Efficient Mobile Edge Computing Development with Drone Integration

by

Talaye SANATKAR

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN ELECTRICAL ENGINEERING
M.A.Sc.

MONTREAL, "SEPTEMBER 18, 2024"

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Michel Kadoch, Thesis supervisor
Department of Electrical Engineering, École de technologie supérieure

Mr. Stéphane Coulombe, Chair, Board of Examiners
Department of Software and IT Engineering, École de technologie supérieure

Mr. Kim-Khoa Nguyen, Member of the Jury
Department of Electrical Engineering, École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON "AUGUST 26, 2024"

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

At the outset, I would like to sincerely thank Professor Kadoch my supervisor. Throughout my research journey, his unfailing support, useful assistance, and insightful critiques have been vital. His knowledge and guidance greatly influenced my academic and professional development in addition to enriching this thesis.

I would like to take this opportunity to sincerely thank my parents for their unchanging conviction in my skills, which has been a constant source of inspiration and motivation for me as I have pursued my academic goals. Their unending love, constant support, and innumerable sacrifices have not only motivated me but also given me the crucial groundwork that has allowed me to firmly establish my academic goals.

Finally, I would like to express my gratitude to my friends, who have supported me through all of my highs and lows by giving me much-needed support, empathy, and time off. Their scholarly and personal assistance has been crucial to the completion of this thesis. This work is not only the result of one person's labor; rather, it is formed and enhanced by the support, faith, and efforts of all these people.

Développement efficace de l'informatique en périphérie mobile avec l'intégration des drones

Talaye SANATKAR

RÉSUMÉ

L'informatique en périphérie mobile (MEC) a transformé les réseaux mobiles en rapprochant le calcul et le stockage des utilisateurs. Les drones augmentent cette innovation en ajoutant mobilité et adaptabilité, renforçant ainsi les capacités de l'informatique en périphérie. L'intégration des drones avec la technologie des réseaux définis par logiciel (SDN) forme une nouvelle approche pour développer des systèmes MEC. Les drones dans le MEC introduisent des défis et des opportunités uniques, servant de serveurs de périphérie mobiles dans des zones dépourvues d'infrastructures conventionnelles ou connaissant des pics de demande soudains. Ce déploiement dynamique optimise l'allocation des ressources et améliore la qualité de service (QoS). Le SDN joue un rôle crucial dans l'intégration des drones, des serveurs MEC et des infrastructures de réseau traditionnelles en séparant les plans de contrôle et de données, permettant ainsi une gestion centralisée et une allocation dynamique des ressources. Cela améliore l'évolutivité, la flexibilité et l'efficacité, garantissant une intégration transparente des réseaux et un routage efficace dans des environnements MEC dynamiques.

ce mémoire de maîtrise présente un cadre complet pour le développement de systèmes MEC efficaces avec l'intégration des drones et du SDN. L'architecture MEC habilitée par drones utilise des véhicules aériens sans pilote (UAV) comme serveurs de périphérie mobiles pour étendre la capacité et la portée de l'infrastructure MEC. Un contrôleur SDN centralisé orchestre dynamiquement le déploiement et l'exploitation des serveurs MEC et des drones en fonction des conditions du réseau en temps réel et des exigences des applications. Des algorithmes avancés d'apprentissage profond sont employés pour prédire et optimiser la QoS, améliorant la précision des modèles pour prévoir des mesures de QoS telles que la latence, le débit et la perte de paquets. Ces modèles identifient des motifs d'entrée complexes pour des prédictions précises et une optimisation de la QoS. Les méthodes d'apprentissage profond optimisent également dynamiquement l'allocation des ressources et le routage à l'aide des métriques de QoS prédites, garantissant ainsi le respect constant des accords de niveau de service (SLA) et améliorant l'expérience utilisateur dans les environnements MEC dynamiques.

En intégrant l'apprentissage profond, l'intégration des drones, le SDN et l'apprentissage automatique, l'efficacité, l'évolutivité et la robustesse des systèmes MEC sont considérablement améliorées. L'apprentissage profond améliore la précision des prévisions, permet une prise de décision adaptative et soutient les mesures de sécurité proactive, optimisant ainsi les performances et faisant progresser le développement des infrastructures d'informatique en périphérie.

Mots-clés: Informatique en périphérie mobile (MEC) Véhicules aériens sans pilote (UAV) Réseaux définis par logiciel (SDN) Techniques d'apprentissage profond Optimisation

Efficient Mobile Edge Computing Development with Drone Integration

Talaye SANATKAR

ABSTRACT

Mobile Edge Computing (MEC) has revolutionized mobile networks by bringing computation and storage closer to users. The integration of drones enhances this approach by introducing dynamic mobility and flexibility, thereby expanding the capabilities of edge computing. Drones and Software-Defined Networking (SDN) technology integrate intricately to form an innovative MEC system development method. Drones within MEC present both unique challenges and opportunities, acting as mobile edge servers in remote or densely populated areas where conventional infrastructure is lacking, dynamically deploying to areas with sudden demand spikes, optimizing resource allocation, and enhancing quality of service (QoS). SDN is a key part of making drones, MEC servers, and traditional network infrastructure work together. It does this by separating the control and data planes, which allows for centralized management and dynamic resource allocation. This increases scalability, flexibility, and efficiency while facilitating seamless integration with existing networks and enabling efficient routing and traffic management in dynamic MEC environments.

This master's thesis delves into a comprehensive framework for developing efficient MEC systems with drone integration and SDN. Drone-Enabled MEC Architecture This innovative design leverages unmanned aerial vehicles (UAVs) as mobile edge servers to enhance the capacity and reach of MEC infrastructure. SDN-Based Using real-time network circumstances and application needs, a centralized SDN controller dynamically orchestrates the deployment and operation of MEC servers and drones.

Advanced deep learning algorithms are utilized for the prediction and optimization of Quality of Service (QoS). Firstly, enhance the effectiveness of deep learning models by training them to accurately forecast QoS measures, including latency, throughput, and packet loss. Such models can utilize intricate patterns in input characteristics to offer more precise predictions and enhance QoS optimization. Secondly, Apply deep learning methods to enhance quality of service (QoS) by dynamically optimizing resource allocation and routing decisions using projected QoS metrics. Implementing these strategies can guarantee the consistent fulfillment of service level agreements (SLAs), hence improving the user experience in dynamic multi-access edge computing (MEC) environments.

By integrating deep learning into the model, we can improve the effectiveness, scalability, and robustness of mobile edge computing (MEC) systems by incorporating drone integration, software-defined networking (SDN), and machine learning approaches. Deep learning techniques improve prediction accuracy, enable adaptive decision-making, and facilitate proactive security measures, thereby optimizing performance and advancing the development of edge computing infrastructures.

Keywords: Mobile Edge Computing (MEC), unmanned aerial vehicles (UAV) Integration, Software-Defined Networking (SDN), Deep Learning Techniques, Optimization

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
AGA	Adaptive Genetic Algorithms
APSO	Adaptive Particle Swarm Optimization
AACO	Adaptive Ant Colony Optimization
DP	Dynamic Programming
DE	Differential Evolution
GA	Genetic Algorithm
LP	Linear Programming
MILP	Mixed Integer Linear ProgramminG
DE-ACO	Different Evaluation - Ant Colony Optimization
ML	Machine Learning
MEC	Multi-Access Edge Computing
PSO	Particle Swarm Optimization
SDN	Software-defined Network
SA	Simulated Annealing
UAV	Unmanned Aerial Vehicle
VANET	Vehicular ad hoc network

INTRODUCTION

Context and Motivation

With the advent of Mobile Edge Computing (MEC) in recent years, the field of mobile networks has experienced a dramatic change. This paradigm change has completely transformed the provisioning and use of computing and storage resources, especially for applications that demand high bandwidth and low latency.

The incorporation of drones into MEC infrastructure is a revolutionary advancement that enhances the capabilities of edge computing. Drones, also known as unmanned aerial vehicles (UAVs), bring a new level of mobility and flexibility to MEC systems, expanding their capabilities beyond conventional limits. By serving as mobile edge servers, drones have the ability to offer computational resources to users in remote or densely populated areas that may lack traditional infrastructure. Additionally, by strategically deploying them to areas experiencing sudden increases in demand, we can efficiently allocate resources and enhance the quality of service. This integration presents distinct challenges and opportunities, which is the main focus of this thesis.

Software-defined networking (SDN) is the core component of MEC systems. It is responsible for coordinating the communication between drones, MEC servers, and the conventional network infrastructure. Software-defined networking (SDN) separates the control plane from the data plane, allowing for centralized management and the allocation of resources flexibly and responsively. This architectural adaptability not only improves the ability to scale, be flexible, and operate efficiently in MEC contexts but also enables smooth interaction with current network infrastructure, allowing for effective routing and traffic management in dynamic situations.

Deep learning techniques have proven to be highly effective in optimizing MEC systems, especially in the area of predicting and optimizing quality of service (QoS). Through the

utilization of intricate data patterns, deep learning models have the capability to make precise predictions regarding QoS metrics like latency, throughput, and packet loss by utilizing intricate data patterns. As a result, dynamic resource allocation and routing decisions based on projected QoS metrics can be implemented. By maintaining a meticulous approach, we assure the reliable execution of service level agreements (SLAs) and improve the user experience in dynamic MEC environments.

This thesis seeks to investigate the complex advancements in a unique way for developing a Mobile Edge Computing (MEC) system by incorporating drones, Software-Defined Networking (SDN), and deep learning techniques. This project aims to improve the efficiency, scalability, and reliability of MEC systems through thorough investigation, experimentation, and assessment. Ultimately, it seeks to further the development of mobile and IoV applications in the era of edge computing.

Challenges

Given the subject matter and the presented introduction, we face three primary challenges:

1. **Integration Complexity:** The integration of drones into mobile edge computing (MEC) infrastructure presents a substantial level of intricacy. Advanced management and coordination techniques are required to efficiently manage and synchronize the movement and activities of drones with conventional MEC servers. Achieving smooth integration among drones, MEC servers, and Software-Defined Networking (SDN) controllers is a significant technical hurdle.
2. **Resource Optimization:** Efficiently allocating and managing resources in dynamic Mobile Edge Computing (MEC) systems that incorporate drones is a complex undertaking. Optimizing computational resources, network bandwidth, and energy usage to meet Quality of Service (QoS) criteria is a challenging task. Maximizing the performance and efficiency

of MEC systems requires addressing the challenge of developing efficient algorithms and mechanisms for dynamic resource allocation and load balancing.

3. **Security and Privacy Concerns:** The incorporation of drones into mobile edge computing (MEC) gives rise to novel security and privacy apprehensions. Mobile edge servers in the form of drones are susceptible to both physical and cyber assaults, which can jeopardize the security and reliability of data. Furthermore, the transmission of confidential information between drones and MEC computers via wireless connections gives rise to privacy concerns. It is crucial to create strong security measures, such as encryption, authentication, and access control, to reduce these dangers and guarantee the security and privacy of MEC systems that incorporate drones.

Research Questions

To improve system performance, the optimization of resource allocation in UAV-based Mobile Edge Computing (MEC) systems entails the effective distribution of computational and communication resources among numerous operations and users, taking into account various restrictions and objectives. This is the fundamental question at the heart of this problem:

- **QR:** What strategies may be employed to maximize resource allocation in UAV-based MEC systems, ensuring a fair distribution of computing workloads while taking into account limitations such as limited energy and processing resources in UAVs, fluctuating network conditions, and changing user demands?

Objectives of the thesis

We can further explore the thesis's objectives by examining the specific purposes and targets that the proposed strategy aims to achieve:

•**SO1:** Efficiently allocating resources within UAV-based MEC systems is a key goal in resource optimization. This entails the dynamic allocation of computing workloads to available unmanned aerial vehicles (UAVs) and edge servers, taking into account parameters such as processing capability, communication bandwidth, and energy limitations. By optimizing resource allocation, the system can achieve the highest possible utilization of resources and improve its overall performance.

•**SO2:** Reducing latency is another primary goal to decrease the delay in job execution and data transfer. Reducing latency is crucial in UAV-based MEC systems to ensure the smooth operation of real-time applications like video streaming, augmented reality, and autonomous navigation. The suggested approach attempts to minimize latency and improve the user experience by optimizing task offloading decisions and resource allocation, resulting in faster task processing and data transmission.

Methodology

The process consists of multiple sequential steps:

- We are defining the optimization problem by taking into account objectives such as minimizing latency, maximizing throughput, and optimizing energy use.
- The hybrid DE-ACO method solved the iterative optimization problem.
- We are performing simulations or experiments to assess the effectiveness of the proposed method.
- To demonstrate the efficacy of the proposed solution, we are comparing the outcomes to establish methodologies or existing ways.

