'optimisation du déchargement des tâches, répondant ainsi aux exigences strictes sur leurs délais d'exécution.

Nous avons formulé le problème de déchargement des tâches comme un problème d'optimisation mathématique, cherchant à maximiser le nombre de tâches déchargées sur les serveurs des Road Side Units (RSUs) tout en respectant des contraintes de faible latence. Pour résoudre ce probléme, notre contribution majeure a été le développement d'un algorithme génétique évolutif de déchargement (AGED). Les évaluations approfondies ont clairement démontré que notre approche génétique surpassait de manière significative les méthodes heuristiques simples et les approches aléatoires en termes de qualité des solutions et du pourcentage de tâches déchargées avec succès sur les serveurs.

Cependant, ce domaine en constante évolution ouvre la voie à de nombreuses opportunités de recherche. Dans le futur, nous envisageons d'intégrer l'apprentissage automatique pour améliorer la prédiction des besoins de déchargement des tâches et optimiser la gestion des ressources énergétiques, étant donné que de nombreux dispositifs mobiles dépendent de batteries. Notre objectif sera de minimiser la consommation d'énergie des RSUs tout en maintenant des performances élevées.

En définitive, ce mémoire ne constitue pas seulement une contribution à la résolution d'un problème spécifique, mais également une invitation à une réflexion continue sur les défis et les opportunités offerts par l'intersection du MEC et des réseaux véhiculaires. Les résultats obtenus ouvrent des perspectives prometteuses pour l'amélioration des performances des applications mobiles dans les STI, contribuant ainsi à la conception de réseaux véhiculaires plus réactifs et performants.

RÉFÉRENCES

Abbas, N., Zhang, Y., Taherkordi, A. et Skeie, T. (2017). Mobile Edge Computing : A Survey. *IEEE Internet of Things Journal*, *5*(1), 450–465.

Akbari, M., Rashidi, H. et Alizadeh, S. H. (2017). An Enhanced Genetic Algorithm with New Operators for Task Scheduling in Heterogeneous Computing Systems. *Engineering Applications of Artificial Intelligence*, *61*, 35–46.

Alouache, L., Nguyen, N., Aliouat, M. et Chelouah, R. (2018). *New Robust Protocol for IoV Communications*, Dans *Challenges of the Internet of Things : Technique, Use, Ethics*, (p. 137–163).

Chang, C.-Y., Alexandris, K., Nikaein, N., Katsalis, K. et Spyropoulos, T. (2016). MEC Architectural Implications for LTE/LTE-A Networks. Dans *les actes de Workshop on Mobility in the Evolving Internet Architecture*, 13–18.

Cui, Y., Liang, Y. et Wang, R. (2020). Intelligent Task Offloading Algorithm for Mobile Edge Computing in Vehicular Networks. Dans *les actes de IEEE Vehicular Technology Conference (VTC2020-spring)*, 1–5.

Darwish, T. S. et Bakar, K. A. (2018). Fog Based Intelligent Transportation Big Data Analytics in The Internet of Vehicles Environment : Motivations, Architecture, Challenges, and Critical Issues. *IEEE Access*, *6*, 15679–15701.

Djigal, H., Xu, J., Liu, L. et Zhang, Y. (2022). Machine and Deep Learning for Resource Allocation in Multi-Access Edge Computing : A Survey. *IEEE Communications Surveys & Tutorials*, *24*(4), 2449–2494.

Du, J., Yu, F. R., Chu, X., Feng, J. et Lu, G. (2018). Computation Offloading and Resource Allocation in Vehicular Networks Based on Dual-Side Cost Minimization. *IEEE Transactions on Vehicular Technology*, *68*(2), 1079–1092.

Fan, W., Su, Y., Liu, J., Li, S., Huang, W., Wu, F. et Liu, Y. (2023). Joint Task Offloading and Resource Allocation for Vehicular Edge Computing Based on V2I and V2V Modes. *IEEE Transactions on Intelligent Transportation Systems*, *24*(4), 4277–4292.

Filali, A., Abouaomar, A., Cherkaoui, S., Kobbane, A. et Guizani, M. (2020). Multi-Access Edge Computing : A Survey. *IEEE Access*, *8*, 197017–197046.

Hong, G., Wen, Q., Su, W. et Wu, P. (2019). An Optimal Resource Allocation Mechanism in Vehicular MEC Systems. Dans *les actes de International Conference on Networking and Network Applications (NaNA)*, 34–38.