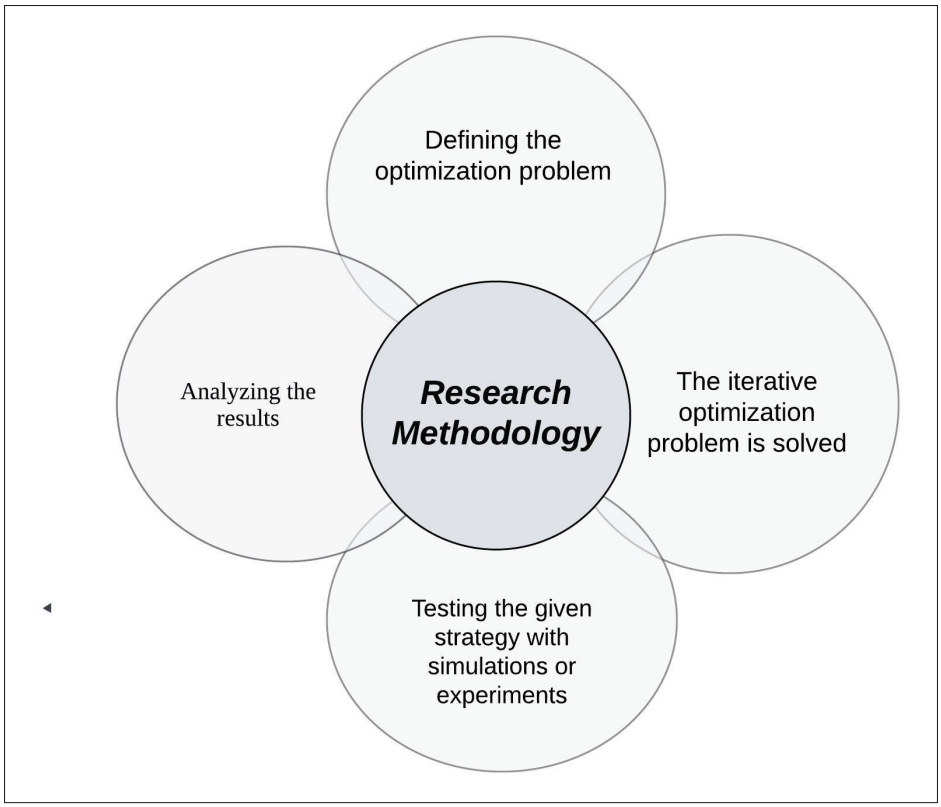


Figure 0.1 Methodology for Efficient Development of MEC with Drone

Contribution

This thesis introduces an innovative application of Adaptive Particle Swarm Optimization (APSO) to optimize the clustering of large-scale car positioning data. By integrating APSO with constraints such as communication range, time thresholds, and energy limits, the research addresses the challenge of efficient clustering in complex real-world scenarios. Comparative analysis with traditional methods demonstrates APSO's superior performance in accuracy and computational efficiency. The findings offer practical insights into optimizing spatial data clustering for applications in urban mobility and logistics, advancing the field of optimization in data-driven decision-making contexts.

Thesis organization

This thesis comprises an introductory section, three main chapters, and a concluding section. The introduction section presents a comprehensive overview of the overall situation and the challenges that require attention. Additionally, it includes the rationales behind this thesis, followed by the specific objectives of the thesis.

In Chapter 1: We review the related tasks of task offloading and energy optimization for the UAV-enabled MEC server. We also review different resource provisioning techniques for task offloading, such as PSO and Ant cloning.

In Chapter 2: We describe the proposed model and research methodology aimed at enhancing the system, ultimately developing an optimal system model to improve the energy efficiency of task offloading.

In Chapters 3 and 4: We focus on reformulating the design of an optimized task offloading orientation problem using two alternative methods of optimization: K-Means and APSO. Initially, we introduce the K-means algorithm for clustering. After that, we use APSO (Adaptive Particle Swarm Optimization) to improve the clustering process and make the centers of each cluster

more accurate so that the chosen area is better covered. Chapter 4 presents the simulation results of each step. We also compare them in terms of fitness versus generation.

The Conclusion The conclusion concisely summarizes the key findings of the thesis and suggests potential directions for future research.

CHAPTER 1

LITERATURE REVIEW

The literature review chapter looks at previous academic and scientific literature. It gives a full summary of what is known now and suggests a way to efficiently distribute resources in UAV-based MEC systems production methods, focusing on more recent approaches. This chapter thoroughly analyzes previous research, pinpoints gaps in the current literature, and lays the theoretical foundation for the subsequent chapters.

Xu, Zhang, Feng, Qin & Xie (2022) the primary focus was on the optimal implementation of Mobile Edge Computing (MEC) using several unmanned aerial vehicles (UAVs), considering cost and energy efficiency. Service providers must consider the limited budget for deploying edge servers before constructing a mobile edge service network. Furthermore, the geographical location of the UAV edge server influences its energy consumption. Drones aided in the creation of a MEC system. The Unmanned Aerial Vehicle (UAV) functions as a portable edge server, delivering computational services to the User Equipment (UE). The goal of this system is to reduce the overall energy usage and deployment expenses required by UAVs to successfully carry out evacuation and hovering functions. The author suggested an enhanced mean shift (IMS) method to concurrently optimize the positioning and quantity of UAV edge servers, thereby enhancing overall energy efficiency and reducing deployment expenses. When compared to alternative deployment approaches, the outcomes of various simulations demonstrated the efficacy of the proposed strategy in reducing energy usage.

Zhang, Zhang, Ma, Yang & Wang (2020) conducted a study on optimizing multi-task scheduling in UAV-assisted MEC to reduce the time it takes to complete tasks. MEC can alleviate the issue of insufficient computational resources for mobile user equipment (UE). Nevertheless, it is possible that a direct communication link between a UE node and a MEC is not available due to either a significant distance between them or the presence of substantial impediments. As a result, it inhibits the loading of tasks in the MEC. Unmanned Aerial Vehicles (UAVs) possess exceptional mobility and can transport compact computing and storage units. This study

presents the UAV-assisted MEC system, which enables a UAV to transmit task input data from user equipment (UE) to a MEC node. Additionally, the UAV can utilize processing and storage resources in the air to reduce the time it takes to complete offloaded tasks. This article aims to collaboratively optimize the UAV's task scheduling and flight path, taking into account the significant order dependence among numerous loaded jobs. Additionally, the author created a heuristic technique using particle swarm optimization (PSO) to efficiently determine the ideal solution. The simulations showed that the suggested multi-task scheduling strategy always finds the best balance between the UAV's position and the conditions of the wireless channel. In comparison to the other three fundamental scheduling strategies, the suggested approach was capable of utilizing minimal execution time to accomplish all the assigned tasks.

Yu & Fan (2022) present a new framework in their research paper that integrates Differential Evolution (DE) with Successive Convex Approximation (SCA) to improve the efficiency of UAV-enabled MEC systems. Their proposed architecture aims to tackle the issues of allocating resources and offloading tasks in dynamic wireless environments. The design aims to optimize the allocation of computational resources and tasks among UAVs and ground users by utilizing Differential Evolution (DE) for global optimization and Simulated Annealing (SCA) for local optimization. The suggested design primarily focuses on optimizing energy usage and reducing latency in MEC systems that involve UAVs. The architecture aims to optimize energy consumption and meet latency limitations for different workloads offloaded to UAVs by employing collaborative optimization using DE and SCA. The study's findings indicate substantial enhancements in system performance when compared to traditional methods. Yu and Fan demonstrate, through simulations and experiments, that their design achieves superior resource allocation, decreased latency, and enhanced energy efficiency in UAV-enabled MEC systems. This work's accomplishment rests in offering a pragmatic and efficient resolution to the optimization obstacles encountered by UAV-enabled MEC systems. Using Differential Evolution (DE) and Simulated Annealing (SCA) together in the architecture makes it strong and reliable for managing resource allocation and task offloading efficiently in wireless settings. To summarize, the suggested design offers a favorable method to improve the efficiency of

UAV-enabled MEC systems. This approach tackles optimization issues by combining the advantages of DE (Differential Evolution) and SCA (Sine Cosine Algorithm), facilitating the effective implementation and functioning of UAVs in mobile edge computing situations.

Xu & Zi (2023) present a novel method that utilizes an enhanced bat algorithm to address this problem. The suggested architecture aims to optimize the allocation of computing workloads between UAVs and ground stations, to improve performance and minimize energy consumption. The proposed architecture by Xu et al. (2023) introduces a new approach that combines an enhanced bat algorithm with compute offloading methodologies for UAVs. This architecture enables flexible decision-making in terms of job distribution, effectively managing computing workloads between UAVs and base stations. The main objective of the proposed architecture is to enhance the efficiency of compute offloading strategies for unmanned aerial vehicles (UAVs). The goal is to improve work allocation, reduce delays, and optimize energy efficiency in UAV operations by utilizing the bat algorithm's characteristics. The study focuses on addressing various significant issues in UAV computing offloading, such as dynamic job allocation, delay reduction, and energy conservation. These difficulties are essential for improving the overall efficiency and practicability of UAV-based operations. Xu et al. (2023) have substantiated the efficacy of their proposed approach through thorough simulations and experiments. When compared to conventional approaches, the findings demonstrate significant improvements in the efficiency of work allocation, reduction in delay, and optimization of energy use. The use of an enhanced bat algorithm in calculating offloading techniques for UAVs shows potential for optimization. The study emphasizes the importance of dynamically allocating tasks and operating efficiently in terms of energy in order to improve the performance of systems that use unmanned aerial vehicles (UAVs).

The study by Mousa & Hussein (2022a) aims to enhance UAV-based mobile edge computing (MEC) by combining Differential Evolution (DE) and Ant Colony Optimization (ACO). The goal is to optimize UAV deployment and task scheduling to reduce computational costs, energy consumption, and task latency in applications like IoT, smart cities, and emergency response. DE is used to find optimal UAV positions by initializing and iteratively improving candidate

solutions through mutation, crossover, and selection. Once UAV positions are optimized, ACO is employed for task scheduling, simulating ant foraging behavior to find the most efficient routes. This integration leverages DE's robustness in continuous optimization and ACO's efficiency in discrete combinatorial optimization. Extensive simulations demonstrate that the DE-ACO algorithm outperforms standalone DE, ACO, and other benchmarks. It shows high efficiency, adaptability to varying UAV numbers and tasks, and significant energy savings. The optimized task scheduling reduces latency and workload on individual UAVs, enhancing system reliability and performance. In conclusion, the DE-ACO algorithm effectively optimizes UAV-based MEC systems, improving computational efficiency, reducing energy consumption, and minimizing task latency, making it a robust solution for modern MEC applications.

Zhen *et al.* (2024) introduce a framework that combines aerial and terrestrial components to improve Mobile Edge Computing (MEC) services. The architecture facilitates the operation of real-time, low-latency applications such as intelligent transportation and connected healthcare by placing Mobile Edge Computing (MEC) servers on both the ground and unmanned aerial vehicles (UAVs) to efficiently handle heavy traffic loads. In densely populated areas like stadiums and tourist destinations, it efficiently allocates resources and provides essential computing services during emergencies due to infrastructure compromise. The challenges are the integration of several MEC platforms through network function virtualization, as well as the optimization of UAV deployment to enable efficient resource utilization. The simulations indicate substantial enhancements in throughput, latency, and user experience, showcasing the architecture's capacity to efficiently adjust to diverse needs.

Huda & Moh (2022) conduct a comprehensive analysis of compute offloading in the context of UAV-enabled mobile edge computing (MEC). Their suggested design promotes the use of UAVs as mobile edge servers, intending to optimize compute offloading in dynamic situations. Their main goal is to analyze and examine the current state of computation offloading techniques, as well as provide insights into the difficulties and possibilities that exist in UAV-MEC systems. Huda and Moh (2022) provide a detailed and precise description of their suggested design, highlighting its ability to efficiently use resources and enhance the overall performance of the

system. The focal point of their investigation revolves around the several challenges faced in computing offloading, such as energy usage, latency, and communication overhead. The authors thoroughly analyze the problems and propose ways to reduce them, presenting practical solutions for efficient offloading in UAV-MEC settings.

Cheng, Liao & Zhai (2020) introduce a new architecture in their research titled "Energy-efficient resource allocation for UAV-empowered mobile edge computing systems." The design concentrates on enhancing resource allocation in UAV-enhanced mobile edge computing systems. Their research aims to address the energy efficiency issues inherent in such systems, where UAVs play a critical role in transferring computational jobs from devices with limited resources to the edge. The authors acknowledge the difficulty of maintaining a harmonious equilibrium between energy usage and computing efficiency in ever-changing contexts. In their proposed architecture, they address the challenge by dynamically allocating resources depending on task characteristics and the movement patterns of unmanned aerial vehicles (UAVs). Their strategy focuses on reducing energy consumption while meeting computational requirements, thereby enhancing the system's overall efficiency. The results indicate substantial enhancements in energy efficiency when compared to conventional methods, hence confirming the effectiveness of their suggested remedy. This accomplishment signifies progress in improving the allocation of resources in mobile edge computing systems provided by unmanned aerial vehicles (UAVs). It holds the potential for creating sustainable and efficient edge computing infrastructures. Cheng, Liao, and Zhai's research highlights the significance of energy-efficient resource management in new paradigms such as mobile edge computing. This research sets the stage for more resilient and sustainable deployments in the future (Cheng et al., 2020).

Chen *et al.* (2022) presented HNIO, a cutting-edge hybrid nature-inspired optimization algorithm tailored for energy minimization in the realm of UAV-assisted mobile edge computing (MEC). The proposed architecture incorporates various nature-inspired optimization techniques to address the intricate optimization problem in this field. Harnessing the power of different algorithms, HNIO strives to optimize resource allocation and reduce energy usage. HNIO's goal is to address the challenges of energy management in UAV-assisted MEC systems. In

these systems, the choices made regarding resource allocation have a direct impact on energy consumption and the overall performance of the system. Conventional optimization techniques may not be well-suited for dynamic and resource-limited settings such as UAV-assisted MEC. This necessitates the development of specialized algorithms such as HNIO. HNIO's primary goal is to address the issue of resource allocation in UAV-assisted MEC systems. HNIO optimizes the use of computing, storage, and communication resources to reduce energy consumption and meet performance requirements. This necessitates optimizing resource allocation decisions by taking into account a variety of factors such as task characteristics, network conditions, and energy constraints. The experiments conducted by Chen et al. provide evidence of the efficiency of HNIO in reducing energy usage while still achieving performance goals in UAV-assisted MEC scenarios. In comparison to existing optimization approaches, HNIO demonstrates significant enhancements in energy efficiency and system performance metrics. HNIO's success stems from its adept management of trade-offs between energy minimization and system performance optimization in UAV-assisted MEC environments. HNIO presents a flexible solution that can adapt to a wide range of optimization objectives and system dynamics by utilizing a hybrid approach that incorporates various nature-inspired optimization techniques. Chen et al. (2022) highlight the potential of HNIO as an effective method for minimizing energy consumption in UAV-assisted MEC systems. The hybrid nature-inspired optimization algorithm showcases exceptional performance when compared to traditional optimization methods, providing a scalable and efficient solution for resource allocation in dynamic and resource-constrained environments.

In vehicular edge computing, task offloading is a critical field that focuses on optimizing resource consumption and improving system performance. Alqarni, Mousa & Hussein (2022) propose an innovative method that employs GPU-based particle swarm optimization (PSO) to tackle this problem. The suggested architecture utilizes the parallel processing capacity of GPUs to effectively optimize task offloading decisions in real-time situations. This study aims to improve task offloading's effectiveness in automotive edge computing environments by utilizing GPUs' computational capabilities and PSO's optimization features. This approach aims to

reduce latency, optimize resource use, and improve overall system performance, according to the researchers. The suggested architecture aims to tackle the dynamic nature of vehicle environments, where prompt task offloading decisions are necessary to adapt to changing conditions. Conventional methods frequently face difficulties in efficiently adjusting to dynamic settings. The experimental results indicate substantial enhancements in reducing latency and optimizing resource use as compared to traditional approaches. The PSO method, which utilizes the GPU, efficiently optimizes decisions regarding job offloading. This results in improved system efficiency and responsiveness. This study's achievement is the efficient use of GPU-based particle swarm optimization (PSO) for optimizing job offloading in vehicular edge computing. It effectively demonstrates its efficacy in real-world situations. The design provides a flexible and effective method for handling computational tasks in ever-changing automotive contexts. Alqarni et al. (2022) propose a suitable method for optimizing work offloading in-vehicle edge computing using GPU-based particle swarm optimization (PSO). The study emphasizes the potential of using advanced optimization strategies to tackle the difficulties presented by dynamic environments, ultimately enhancing system efficiency and performance.

Adnan, Zukarnain & Amodu (2024) thoroughly examine the complex terrain of UAV-enabled Mobile Edge Computing (MEC) systems, offering a comprehensive analysis of existing literature. Their research presents a detailed plan for incorporating UAVs into MEC frameworks, highlighting the advantages and difficulties that come with this novel method. Their review aims to scrutinize existing models and deployment strategies for UAV-enabled MEC systems, focusing on crucial design elements and potential areas for improvement. By synthesizing existing studies, they provide valuable insights into the possible applications and consequences of this developing technology. Adnan et al. (2024) have identified several obstacles that impede the smooth incorporation of UAVs into MEC systems, including constraints on battery capacity, delays in communication, and problems with allocating resources. These challenges highlight the necessity for innovative solutions and optimization strategies to improve the performance and reliability of the system. The authors provide a comprehensive analysis of the accomplishments and constraints of current UAV-enabled MEC models in terms of outcomes. They analyze

the effectiveness of various architectural designs and emphasize successful implementations while also recognizing areas that require further research. The analysis highlights notable advancements in the field, including greater data offloading capabilities, improved coverage and connection, and efficient resource management. These developments create opportunities for revolutionary applications in various areas, including disaster management and precision agriculture. Adnan, Zukarnain, and Amodu (2024) assert that UAV-enabled MEC systems have significant potential but acknowledge the necessity for further study to tackle current obstacles. They promote interdisciplinary collaboration and innovation to fully exploit the advantages of this technology, paving the path for a future where UAVs play a crucial role in expanding computing capabilities to the edge.

In the quest for energy-efficient transmission strategies within Mobile Edge Computing (MEC) networks, Zhao *et al.* (2020) present a notable contribution with their study focusing on UAV-based patrol inspection systems. The proposed architecture ingeniously integrates UAV technology with MEC, promising enhanced efficiency in surveillance tasks. Their goal is to address the pressing need for energy conservation in such systems without compromising performance. By leveraging MEC capabilities, the authors aim to optimize data transmission, processing, and storage at the network edge, thus mitigating energy consumption concerns. However, the deployment of UAVs in surveillance applications poses unique challenges, including limited onboard power and computational resources. Yang et al. delve into these problems, striving to devise transmission strategies that balance energy efficiency with task completion within stringent operational constraints. Their research yields promising results, demonstrating substantial improvements in energy efficiency compared to traditional methods. They demonstrate the efficacy of their proposed strategies in reducing energy consumption while maintaining satisfactory system performance through simulations or empirical analysis. Yang et al.'s achievement lies in their successful integration of MEC with UAV-based surveillance, which offers a viable solution to the energy efficiency conundrum in patrol inspection systems. Their work opens avenues for further exploration and refinement in optimizing resource utilization in similar applications.

El-Emary, Ranjha, Naboulsi & Stanica (2023) conducted a thorough investigation on data transfer and path design in UAV-based Mobile Edge Computing (MEC) systems to enhance energy efficiency. The proposed framework incorporates unmanned aerial vehicles (UAVs), leveraging their agility and processing capacities to improve mobile edge computing (MEC) operations. The study aims to reduce energy usage while ensuring efficient task execution, and it addresses the key issues faced by these systems. The authors highlight significant challenges related to energy usage in UAV-based MEC, such as ever-changing user needs, resource restrictions, and UAV mobility constraints. Their research aims to develop solutions that effectively balance these elements, resulting in optimal performance while also preserving energy resources. The study's findings reveal substantial advancements in enhancing energy efficiency. The proposed approach efficiently decreases energy consumption without affecting the quality of service by strategically assigning jobs to UAVs and optimizing their trajectories. The authors present quantitative evidence to demonstrate the effectiveness of their methods through simulations and experiments. The research findings highlight the practicality and efficacy of utilizing UAVs in MEC systems to improve energy efficiency. The suggested design presents a hopeful resolution to the energy obstacles encountered by contemporary computer systems, especially in situations where mobility and dynamic resource allocation are of utmost importance.

Table 1.1 Literature Review for Efficient Development of MEC with Drone

Number	Author		Type of Data	Methodology	Propose	Result
1	Mohamed El-Emary	2023	Simulation-based	Energy-efficient task offloading and trajectory planning for UAV based MEC systems using optimization methods.	Optimizing UAV-based MEC energy efficiency through task offloading and trajectory optimization.	Optimized task offloading , trajectory design improve UAV-based MEC energy economy and performance.
2	Adnan, Zukarnain	2024	Random simulated environment	Thorough analysis of models, difficulties, and prospects.	Provide UAV-enabled MEC system design insights.	framework focusing on fundamental design aspects of UAV-enabled MEC systems.
3	Alqarni, Mousa	2022	Random simulated environment	leveraging GPU acceleration	simulation-based experimentation using GPU-based particle swarm optimization	an optimized task offloading scheme, resulting in improved performance metrics.
4	Chen et al	2022	Random simulated environment	ature-inspired optimization for energy minimization in UAV-enabled mobile edge computing.	HNIO, a hybrid nature inspired optimization algorithm	minimize energy consumption in UAV-assisted mobile edge computing
5	Cheng, Y.	2023	Random simulated environment	optimizes resource allocation for UAV-enabled mobile edge computing,	utilizing data on Utility and Cloud Computing	optimize resource utilization and enhance system performance
6	El-Emary et al	2023	Random simulated environment	simulation-based analysis.	an energy-efficient approach	demonstrate improved performance
7	Mousa et al	2022	Random simulated environment	UAV-based mobile edge computing using ant colony optimization.	MEC efficiency through ant optimization for task delay.	The proposed strategy improves UAV-based MEC performance and efficiency.
8	Xu, F. & Zi.	2023	Random simulated environment	Improved bat algorithm	offloading strategy for UAVs using an enhanced bat algorithm	improved efficiency in cognitive robotics applications
9	Yu and Fan	2022	Random simulated environment	Differential Evolution and Successive Convex Approximation for optimizing UAV-enabled Mobile Edge Computing	a novel methodology to enhance resource allocation	showing mobile edge network task offloading and resource allocation efficiency improvements.
10	Zhang et al.	2020	Random simulated environment	using mobile edge computing with UAVs	an efficient multitask scheduling aiming to minimize completion time	indicating improved task completion efficiency
11	Zhao et al.	2020	Random simulated environment	utilizing simulation-based data analysis and a mixed integer linear programming (MILP) model	a novel cost optimization for SDN-enabled UAV assisted vehicular computation offloading	enhanced computational efficiency and reduced operational costs.

This table provides a literature review highlights the significant advancements in optimizing UAV-based Mobile Edge Computing (MEC) systems through various simulation-based methodologies. Key approaches, such as energy-efficient task offloading, trajectory optimization, and the application of advanced algorithms like particle swarm optimization and hybrid nature-inspired techniques, have consistently demonstrated improved energy efficiency, resource utilization, and overall system performance. These findings underscore the potential of innovative optimization strategies in enhancing the effectiveness of UAV-enabled MEC systems, paving the way for more efficient and reliable deployments in the future.

CHAPTER 2

METHODOLOGY

The metropolitan environment features unmanned aerial vehicles (UAVs) deployed to monitor the airspace above the busy cityscape. These airborne cars maintain a stable location by hovering over a network of Internet of Vehicles (IoV) and self-driving vehicles that are traversing the urban environment. We propose a novel multi-UAV-enabled mobile edge computing (MEC) architecture, where multiple UAVs provide both communication and computation services for IoV devices and autonomous vehicles that cannot directly access the ground-edge clouds. To get the best task offloading between UAVs, we use a adaptive particle swarm optimization (APSO) algorithm to jointly optimize decisions about which tasks to split, how to communicate and compute resources to be used, and the positions of the UAVs. We do this while also making sure that all tasks' delay requirements are met (Zhang, Gong & Guo (2024)).

APSO allows for dynamic and efficient optimization of task offloading decisions, resource allocation, and UAV positioning in a complex urban environment. By adapting the PSO algorithm to the specific needs of the network, APSO ensures that tasks are processed with minimal delay, enhancing the overall performance and responsiveness of the UAV-based system. This approach is particularly beneficial in scenarios where rapid changes in network conditions and varying computational demands require a flexible and adaptive solution to maintain optimal system performance.

2.1 Component and Infrastructure

Software-Defined Networking (SDN) and Multi-Access Edge Computing (MEC) are advanced technologies that focus on different areas of contemporary network architecture and functionality. Software-defined networking (SDN)(Yan, Yu, Gong & Li (2015)) is a cutting-edge method that enables intelligent and unified control of networks using software applications. The main objective of Software-Defined Networking (SDN) is to offer a more adaptable, effective, and expandable method of controlling and enhancing network resources. Separating the control plane, responsible for traffic orientation decisions, from the data plane, responsible for packet

forwarding, accomplishes this. The SDN controller is a central software entity that supervises the control plane and configures data plane devices. This segregation enables efficient and automated network administration, improves resource allocation efficiency, and streamlines the processes of configuring and resolving issues. SDN is highly beneficial for network virtualization, dynamic traffic control, the implementation of security protocols, and the optimization of network resource allocation.