Hu, Y. C., Patel, M., Sabella, D., Sprecher, N. et Young, V. (2015). Mobile Edge Computing—A Key Technology Towards 5G. *ETSI white paper*, *11*(11), 1–16.

Lee, J. et Na, W. (2022). A Survey on Vehicular Edge Computing Architectures. Dans *les actes de International Conference on Information and Communication Technology Convergence (ICTC)*, 2198–2200.

Li, H., Li, W. et Zhang, X. (2022). A Genetic Algorithm for Task Offloading problem in Vehicular Edge Computing. Dans *les actes de China Automation Congress (CAC)*, 6242–6247.

Li, S., Chen, H., Lin, S. et Zhang, N. (2020a). Distributed Task Offloading and Resource Allocation in Vehicular Edge Computing. Dans *les actes de International Conference on Space-Air-Ground Computing (SAGC)*, 13–18.

Li, S., Lin, S., Cai, L., Li, W. et Zhu, G. (2020b). Joint Resource Allocation and Computation Offloading With Time-Varying Fading Channel in Vehicular Edge Computing. *IEEE Transactions on Vehicular Technology*, *69*(3), 3384–3398.

Li, S., Zhang, N., Chen, H., Lin, S., Dobre, O. A. et Wang, H. (2021). Joint Road Side Units Selection and Resource Allocation in Vehicular Edge Computing. *IEEE Transactions on Vehicular Technology*, *70*(12), 13190–13204.

Li, S., Zhu, G. et Lin, S. (2018). Computation Offloading with Time-Varying Fading Channel in Vehicular Edge Computing. Dans *les actes de IEEE Globecom Workshops (GC Wkshps)*, 1–6. IEEE.

Liu, D., Chen, B., Yang, C. et Molisch, A. F. (2016). Caching at the Wireless Edge : Design Aspects, Challenges, and Future Directions. *IEEE Communications Magazine*, *54*(9), 22–28.

Liu, H., Eldarrat, F., Alqahtani, H., Reznik, A., De Foy, X. et Zhang, Y. (2017). Mobile Edge Cloud System : Architectures, Challenges, and Approaches. *IEEE Systems Journal*, *12*(3), 2495–2508.

Liu, L., Yuan, X., Zhang, N., Chen, D., Yu, K. et Taherkordi, A. (2023). Joint Computation Offloading and Data Caching in Multi-Access Edge Computing Enabled Internet of Vehicles. *IEEE Transactions on Vehicular Technology, Early access.*

Lu, N., Cheng, N., Zhang, N., Shen, X. et Mark, J. W. (2014). Connected Vehicles : Solutions and Challenges. *IEEE internet of things journal*, *1*(4), 289–299.

Mach, P. et Becvar, Z. (2017). Mobile Edge Computing : A Survey on Architecture and Computation Offloading. *IEEE communications surveys & tutorials*, *19*(3), 1628–1656.

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.

Marjanović, M., Antonić, A. et Žarko, I. P. (2018). Edge Computing Architecture for Mobile Crowdsensing. *IEEE Access*, *6*, 10662–10674.

Materwala, H., Ismail, L., Shubair, R. M. et Buyya, R. (2022). Energy-sla-aware genetic algorithm for edge–cloud integrated computation offloading in vehicular networks. *Future Generation Computer Systems*, *135*, 205–222.

Muniswamaiah, M., Agerwala, T. et Tappert, C. C. (2021). A survey on cloudlets, mobile edge, and fog computing. Dans *les actes de IEEE international conference on cyber security and cloud computing (CSCloud)/2021 7th IEEE international conference on edge computing and scalable cloud (EdgeCom)*, 139–142.

Ning, Z., Huang, J., Wang, X., Rodrigues, J. J. et Guo, L. (2019). Mobile edge computing-enabled internet of vehicles : Toward energy-efficient scheduling. *IEEE Network*, *33*(5), 198–205.

Qiu, T., Chi, J., Zhou, X., Ning, Z., Atiquzzaman, M. et Wu, D. O. (2020). Edge computing in industrial internet of things : Architecture, advances and challenges. *IEEE Communications Surveys & Tutorials*, *22*(4), 2462–2488.

Ranadheera, S., Maghsudi, S. et Hossain, E. (2017). Mobile edge computation offloading using game theory and reinforcement learning. *arXiv preprint arXiv :1711.09012*.

Rodriguez, M. A. et Buyya, R. (2017). A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments. *Concurrency and Computation : Practice and Experience*, *29*(8), e4041.