On the other hand, multi-access edge computing (MEC)(Mao, You, Zhang, Huang & Letaief (2017)) extends the power of cloud computing to the network edge, which is closer to end users. The main goal of MEC is to decrease latency and optimize instantaneous interpreting, thereby enhancing the user experience. MEC reduces the necessity for data to travel to centralized cloud data centers by placing computing resources closer to the data source. Edge servers are essential elements of MEC as they offer computing, storage, and networking functions. The applications running on these edge servers carry out operations locally, resulting in a substantial reduction in latency. MEC provides support for many low-latency applications, such as autonomous driving, Internet of Things (IoT) services, augmented reality, and real-time analytics. These applications demand quick data processing and little delay.

The core distinctions between SDN and MEC lie in their primary objectives and methods of execution. Software-defined networking (SDN) prioritizes network management and enhancement by separating the control and data planes. This enables centralized control and the ability to program the network. MEC focuses on putting compute and storage closer to customers in order to decrease latency and improve real-time processing capabilities. To optimize network control, SDN concentrates its resources in a central location, while MEC distributes its resources to the network's edge for applications that demand low latency. The main advantage of SDN is its ability to offer centrally managed, configurable network management and efficiency. On the other hand, MEC improves throughput by processing data at the edge of the network.

In this thesis, we utilized the combination of UAV and MEC, which is a unique integration of two modern technologies that improve computational capabilities and network performance. The term UAV-enabled Mobile Edge Computing (MEC) describes the use of UAVs, commonly

referred to as drones, equipped with edge computing capabilities. We deploy these UAVs to provide computing, storage, and networking services close to end users or devices. This configuration utilizes the mobility and adaptability of UAVs to introduce edge computing to regions that have little or no permanent infrastructure. As a result, it greatly enhances the provision of services in different situations.

The integration of UAVs with MEC is particularly beneficial in situations where conventional infrastructure is either unfeasible or too expensive (Sun, Wan & Wang (2021)). For instance, disaster-affected areas can promptly deploy UAVs to establish temporary communication networks and provide essential computational resources. This helps facilitate emergency response and recovery operations. UAV-enabled MEC can be used in areas with limited network infrastructure, such as remote, rural locations or metropolises, to provide and manage traffic and communication. This helps to close the gap in access to digital resources and improve the overall quality of life.

An important benefit of UAV-enabled Mobile Edge Computing (MEC) is its capacity to flexibly adjust to evolving network requirements and environmental circumstances. UAVs have the ability to quickly and efficiently adjust their settings and positions in order to maximize network coverage, minimize delays, and distribute computational tasks evenly. The flexibility of edge computing means that resources are accessible at the exact location and time they are required, hence improving the efficiency of service delivery. Furthermore, UAVs have the capability to facilitate various tasks, such as real-time data analysis, streaming of videos, augmented reality, and IoV services. These applications greatly benefit from the decreased delay and enhanced computational capabilities offered by edge computing (Yao, Ma, Wang, Chen & Xu (2024)).

Moreover, the use of UAVs in Mobile Edge Computing (MEC) has notable implications for improving the efficiency and expandability of future networks. By transferring processing jobs from centralized data centers to UAVs, network operators can decrease congestion, decrease operational costs, and enhance overall network resiliency. By processing sensitive data directly on the UAVs, the decentralized nature of this computing architecture enhances data privacy and security, eliminating the need to transmit data over potentially vulnerable networks.

2.2 Task offloading optimization methods

Energy offloading optimization is the efficient allocation of computational tasks between local devices and remote servers to minimize energy usage while still achieving performance criteria. Below are several highly effective methods used for energy offloading:

2.2.1 Exact Method

Exact approaches refer to improvement strategies that strive to identify the optimal solution to a given problem within a certain timeframe. These methods guarantee a mathematical demonstration of the obtained solution as the optimal solution, taking into account the problem's constraints and the objective function. Here is a more comprehensive exploration of several notable, precise techniques:

•**Linear Programming (LP):** Linear programming entails optimizing a linear objective function by linear equality and inequality constraints. It is suitable for continuous decision variables (Dantzig (2002)).

•**Mixed Integer Linear Programming (MILP):** Mixed Integer Linear Programming (MILP) deals with optimization problems that include both integer and continuous variables, as well as linear relationships. It is commonly employed in situations involving the allocation of resources, scheduling, and routing, where decisions are made in a discrete manner (Floudas & Lin (2005)).

•**Dynamic Programming (DP):** Dynamic programming decomposes a problem into smaller subproblems and solves them iteratively. It is highly effective for issues with overlapping subproblems and an optimal substructure (Denardo (2012)).

These exact methods provide rigorous approaches for solving optimization problems with guaranteed optimality, making them indispensable in various fields where precise solutions are required. However, the size, complexity, and available computational resources may limit their applicability.

2.2.2 Heuristic Method

Heuristics are problem-specific techniques that employ intuitive rules or informed estimations to efficiently identify satisfactory solutions. Heuristic approaches are typically more straightforward and quicker than exact procedures, although they may not always discover the ideal solution. Common heuristic methods are:

•**Greedy Algorithms:** Make decisions that are best for the current situation at each phase, aiming to eventually find the best possible outcome overall. Assign tasks quickly based on immediate criteria, such as the server that is closest to you or the unmanned aerial vehicle (UAV) that has the most remaining battery capacity (Azizi, Shojafar, Abawajy & Buyya (2022)).

•**Local Search:** Commence with an original answer and progressively make incremental modifications to enhance it. Optimize task allocations to enhance overall performance, such as minimizing delay or distributing load evenly.

•**Rule-Based Systems:** Utilize pre-established guidelines to direct the process of making decisions. Develop and enforce targeted protocols for transferring tasks to unmanned aerial vehicles (UAVs) based on their capacities and the state of the network.

2.2.3 Meta-heuristic Methods

Meta-heuristics are advanced algorithms specifically developed to systematically examine the search space more comprehensively compared to traditional heuristics. They frequently employ various tactics to achieve a balance between exploration, which involves doing a wide search, and exploitation, which involves conducting a thorough search.

•**Particle Swarm Optimization (PSO)**: PSO is a simulation that models the collective behavior of birds flocking or fish schooling. Potential solutions, represented as particles, move through the solution space, shaped by their individual experiences and those of their neighboring particles. We aim to enhance the efficiency of task offloading by strategically rearranging tasks and resources to create configurations that reduce latency and energy consumption (Wang, Tan & Liu (2018)).

•**Ant Colony Optimization (ACO):** Ants' foraging activity forms the basis of ACO. Artificial

ants build solutions by following pheromone trails, which change based on the quality of the solutions. Simulating the behavior of ants to locate the quickest path to food sources is necessary in order to determine the best paths for data transfer and offloading jobs (Blum (2005)).

•**Simulated Annealing (SA):** Metallurgy's annealing process is the source of SA. A probabilistic mechanism in the algorithm enables it to accept suboptimal solutions, thereby avoiding reliance on local optimal solutions. Over time, the likelihood of accepting poorer solutions decreases steadily. We tackle task scheduling issues where the solution space is marked by challenging terrain, encompassing a multitude of local optimal solutions (Van Laarhoven, Aarts, van Laarhoven & Aarts (1987)).

•**Differential Evolution (DE)**: Differential Evolution (DE) is an optimization technique that uses differential mutation and crossover operations to systematically search the solution space based on the population. We optimize parameters in unmanned aerial vehicle (UAV) control systems and scenarios that involve continuous task offloading (Price (2013)).

2.2.4 Hybrid Methods

Hybrid methods integrate components from both exact and heuristic metaheuristic approaches to capitalize on the advantages of each. These methods offer a favorable equilibrium between discovering solutions of high quality and preserving computational efficiency.

•**PSO with GA:** Utilize particle swarm optimization (PSO) to rapidly converge on a favorable region within the solution space and subsequently employ genetic algorithms (GA) to thoroughly explore and refine the solutions (Garg (2016)).

•**SA with MILP:** Utilize Simulated annealing (SA) to obtain a solution that is close to optimal and subsequently enhance it using mixed integer linear programming (MILP) to guarantee accuracy and compliance with constraints.

•**ACO with DE:** Utilize Ant Colony Optimization (ACO) for the purpose of routing, and Differential Evolution (DE) for parameter optimization. This approach combines both discrete

and continuous optimization approaches (Duan, Yu, Zhang & Shao (2010)).

2.2.5 Adaptive Algorithms:

Adaptive algorithms are a type of optimization technique that has the ability to modify its parameters and tactics in response to real-time feedback and changing environmental variables. These methods are especially useful in intricate and ever-changing systems like UAV task offloading, where factors such as network bandwidth, computational load, and UAV battery levels can rapidly fluctuate.

•**Adaptive Genetic Algorithms (AGA):** As the evolution process unfolds, AGAs, or adaptive genetic algorithms, adjust parameters like crossover and mutation rates in response to the population's performance (Deb & Beyer (2001)).
Task Scheduling: Modify mutation rates in real time to prevent early convergence and maintain a wide range of scheduling solutions. Resource Allocation: Utilize the adaptive crossover technique to investigate novel allocation options according to the present system load.

•**Adaptive Particle Swarm Optimization (APSO):** APSO adjusts the inertia weight and learning variables in the PSO algorithm based on the swarm's performance. The aims of employing this approach are : Path planning entails adjusting the velocities of particles to effectively navigate around moving obstacles or reach desired destinations. Task offloading refers to the process of modifying learning parameters in order to achieve dynamic load distribution, taking into account the present conditions of the network (Zhan, Zhang, Li & Chung (2009b)).

•**Adaptive Ant Colony Optimization (AACO):** The AACO algorithm modifies the rate at which pheromones vanish and the techniques ants use to explore based on the effectiveness of the solution paths. One of the purposes of using this method is: Routing (Adjust pheromone levels to prioritize successful routes in a network architecture that is always changing.) Task Assignment Modify exploration strategies to efficiently allocate work based on varying computing loads (Ping, Chunbo, Yi, Jing & Yanqing (2014)).

2.2.6 Machine Learning Integration

Machine learning (ML) integration involves using ML models to enhance the performance and adaptability of optimization algorithms in UAV task offloading.

1. Supervised Learning 2. Unsupervised Learning 3. Reinforcement Learning (RL)

2.3 Proposed Method of Optimization

Adaptive particle swarm optimization (APSO) is an advanced technique that enhances computing efficiency, reduces latency, and improves overall network performance in vehicular networks through task offloading among cars, UAV-enabled mobile edge computing (MEC), and software-defined networking (SDN). This integration leverages the benefits of all technologies to dynamically oversee and allocate computing activities, ensuring smooth and effective operations. The offloading process commences when the vehicle's internal system identifies the requirement for extra computational resources, which may be prompted by intricate driving situations necessitating real-time data processing that surpasses the vehicle's capabilities. The vehicle then transmits a request to offload a job to the closest Mobile Edge Computing (MEC) platform equipped with Unmanned Aerial Vehicle (UAV) capabilities. APSO is utilized to assess various potential offloading alternatives, considering factors such as computational capacity, current workload, and network latency. APSO algorithms adaptively optimize and choose the most suitable UAV for task execution, guaranteeing effective usage of resources.

SDN controllers are essential for overseeing the network infrastructure. They route the task request over the most effective network pathway and guarantee that the connection between the automobile and the UAV has been optimized for minimal delay and maximum dependability. The SDN controller facilitates the required network configurations to enable the job offloading procedure overseen by APSO.

Upon receiving the task request, the UAV-enabled MEC platform, directed by APSO and K-Means, evaluates its processing capabilities and present workload. If the necessary resources are accessible, the UAV will accept the assigned assignment and carry out the processing on-site. Performing computations locally greatly reduces the amount of time it takes for tasks to be completed in comparison to transferring them to remote cloud data centers. The task results are communicated back to the car through the SDN-managed network, guaranteeing that the vehicle receives immediate computational assistance. The prompt exchange of input is crucial for applications like autonomous driving since any delay in reactions might have an adverse effect on safety and performance.

The APSO and K-Means algorithms employ continuous monitoring and adaptation to dynamically adjust to variations in network conditions and compute workloads, thereby maximizing the real-time allocation of tasks. SDN is a technology that automatically adapts network regulations and allocates resources in response to changing demands. In situations where there is heavy traffic and numerous vehicles are performing activities at the same time, the SDN controller, with the help of APSO and K-Means, gives priority to important tasks and assigns network bandwidth accordingly. This ensures that key tasks have the resources they need without causing the network to become overloaded.

In addition, the use of SDN with Unmanned Aerial Vehicle (UAV)-enabled MEC and k-Means APSO improves the system's ability to scale and adjust. SDN controllers may dynamically adjust network topologies and resource allocations in response to changing traffic patterns and computing requirements. This enables the network to consistently achieve optimal performance. UAVs can be moved to different positions in response to real-time data analysis from the SDN controller and autonomous path planning and scheduling optimization systems. This ensures that the UAVs are strategically placed to effectively support vehicular networks.

This section outlines our proposed optimization technique. Initially, we suggest dividing the collection of wireless devices, D, by employing the K-means clustering technique, taking into account the communication range (R_c) of the UAV. Furthermore, the adaptive PSO algorithm is employed to optimize the clustering and determine the most efficient path for the UAV. We suggest the incorporation of a tailored adaptive particle swarm optimization (PSO) algorithm. The customized adaptive PSO is comparable in simplicity to the traditional PSO, although it demonstrates superior performance. Our objective is to improve computing efficiency and establish a more advanced regional interaction architecture between the particles in the community, which will be integrated into the global search process. The algorithm's adaptive methodology enables it to dynamically modify parameters in order to achieve better convergence and accuracy. The enhanced communication topology improves the ability to search locally while also maintaining a strong ability to search globally. This allows for more efficient exploration of search spaces with many dimensions, employing a large population swarm.

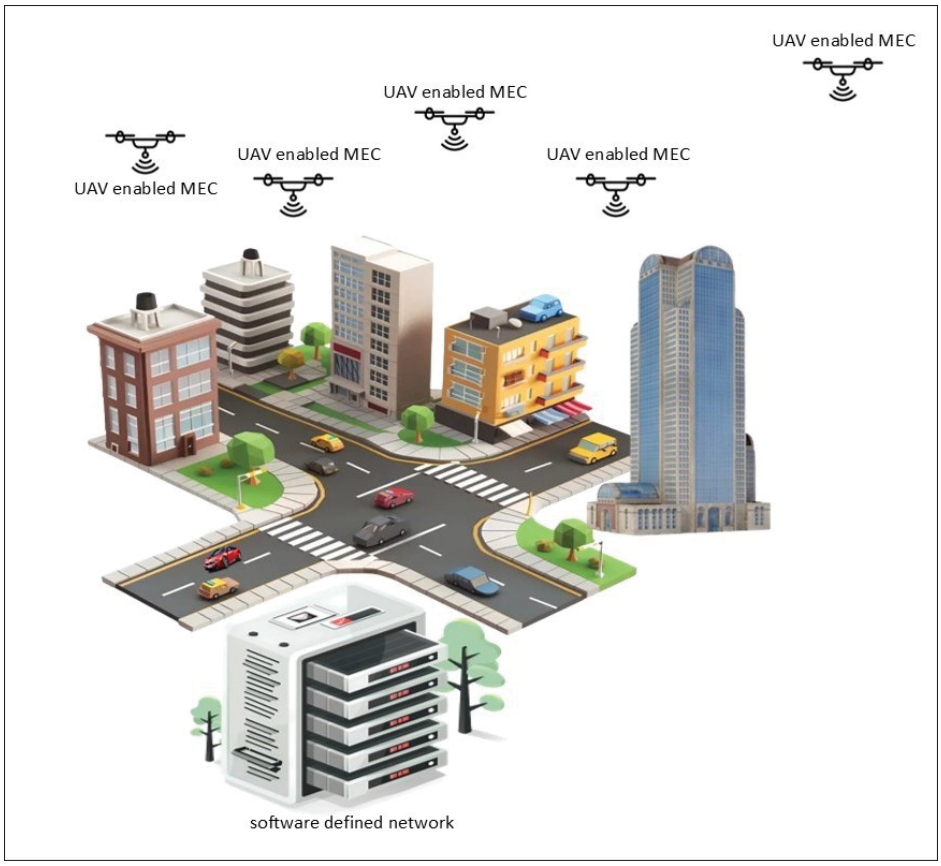
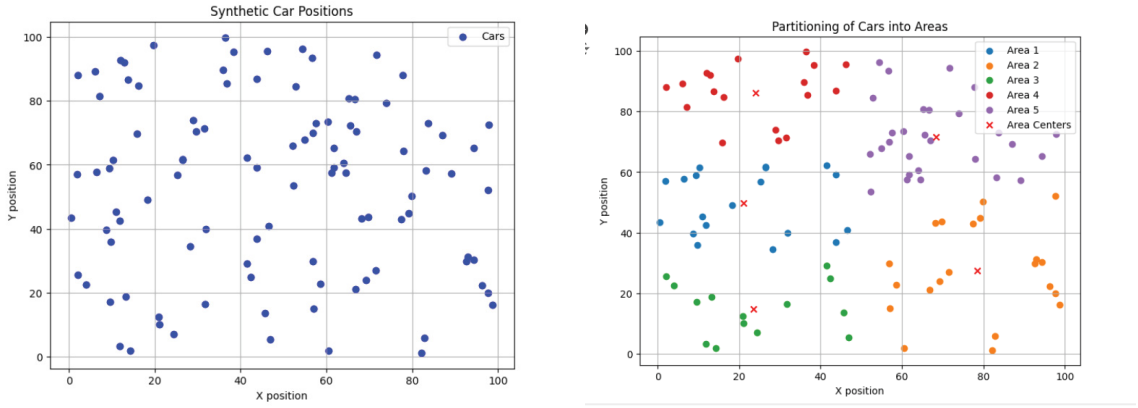


Figure 2.1 System Model

2.3.1 K-means clustering

K-means clustering (Kodinariya, Makwana *et al.* (2013)) divide a two-dimensional plane that has a collection of N anchor points into separate sections, referred to as centroids, into K clusters. Every cluster is linked to a distinct centroid (refer to figure 2a), and any data points within that region are assigned to the nearest adjacent to the designated centroid. Additionally, the points within a cluster form a distinct region around the centroid (see figure. 2b). The midpoint between centroids determines the boundaries between these regions, assigning each point to the nearest centroid. Unlike Voronoi diagrams, K-means clustering does not form rigid polygons but rather flexible regions that can adapt based on the distribution of data points.



a) Dataset without clustering

b) Clustering dataset with K-means

Figure 2.2 k-means Simulation for 100 vehicles in 5 regions

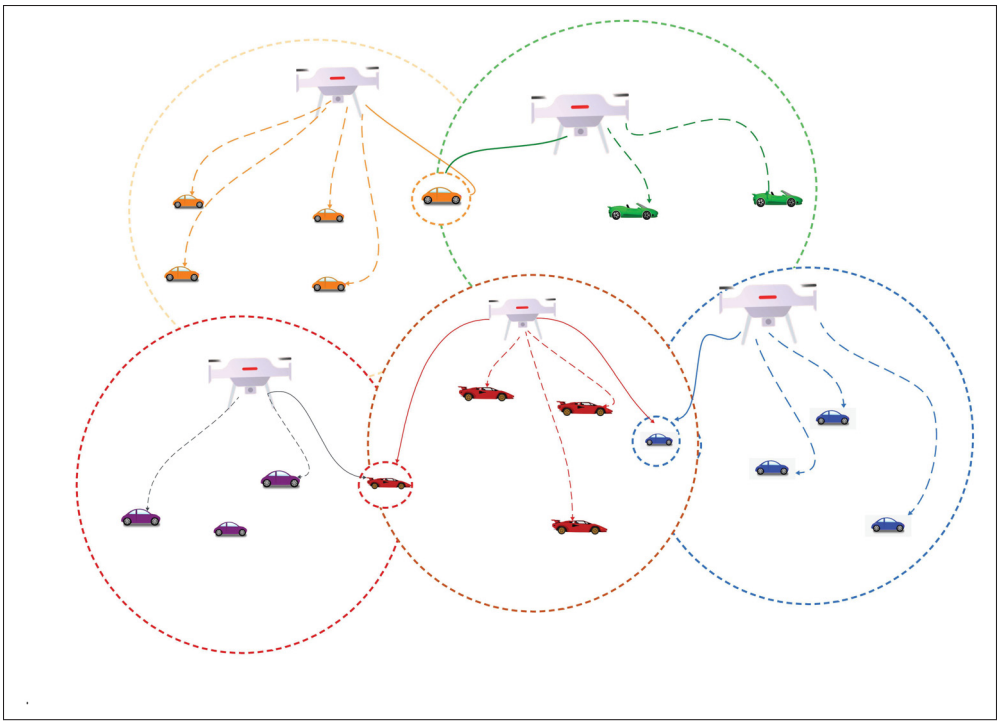


Figure 2.3 System Model Partitioning Problem

Assuming that there is a collection of vehicle D in an area R, with $D = \{\mathbf{d}_1, ..., \mathbf{d}_N\}$, K-means divides the region R into a set of cluster. Considered an offloading region to be visited by the UAV, each cluster comprises a single mobile device. With the understanding that any generated area must meet the following, we demonstrate how to combine these regions in order to offload numerous devices in a single visit.

1. It is imperative that all linked mobile devices in the vicinity fall within the UAV's coverage area, Rc.
2. According to limitation C2, the total time spent transmitting and computing by the designated devices in the area must be less than the maximum permissible duration for flying (T_u)

Algorithm 1 explains the procedure for the preliminary partitioning of D. The algorithm requires as input the set of vehicles D. The algorithm commences by randomly selecting k centroids from the dataset D and initializing them. Subsequently, each device d_i is allocated to the centroid that is closest to it, so establishing the initial clusters. The centroids are recalculated as the average of the devices inside their corresponding clusters. The process of assigning data points to centroids and updating the centroids is repeated until convergence, which is defined as the centroids changing by an amount that is smaller than a predefined tolerance or after reaching a maximum number of iterations. The following text provides a detailed explanation of the sequential actions involved in the algorithmic process.

1. Input: A collection of portable electronic gadgets The set D is defined as the collection of elements $D = \{d_1, d_2, \ldots, d_N\}$. The set D consists of elements $C = \{c_1, c_2, \ldots, c_N\}$ and we have a total of k clusters.
2. initialize Centroids: Choose k machines from D as the first centroids.
3. Repeat Until Convergence:

    - Assign Devices to Clusters: For each device d_i in the set D, calculate the Euclidean distance between d_i and each centroid c_j. Assign the data point d_i to the cluster C_j whose centroid c_j is closest to d_i.
    - Update Centroids: For each cluster c_j compute the new centroid c_j as the mean of all devices assigned to C_j.

Algorithm 2.1 K-Means Partitioning Algorithm

Input: A set of mobile devices $D = \{d_1, d_2, \ldots, d_N\}$, number of clusters k
Output: The set of clusters $\{C_1, C_2, \ldots, C_k\}$ and their centroids $\{c_1, c_2, \ldots, c_k\}$

Initialize centroids: Randomly select k devices from D as initial centroids $C = \{c_1, c_2, \ldots, c_k\}$
Initialize previous_centroids ← `None`
Initialize max_iterations ← 100
Initialize tolerance ← 10^{-4}
for iteration ← 1 to max_iterations: Assign devices to clusters: **do**
    for each device $d_i \in D$ **do**
        Compute the Euclidean distance between d_i and each centroid c_j Assign d_i to the cluster C_j whose centroid c_j is closest to d_i
    end for
end for
Update centroids:
for each cluster C_j **do**
    Compute the new centroid c_j as the mean of all devices assigned to C_j
end for
if previous_centroids is not `None` **then**
    converged ← `True`
    for each centroid c_j **do**
        if converged **then**
            break
        end if
    end for
end if
if previous_centroids ← **copy of** C **then**
end if
Return The set of clusters $\{C_1, C_2, \ldots, C_k\}$ and their centroids $\{c_1, c_2, \ldots, c_k\}$

- Check for Convergence: If the centroids do not change significantly (defined by a tolerance threshold) or the maximum number of iterations is reached, stop the iteration.

4. Output: The set of clusters $C = \{C_1, C_2, \ldots, C_N\}$ and their centroids $C = \{c_1, c_2, \ldots, c_N\}$

The initial partitioning of the area R using the K-means algorithm results in clusters that are formed based on the proximity of devices to the centroids. This method ensures that each cluster contains devices that are closer to their centroid compared to others. The final clusters provide a

partitioning of R that groups devices based on their spatial distribution, facilitating efficient task offloading and resource management.

In fact, the first cluster of the provided devices may result in imbalanced sections, as shown in figure 2.3 While the initial partitions are designed to meet the UAV's time limitations, they may also result in longer transfer times required to complete all operations within each partition. When the UAV is within a designated area, it has the ability to provide coverage for devices located in adjacent partitions. In Figure 2.3, the blue car may be covered by a red zone. The challenge at hand is to determine if placing the device in zone blue or red is more efficient.