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, *50*(1), 30–39.

Satyanarayanan, M., Bahl, P., Caceres, R. et Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, *8*(4), 14–23.

Sewalkar, P. et Seitz, J. (2019). Vehicle-to-pedestrian communication for vulnerable road users : Survey, design considerations, and challenges. *Sensors*, *19*(2), 358.

Song, H., Gu, B., Son, K. et Choi, W. (2022). Joint optimization of edge computing server deployment and user offloading associations in wireless edge network via a genetic algorithm. *IEEE Transactions on Network Science and Engineering*, *9*(4), 2535–2548.

Sorkhoh, I., Ebrahimi, D., Atallah, R. et Assi, C. (2019). Workload scheduling in vehicular networks with edge cloud capabilities. *IEEE Transactions on Vehicular Technology*, *68*(9), 8472–8486.

Su, M., Cao, C., Dai, M., Li, J. et Li, Y. (2023). Towards fast and energy-efficient offloading for vehicular edge computing. Dans *les actes de IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, 649–656.

Tabassum, M., Mathew, K. *et al.* (2014). A genetic algorithm analysis towards optimization solutions. *International Journal of Digital Information and Wireless Communications (IJDIWC)*, *4*(1), 124–142.

Thakur, M., Meghwani, S. S. et Jalota, H. (2014). A modified real coded genetic algorithm for constrained optimization. *Applied Mathematics and Computation*, *235*, 292–317.

Umbarkar, A. J. et Sheth, P. D. (2015). Crossover operators in genetic algorithms : a review. *ICTACT journal on soft computing*, *6*(1).

Varghese, B. M. et Raj, R. J. S. (2016). A survey on variants of genetic algorithm for scheduling workflow of tasks. Dans *les actes de International Conference on Science Technology Engineering and Management (ICONSTEM)*, 489–492. IEEE.

Vhora, F. et Gandhi, J. (2020). A comprehensive survey on mobile edge computing : challenges, tools, applications. Dans *les actes de International Conference on Computing Methodologies and Communication (ICCMC)*, 49–55. IEEE.

Wang, H., Li, X., Ji, H. et Zhang, H. (2018). Federated Offloading Scheme to Minimize Latency in MEC-enabled Vehicular Networks. Dans *les actes de IEEE Globecom Workshops*, 1–6.

Wang, S., Li, J., Wu, G., Chen, H. et Sun, S. (2021). Joint optimization of task offloading and resource allocation based on differential privacy in vehicular edge computing. *IEEE Transactions on Computational Social Systems*, *9*(1), 109–119.

Xie, R., Tang, Q., Wang, Q., Liu, X., Yu, F. R. et Huang, T. (2019). Collaborative vehicular edge computing networks : Architecture design and research challenges. *IEEE Access*, *7*, 178942–178952.

Xing, M., He, J. et Cai, L. (2015). Maximum-utility scheduling for multimedia transmission in drive-thru internet. *IEEE Transactions on Vehicular Technology*, *65*(4), 2649–2658.

Xiong, K., Leng, S., Huang, C., Yuen, C. et Guan, Y. L. (2020). Intelligent task

offloading for heterogeneous v2x communications. *IEEE Transactions on Intelligent Transportation Systems*, *22*(4), 2226–2238.

Xu, X., Xue, Y., Li, X., Qi, L. et Wan, S. (2019). A computation offloading method for edge computing with vehicle-to-everything. *IEEE Access*, *7*, 131068–131077.

Yang, F., Wang, S., Li, J., Liu, Z. et Sun, Q. (2014). An overview of internet of vehicles. *China communications*, *11*(10), 1–15.

Yang, X., Fei, Z., Zheng, J., Zhang, N. et Anpalagan, A. (2019). Joint multi-user computation offloading and data caching for hybrid mobile cloud/edge computing. *IEEE Transactions on Vehicular Technology*, *68*(11), 11018–11030.

Zeng, F., Chen, Q., Meng, L. et Wu, J. (2020). Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, *22*(6), 3247–3257.

Zhao, J., Li, Q., Gong, Y. et Zhang, K. (2019). Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Transactions on Vehicular Technology*, *68*(8), 7944–7956.

Zhou, S., Sun, Y., Jiang, Z. et Niu, Z. (2019). Exploiting moving intelligence : Delay-optimized computation offloading in vehicular fog networks. *IEEE Communications Magazine*, *57*(5), 49–55.