In order to address this issue, we employ Adoptive Particle Swarm Optimization (APSO) to establish a connection between the variable "s" and its related region. The goal of this optimization procedure is to reduce the total duration along the path of the unmanned aerial vehicle (UAV), including the time needed for unloading. In the following section, I explain the planned implementation of an adaptive particle swarm optimization (APSO) algorithm. This algorithm aims to find the most efficient combination of shared vehicles and the ideal path for unmanned aerial vehicles (UAVs), to minimize both time and energy consumption.

2.3.2 Adaptive Particle Swarm Optimization (PSO)

Adaptive Particle Swarm Optimization (PSO) is an improved iteration of the conventional Particle Swarm Optimization technique (Zhan, Zhang, Li & Chung (2009a)). PSO, a computational technique, replicates the social behavior of bird flocking or fish schooling to optimize a diverse set of functions. Every individual "particle" within the swarm symbolizes a potential solution. These particles navigate around the search space in order to discover the most ideal solution. They achieve this by modifying their placements according to their personal experiences as well as the experiences of their neighboring particles. Kennedy & Eberhart (1995).

Particle Swarm Optimization (PSO)

- Key Components: Particle Swarm Optimization (PSO) encapsulates a dynamic optimization approach, drawing inspiration from the coordinated movements observed in flocks of birds or schools of fish. In this algorithmic framework, particles, representing potential solutions, navigate a multidimensional search space, continually adapting their positions and velocities. Key to PSO are the notions of personal best (pBest) and global best (gBest), reflecting the individual and collective successes within the swarm. Kennedy and Eberhart's seminal work on PSO elucidates its fundamental concepts, emphasizing the interplay between exploration and exploitation in the pursuit of optimal solutions. Their research, underscored by practical applications in various domains, provides a foundational understanding of PSO's mechanics and its significance in addressing complex optimization challenges.

- Algorithm Steps (Initialization and Iterative Process): Initialize a group of particles by assigning them random places and velocities. Assess the physical condition of every individual particle.

  Assign the initial position of each particle as its first individual best position.

  Determine the optimal global position across each particles. For each particle, update the velocity and position using the following formulas:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t)) \qquad (2.1)$$

  This formula calculates the new velocity for particle i. The velocity is influenced by three components:

  Inertia Weight ($w \cdot v_i(t)$): Maintains the particle's previous velocity, balancing exploration and exploitation.

  Cognitive Component $c_1 \cdot r_1 \cdot (pBest_i - x_i(t))$: Pulls the particle towards its personal best position.

  Social Component $c_2 \cdot r_2 \cdot (gBest - x_i(t))$: Pulls the particle towards the global best position.

Equation for Updating Position:

$$x_i(t+1) = x_i + v_i(t+1) \tag{2.2}$$

Adaptive Particle Swarm Optimization (APSO)

Adaptive Particle Swarm Optimization (APSO) is an improved iteration of the conventional Particle Swarm Optimization (PSO) technique. This adaptive model seeks to enhance the efficiency of the traditional Particle Swarm Optimization (PSO) algorithm by constantly modifying its parameters in response to the current condition of the swarm.

- Main adaptation: Adaptive Particle Swarm Optimization (APSO) improves on the original Particle Swarm Optimization (PSO) by changing key variables like the inertia weight (w) and the acceleration coefficients (c1 and c2) based on how the swarm is acting at the moment. APSO's adaptability allows it to optimize the balance between exploration and exploitation, resulting in improved convergence speed and premature convergence prevention. The program determines the status of the swarm by analyzing measures such as particle diversity and global optimal fitness, and adjusts the variables accordingly. By making these modifications, APSO is able to consistently perform well throughout all stages of the optimization process, resulting in increased efficiency and reliability compared to the conventional PSO with unchanging parameters.
- Algorithm below presents a step-by-step process for implementing APSO in a clustering context. The algorithm begins by defining input parameters such as the number of cars, clusters, communication radius, and various thresholds. It then outlines the initialization steps, including generating synthetic data and initializing particles, velocities, and best positions. The main loop of the algorithm iterates through a specified number of iterations, updating particle velocities and positions, evaluating fitness, and updating personal and global best scores. The algorithm concludes by returning the global best position, score, and fitness over time. This pseudocode provides a comprehensive overview of how APSO can be applied to solve clustering problems.

Algorithm 2.2 Adaptive Particle Swarm Optimization (APSO) for Clustering

Input: N: Number of cars, k: Number of clusters, Rc: Communication radius, Tu: Time threshold, Eu: Energy threshold, γ: Penalty coefficient, max_iters: Maximum iterations, $swarm\_size$: Number of particles, w: Inertia weight, $c1, c2$: Acceleration coefficients
Output: Best centroids and best fitness score

Generate synthetic data for car positions: cars_data $\leftarrow$ Uniform$(0, 1000, (N, 1))$
Initialize particles: particles $\leftarrow$ Uniform$(0, 100, (swarm_size, k, 2))$
Initialize velocities: velocities $\leftarrow$ Uniform$(-1, 1, (swarm_size, k, 2))$
Initialize personal best positions: personal_best_positions $\leftarrow$ particles
Initialize personal best scores:
  personal_best_scores $\leftarrow$ {fitness_function$(p$, cars_data$)$ | $p \in$ particles}
Initialize global best position:
global_best_position $\leftarrow$ personal_best_positions $[\arg\min($personal_best_scores$)]$
Initialize global best score: global_best_score $\leftarrow \min($personal_best_scores$)$
Initialize fitness over time: fitness_over_time $\leftarrow$ [global_best_score]
for each iteration in 1 to max_iters **do**
    for each particle i in 1 to $swarm\_size$ **do**
        Generate random numbers $r1, r2 \leftarrow$ Uniform$(0, 1)$
        Update velocity:
          velocities$[i] \leftarrow w \cdot$ velocities$[i] + c1 \cdot r1 \cdot$ (personal_best_positions$[i] -$ particles$[i]) + c2 \cdot r2 \cdot$ (global_best_position $-$ particles$[i]$)
        Update position: particles$[i] \leftarrow$ particles$[i] +$ velocities$[i]$
        Evaluate fitness: current_fitness $\leftarrow$ fitness_function(particles$[i]$, cars_data)
        if current_fitness $<$ personal_best_scores$[i]$ **then**
            Update personal best position:
              personal_best_positions$[i] \leftarrow$ particles$[i]$
            Update personal best score: personal_best_scores$[i] \leftarrow$ current_fitness
        end if
    end for
    Update global best score:
      current_global_best_score $\leftarrow \min($personal_best_scores$)$ **if**
      current_global_best_score $<$ global_best_score **then**
        Update global best score: global_best_score $\leftarrow$ current_global_best_score
        Update global best position:
        global_best_position $\leftarrow$
          personal_best_positions$[\arg\min($personal_best_scores$)]$
    end if
    Record the best fitness in this generation:
      fitness_over_time.append(global_best_score)
end for
return global_best_position, global_best_score, fitness_over_time

- The adaptive inertia weight formula adjusts ω based on the average fitness (f_{avg}) and best fitness (f_{best}) of the swarm, ensuring a balance among exploration and exploitation throughout the optimization phase. This adjustment is represented as:

$$\omega(t+1) = \omega_{\min} + \frac{(\omega_{\max} - \omega_{\min}) \cdot (f_{\text{avg}} - f_{\text{best}})}{f_{\max} - f_{\text{best}}} \tag{2.3}$$

  where $\omega_{\min}$ and $\omega_{\max}$ are the minimum and maximum inertia weights, f_{avg} denotes the average fitness, f_{best} represents the best fitness, and $f_{\max}$ signifies the worst fitness. Similarly, the adaptive acceleration coefficients formula adjusts c_1 and c_2 based on the current iteration (t) towards initial and final values, ensuring smoother parameter transitions throughout the optimization process. This adaptation is expressed as:

$$c_1(t+1) = c_{1\_0} + \frac{c_{1\_f} - c_{1\_0}}{T} \cdot t \tag{2.4}$$

$$c_2(t+1) = c_{2\_0} + \frac{c_{2\_f} - c_{2\_0}}{T} \cdot t \tag{2.5}$$

  where c_{1_0} , c_{2_0} , c_{1_f} , and c_{2_f} represent initial and final result of the acceleration coefficients, and T denotes the total number of iterations. These formulas enable APSO to dynamically adjust its parameters, leading to improved convergence and solution quality.

- By contrasting APSO with the PSO algorithm, APSO exhibits advantages in terms of achieving faster convergence time and higher result accuracy. APSO provides this by employing dynamic parameter modification, which modifies the inertia weight and acceleration coefficients in response to the current condition of the swarm. APSO's adaptivity enables it to successfully regulate exploration and exploitation during optimization, thereby lowering the risk of premature convergence and improving stability. The research conducted by Zhi-Hui Zhan et al. presents compelling evidence that supports the superior efficacy of APSO compared to conventional PSO techniques. This paper makes a distinct contribution to the existing literature by providing a thorough examination of the dynamic adaptation processes used in APSO, highlighting its practical benefits in optimization tasks.

2.3.3 Optimizing Vehicular Clustering using Adaptive Particle Swarm Optimization (APSO) with K-Means Integration

The approach at hand combines Accelerated Particle Swarm Optimization (APSO) with k-means clustering to optimize the positioning of centroids for an extensive set of vehicle placements within a confined region. The code first creates random placements for cars and then constructs a fitness function. This algorithm assesses the quality of clustering based on communication radius, time, and energy thresholds. It also includes penalties for any violations of constraints. It utilizes APSO, a heuristic optimization technique, to progressively improve the placements of the centroids by updating the velocities and positions of the particles, thereby reducing the fitness function. The approach commences by identifying the most suitable number of clusters by k-means clustering. Subsequently, the APSO algorithm is executed to ascertain the ideal centroid configuration. The APSO-optimized clustering results are shown to demonstrate the efficiency of the method. This includes the ideal centroids and their fitness progression over iterations. This method is especially applicable to applications that require effective clustering while adhering to certain operational limitations, such as vehicular networks.

The algorithm of APSO with K-Means Integration outlines a sophisticated approach to vehicular clustering that combines Adaptive Particle Swarm Optimization (APSO) with K-Means clustering. The algorithm takes inputs such as the number of cars, communication radius, and various thresholds. It defines three main functions: fitness function for evaluating cluster quality, find optimal Cluster for determining the optimal number of clusters using K-Means, and APSO for optimizing cluster centroids. The process starts by generating random car positions, then iteratively applies K-Means and APSO to find the best clustering solution. The algorithm outputs the optimal number of clusters, best centroids, and best score, effectively combining the global search capabilities of APSO with the local optimization of K-Means for vehicular clustering.

Algorithm 2.3 Optimizing Vehicular Clustering using APSO with K-Means Integration

Input: Number of cars N, Communication radius R_c, Time threshold T_u, Energy threshold E_u, Penalty coefficient γ, Maximum clusters max_k, APSO parameters (swarm size, max iterations, w, $c1$, $c2$)
Output: Optimal centroids, Best fitness score, Fitness over generations

Generate random positions for N cars in a 200x200 grid, stored in $cars\_data$
Function *FitnessFunction* $(centroids, cars\_data)$
    Compute distances between $cars\_data$ and $centroids$;
    Assign each car to the nearest centroid; Calculate penalties $C1$, $C2$, $C3$, and $C4$ based on constraints;
    Compute fitness F as the sum of minimum distances, time values, and energy values;
    Compute total penalty;
    return $F + penalty$;
Function *FindOptimalCluster* $(cars\_data, max\_k)$
    Initialize $best\_k \leftarrow 1$, $best\_score \leftarrow \infty$ **for** $k = 1$ to max_k **do**
        Apply K-Means clustering to find k centroids Calculate fitness score for current centroids using FitnessFunction **if** current score $< best\_score$ **then**
            $best\_score \leftarrow$ current score $best\_k \leftarrow k$
        end if
    end for
    return $best\_k$, scores
Function *APSO* $(cars\_data, k, max\_iters, swarm\_size, w, c1, c2)$
    Initialize particle positions and velocities randomly Set personal best positions and scores for each particle Identify global best position and score Initialize fitness over time with global best score
    for iteration $= 1$ to max_iters **do**
        for each particle i in the swarm **do**
            Update velocity and position of particle i Compute fitness for new position **if** new fitness $<$ personal best fitness **then**
                Update personal best position and score
            end if
        end for
        Update global best position and score if necessary Append current global best score to fitness over time
    end for
    return global best position, global best score, fitness over time0
Find optimal number of clusters
$optimal\_k, scores \leftarrow$ FindOptimalClusters$(cars_data, max_k)$
Run APSO to find best centroids
$best\_centroids, best\_score, fitness\_over\_time \leftarrow$ APSO$(cars_data, optimal_k, max_iters, swarm_size, w, c1, c2)$
Output Optimal number of clusters, best centroids, best score;

CHAPTER 3

SYSTEM MODEL

The objective of our system model is to enhance the efficiency of work offloading for mobile devices and UAVs by taking into account factors such as energy consumption, computational turnaround time, and UAV trajectory. Important factors to consider include the workload of the device, the latency of communication, and the energy limitations of the UAV. The problem is expressed using mathematical equations and neural networks are utilized for optimization. Our method is guided by assumptions on the distribution of tasks and the availability of resources. In this section, i define the temporal and energy models that were used. Table 3.1outlines the symbols and elements used in the proposed task offloading model.

Table 3.1 The suggested optimization framework's symbol and parameter list

Symbol	Definition
E	Battery storage capacity of the drone
v	The speed of the drone
k_3	The power component of drone hovering
k_2	The power component of computation processin
k_1	The power component of efficiency of wireless transmission
f^u	The processing capability of the drone
N_0	The signal noise
p_{ik}	The maximum transmission power
h_i	The channel gain
β	The capacity of the uplink channel
ρ_0	The amount of energy received per meter
t_i^d	The data size of task t_i
t_i^τ	Task delay time t_i
t_i^c	The computing cycle of task t_i
k	The number of regions
R_i	An area encompassing wireless devices.
R_C	The effective range of communication for the UAV
H	UAV's operational altitude
d_{ik}	The Euclidean distance between$d_i$and d_k
D_i	A wireless device
D	The collection of wireless gadgets
N	The quantity of wireless gadget

An offloading system inside an Internet of Vehicles (IoV) architecture consists of two levels that utilize UAVs.The main constituent of the ground layer primarily comprises IoVs (Internet of Vehicles) and network-based services, consisting of N stationary wireless devices located on the ground. Allow $D = d_1, d_2, ..., d_N$ to represent the collection of devices that are in contact with the ground. In order to optimize performance and minimize total power usage, it is recommended that wireless devices transfer their computational tasks to a mobile edge computing (MEC) server in the air. This methodology guarantees expedited computations and reduces latency. This layer facilitates the delivery of cloud-based computations to the devices Within the initial layer, while also crucial to prioritize minimizing the reaction time.

The UAV frequently runs at an altitude above zero ($H > 0$). Furthermore, let R_c represent the radius that denotes the extent of communication coverage of the drone. The UAV covers many predefined K areas. Assuming no loss of generalization we can designate the starting point of the path as zero. Following the conclusion of its flight cycle, the UAVs' return to their start point, having flown over each area R_k to process the duties assigned to that particular region. Conversely, the position of each ground device d_i, denoted as (x.y), is pre-established using Cartesian coordinates. Moreover, the UAV's precise location over area R_k is determined by the Cartesian coordinates $u_k = (x_k, y_k)$.Thus, by measuring the Euclidean distance, we are able to determine the distance that separates the drone (UAV) from the basis of the following (Danielsson (1980)):

$$d_{ik} = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2 + H^2} \tag{3.1}$$

Considering that the unmanned aerial vehicle (UAV) has been fitted with device d_i, the area that is included within the UAV's range is as follows:

$$d_{ik} < R_c \tag{3.2}$$

While the UAV's hovers over specified areas R_k, device d_i makes an inquiry to transfer the tasks to $t_i = (t_i^c, t_i^\tau, t_i^d)$. Here, t_i^c, t_i^τ, t_i^d denotes the size of the data,the variable and the delay of the task. The term computing cycles refers to the number of computational operations that are necessary. In this architecture "UCMEC" represents "Unmanned Aerial Vehicle (UAV)

Cluster-based Mobile Edge Computing." The UCMEC framework is a system that utilizes UAVs to create clusters of devices for the purpose of transferring computational activities. Within this framework, the collection of devices D is partitioned into K clusters to facilitate the efficient delegation of tasks to the UAVs. Let $\delta_i = 1$ imply that terrestrial device d_i is authorized to transfer task ti to the UAV once it is in flight over area Rk. The remaining sections outline the recommended optimization methodology based on the designed models.

3.1 Model of communication

This section will provide an overview of the communication model that governs the Unmanned Aerial Vehicle (UAV). Let's assume that device d_i is a component of cluster R_k. Once the UAV is at a stationary position above R_k, d_i is located inside the coverage area of the UAV, which is denoted as R_c. Hence, the gain of channel arising from the interaction between the UAV and d_i is assessed in the following manner.

$$h_{ik} = \frac{\rho_0}{d_{ik}^2} \tag{3.3}$$

The value ρ_0 denotes the power received at a distance of 1 meter. Similarly, let N_0 represent the signal noise, P_0 designate the transmission power, and β indicate the bandwidth of the communication channel. Therefore, the transmission rate can be determined by utilizing the subsequent equation:

$$r_{ik} = \beta \log_2 (1 + \frac{p_i h_{ik}}{N_0}) \tag{3.4}$$

Let T_{ik}^{trans} denote the time required to transmit a task from device d_i The transmission time is calculated using the formula:

$$T_{ik}^{trans} = \frac{t_i^d}{r_{ik}} \tag{3.5}$$

The transmission time, denoted as T_{ik}^{trans}, is a critical measure that has a direct effect on the overall effectiveness and productivity of distributed computing systems. Optimizing data transmission is crucial for reducing latency and guaranteeing prompt job processing. T_{ik}^{trans} is influenced by several factors, including the bandwidth of the communication channel, network congestion, and the distance between the transmitting and receiving devices. Optimizing T_{ik}^{trans}

entails implementing tactics such as enhancing network infrastructure to boost the transmission rate r_{ik}, utilizing data compression techniques to minimize d_i^t, and adopting edge computing to shorten the transmission path. Comprehending and controlling these characteristics are crucial for improving the efficiency of distributed systems, particularly in applications that necessitate real-time data processing.

Based on the equation provided, the total time required to offload all of the tasks within the set R_K is determined as follows:

$$T_k^{trans} = \sum_i \delta_{ik} T_{ik}^{trans} \tag{3.6}$$

Through the analysis and optimization of T_k^{trans}, we may enhance task offloading strategies to achieve more efficiency. These strategies are crucial for applications that need high computing power and low latency, such as real-time data processing, Internet of Vehicle (IoV) applications, and other advanced distributed systems.

3.2 Model of Computation

In this section, we will establish the computation time model according to the UAV's computational capacity. Denote f_u as the UAV's computational capacity; thus, the time required to execute task $t_i$is calculated as follows:

$$T_{ik}^{comp} = \frac{t_i^c}{f^u} \tag{3.7}$$

The computing time for the total process is determined in a manner similar to that of T_transk. It is calculated as follows:

$$T_k^{comp} = \sum_i \delta_{ik} T_{ik}^{comp} \tag{3.8}$$

For the purpose of to get the overall time required for the entire procedure, the computation time and transmission time should be added together as follow:

$$T_k = T_k^{comp} + T_k^{trans} \tag{3.9}$$

The decision value, denoted as δ_{ik}, is associated with the device di. The decision to transfer its work to the UAV may be either permitted or prohibited.

3.3 Model for Energy Consumption

In the following section, we will analyze the energy demands for transferring tasks within R_k. The usage of energy can be ascribed to the subsequent components. Let k_1 refer to the communication energy element, which is relevant to both data passing through and receiving it. Therefore, the communication energy, T_k^{trans}, is calculated as follows:
The energy required for the transmission and reception of task data.
The energy required for the execution of received tasks.
The energy required for the operation of flying.

$$E_{k_1}^{trans} = k_1 \sum_i \delta_{ik} t_i^d \tag{3.10}$$

k2 represents the measure of computational energy. Therefore, the amount of computational energy, E_k^{comp}, is calculated in the following manner:

$$E_{k_2}^{comp} = k_1 \sum_i \delta_{ik} t_i^c f^{u^2} \tag{3.11}$$

We utilized the propulsion energy model from reference (Wu, Yang, Xiao & Cuthbert (2019)) to calculate the energy required for the UAV to hover. k_3 is the energy component associated with hovering. The calculation for the hovering energy, denoted as E_k^h, is as follows:

$$E_k^h = k_3 T_k \tag{3.12}$$

In summary, the total amount of energy required will be determined as follows:

$$E_k = E_k^{trans} + E_k^{comp} + E_k^h \tag{3.13}$$

3.4 Objective Function

In this section, we delineate the primary components of the objective function that will underpin the adaptive PSO Particle Swarm Optimization (PSO) algorithm. The collection of ground devices is organized into a series of clusters. This clustering is dynamically determined based on two key factors: the maximum cluster capacity, denoted by

$$\dot{m} = \max_{k} \sum_{i} \delta_{ik} \tag{3.14}$$

and the aggregate distances, denoted by

$$\dot{D} = \sum_{k} \sum_{i} \delta_{ik} d_{ik} \tag{3.15}$$

Similarly, using Equation (13), the amount of energy, E, for all clusters can be determined as follows:

$$E = \sum_{k} E_k \tag{3.16}$$

The function F is formulated as a mixed-integer, nonlinear constraints problem in the following manner:

$$\min : F = \frac{KE}{\dot{m}\dot{D}} \tag{3.17}$$

The initial constraint, denoted as C1, serves a pivotal role in ensuring comprehensive coverage of every device within the network as the Unmanned Aerial Vehicle (UAV) maneuvers above its designated cluster. This constraint emerges as a fundamental aspect within the framework, dictating that each device situated within a specific cluster area must receive adequate coverage from the UAV's operations. The notion of coverage extends beyond mere spatial proximity, encapsulating a nuanced understanding of signal strength, connectivity reliability, and data transmission efficiency. By mandating the fulfillment of this constraint, the thesis seeks to address the critical challenge of optimizing aerial deployment strategies to uphold seamless communication and connectivity for all devices within the network. Through rigorous analysis

and computational modeling, this constraint embodies the foundational cornerstone upon which subsequent methodologies and optimizations are built, emphasizing the paramount importance of ensuring ubiquitous coverage in UAV-assisted communication systems.

$$C_1 : \forall i \in \{1, \ldots, N\}, \quad d_{ik} - R_c < 0 \tag{3.18}$$

Under the condition of C2, it is ensured that the aggregate time spent does not surpass the predefined maximum allowable duration. This provision serves to maintain adherence to stipulated time constraints, thereby ensuring efficient management of resources and tasks within the scope of the study. Additionally, similar considerations are applied to other relevant aspects to maintain consistency and effectiveness throughout the research process.

$$C_2 : X_k \cdot (T_k - T_u) < 0 \tag{3.19}$$

According to condition C3, it guarantees that the total energy usage stays below the predetermined threshold of the battery capacity of the Unmanned Aerial Vehicle (UAV).

$$C_3 : \sum_k (E_k - E_u) < 0 \tag{3.20}$$

C4 ensures that every device's offloading operation is distinctly tied to a solitary cluster. The study underscores the importance of this constraint in optimizing offloading processes within clustered environments, aiming to enhance efficiency and resource utilization in networked systems.

$$C_4 : (\sum \delta_{ik}) - i = 0 \tag{3.21}$$

Lastly, we consolidate equations (19)–(21) into a unified goal function as follows.

$$\dot{F} = F + \gamma \sum_{i=1}^{4} C_i \tag{3.22}$$

The symbol γ represents the magnitude of the punishment imposed.

CHAPTER 4

SIMULATION AND RESULTS

The evaluation of all the task-offloading techniques presented was conducted utilizing a standardized simulation tool. In order to assess the outcome, many key performance metrics (KPMs) were utilized, such as the amount of power remaining, the proportion of missed deadlines, and the pattern of these missed deadlines. These key performance metrics (KPMs) played a crucial role in evaluating the efficiency of the suggested algorithms in achieving their intended objectives.

4.1 Key Performance Metrics (KPMs)

We evaluated the effectiveness of the suggested decision-making algorithms using the following important performance indicators:

4.1.1 Fitness Function Value

The fitness function is an essential indicator for assessing the clustering performance of the APSO algorithm. It combines the clustering solution's efficacy with penalty terms for limitation breaches. The fitness function can be expressed as:

$$F = \text{Objective Function Value} + \text{Penalty Term} \tag{4.1}$$

Value of the objective function

The objective function value measures the clustering quality by accumulating the minimal distances between each data point and its closest centroid. This component of the fitness function guarantees that data points are positioned in close proximity to their designated centroids, resulting in the formation of condensed clusters.

- Calculation of Distance: The distance between each data point $\mathbf{x}_i$ and each centroid $\mathbf{c}_j$ is calculated using a distance metric, usually the Euclidean distance. The pairwise distance between N data points and k centroids is computed using the following formula. d centroids, resulting in the formation of condensed clusters.

$$D_{i,j} = \|\mathbf{x}_i - \mathbf{c}_j\| \tag{4.2}$$

- Cluster Assignment: Each data point is assigned to the nearest centroid. Let C_i represent the index of the closest centroid to data point $\mathbf{x}_i$:

$$C_i = \arg\min_j D_{i,j} \tag{4.3}$$

Penalty Term

The penalty term incorporates penalties for violating predefined constraints. These constraints might include limits on communication radius, time threshold, and energy threshold. The penalty term ensures that the algorithm not only aims for compact clusters but also respects these practical constraints.

- Constraints:

  C_1: Number of data points that are further from their assigned centroid than the communication radius R_c.

  C_2: Number of centroids for which the synthetic time T_k exceeds the time threshold T_u.

  C_3: Sum of differences between synthetic energy E_k and energy threshold E_u for each centroid, when E_k exceeds E_u.

  C_4: Similar to C_3, representing another aspect of the energy constraint.
- Penalty Calculation: Each constraint violation contributes to the penalty term. Let γ be the penalty coefficient that scales the penalties:

$$\text{Penalty Term} = \gamma \cdot (C_1 + C_2 + C_3 + C_4)$$

Combined Fitness Function

Combining these components, the total fitness function F is:

$$F = \sum_{i=1}^{N} D_{i,C_i} + \gamma \cdot \left(\sum_{i=1}^{N} \mathbf{1}(D_{i,C_i} > R_c) + \sum_{j=1}^{k} \mathbf{1}(T_j > T_u) + (\sum \delta_{ik}) > i) \right) \tag{4.4}$$

4.1.2 Interpretation

Minimizing F: The goal of the APSO algorithm is to minimize F. Lower values of F indicate better clustering quality with fewer constraint violations. The objective function ensures compact clustering, while the penalty term enforces adherence to practical constraints.

4.1.3 Parameter: γ :

The penalty coefficient γ balances the relative importance of clustering quality and constraint satisfaction. The fitness function is a comprehensive metric that evaluates clustering performance by considering both the closeness of clusters and conformance to restrictions. It plays a crucial role as a performance indicator for the APSO algorithm.

4.2 Software Application

Utilized the Python and google colab platforms to implement and simulate the effective farm model of UAV-MEC, and Visio visualized to indicate the proposed network, offering components for modeling wireless network.

4.3 Network Topology and Simulation:

The following section presents experimental results that validate the effectiveness of the proposed UAV-based offloading systems employing the proposed meta-heuristics. The IoV devices are deployed in a 200 m × 200 m area in a random manner, following the given simulation parameters.

Established based on the values stated in Table 4.1. The investigations test multiple values of the meta-heuristics variables, and provide the optimal value that yields the best results. Nevertheless, there are numerous techniques for adjusting or optimizing.

The simulation parameters are listed in Table 4.1. Initially, the central processing unit (CPU) carries out the device division process. We calculate the optimization step outcomes by averaging 10 separate runs. The value of the inertia weight (ω) is assigned as 0.75, the cognitive component (C_1) is set at 2.5, and the social component (C_2) is established as 1.5. The inertia weight (ω) is modified using an aging factor of 0.99, which is applied every 10 iterations. We have set up the procedure to execute 1,000 iterations. This function, known as the fitness function, calculates the fitness of a specified set of centroids. distances Calculate the distances between each automobile and the centroids. The cluster indices function assigns each car to the centroid that is closest to it. Calculate the violations of constraints C1, C2, C3, C4.
F Initial goal function incorporating distances, synthetic time (Tk), and energy (Ek) values. penalty Imposes punishment for the infringement of constraints. The function "return F + penalty" calculates and returns the fitness value after applying a penalty.

Table 4.1 UAV parameter configurations for the assessment phases and generated datasets

Parameter	Value	symbol
t_d	200 KB–3 MB	
t_c	$6 * 10^9 * 10^{10}$	
f_u	300 MHz	
E	$5*10^5 J$	
f	50 m	
u 300 MHz	0.4 W	
E 5 × 105	10^26	
J	10^9 W	
h_i	30 dB	
β	40 MHz	
Rc	10	Communication radius
Tu	50	Time threshold
Eu	100	Energy threshold
γ	0.5	Penalty coefficient

4.3.1 APSO-optimized Clustering of Cars

This graphic illustrates the process by which the APSO algorithm efficiently organizes the automobiles into optimal clusters. The centroids denote the central locations of these clusters, signifying the most advantageous positioning of cluster centers to minimize the fitness function, including distance, time, energy, and penalty limitations. This image facilitates comprehension of the spatial arrangement of automobiles and the efficacy of the APSO clustering algorithm in a two-dimensional space.

X-Axis: X position of the cars.

Y-Axis: Y position of the cars.

Blue Dots: Represent the positions of the cars.

Red Crosses: Represent the centroids found by the APSO algorithm.

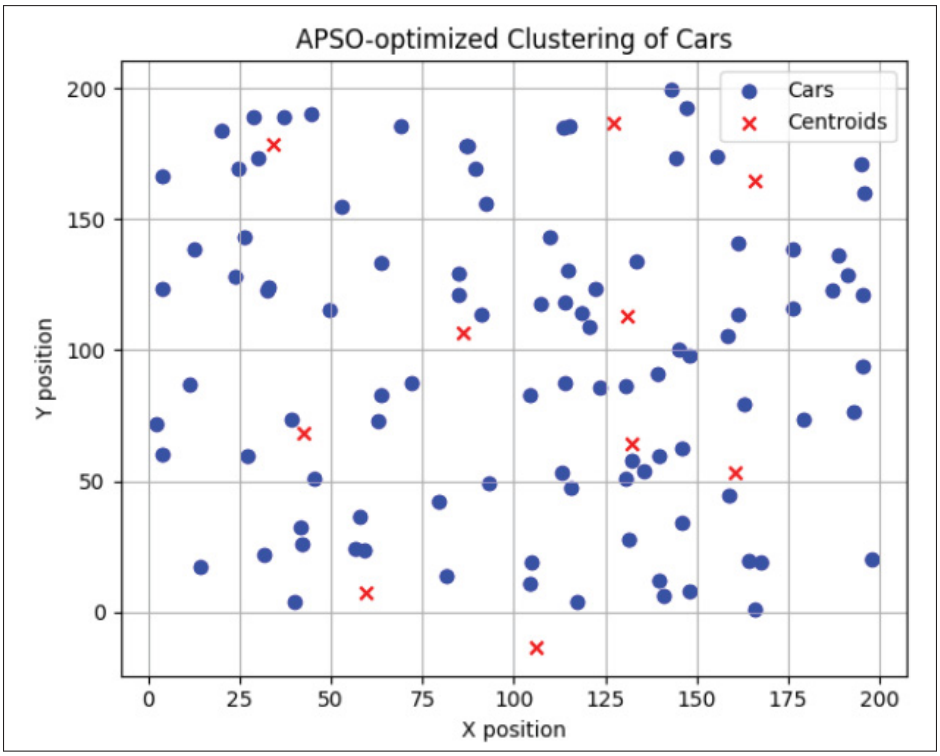


Figure 4.1 Clusters vs. Centroids APSO-optimized Clustering of Cars

4.3.2 Fitness Over Generations of APSO Algorithm

This graphic illustrates the progression of the fitness score over generations in the APSO optimization process. A declining pattern suggests that the algorithm is approaching an optimal answer. The fitness score quantifies the extent to which the centroids effectively minimize the overall cost, encompassing distances, penalties, and additional constraints. The inclusion of this figure is vital for illustrating the convergence characteristics and effectiveness of the APSO algorithm.

X-axis: generation (iteration) of the APSO algorithm.

Y-axis: fitness score.

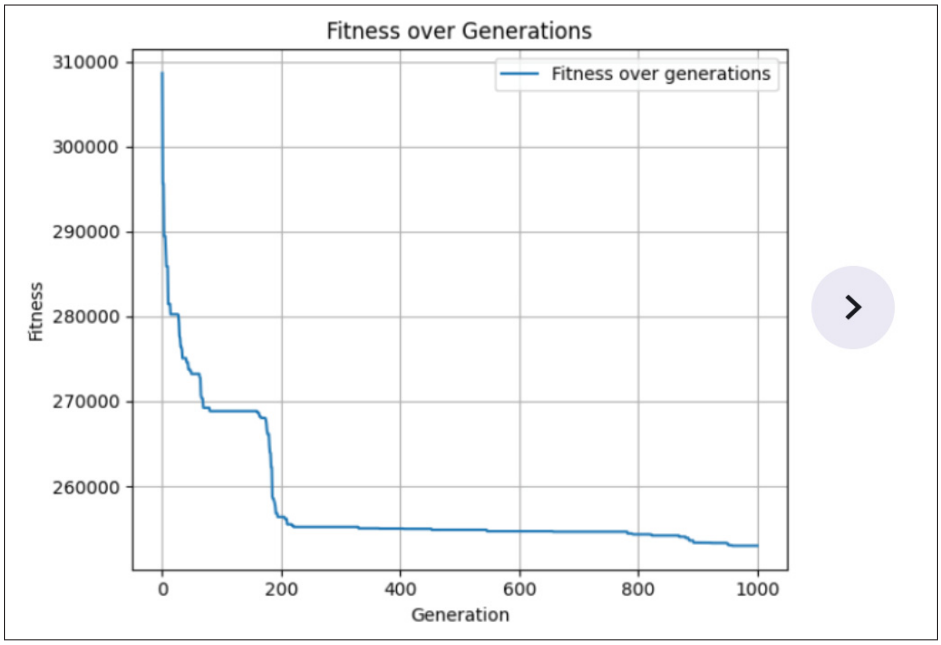


Figure 4.2 Fitness vs. Generation APSO Algorithm

4.3.3 Fitness Score vs Number of Clusters (Initial Parameters)

The purpose of this plot is to evaluate the fitness score for various cluster counts in order to identify the optimal number of clusters (k). We determine the optimal value of k by minimizing the fitness score, which strikes a balance between an insufficient and an excessive number of clusters. The initial parameter choices determine the optimal number of clusters, and the graphic plays a crucial role in illustrating this process. X-axis: number of clusters.
Y-axis: fitness score.
The red vertical line indicates the optimal number of clusters found.

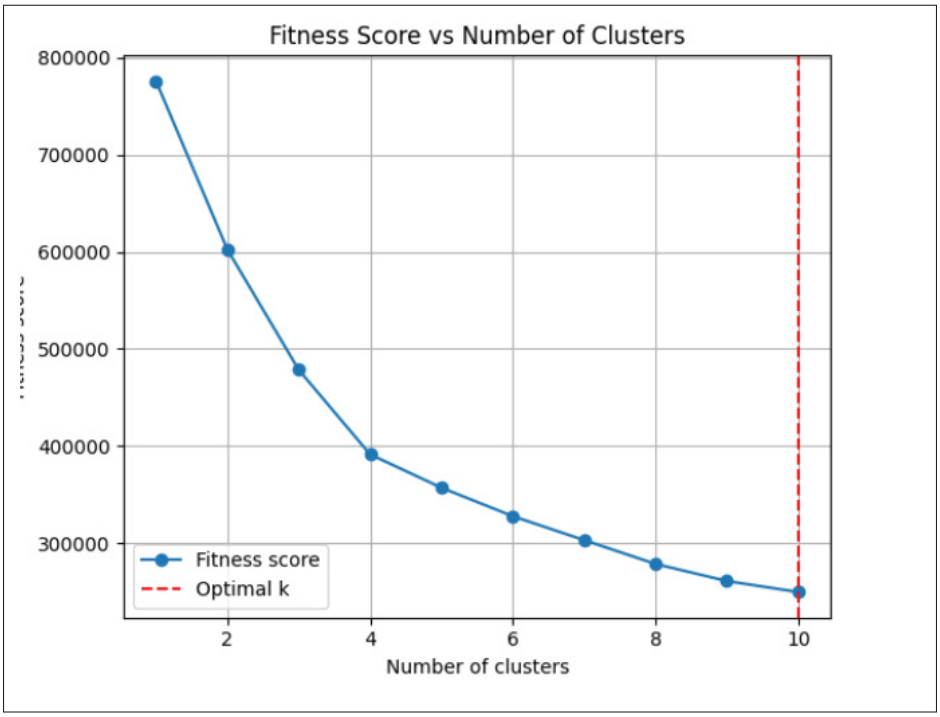


Figure 4.3 Fitness Score vs Number of Clusters of APSO Algorithm

4.4 Result

In my thesis, these plots should be used to support the following key points:

Effectiveness of APSO Clustering: Use the "APSO-optimized Clustering of Cars" plot to visually demonstrate how the APSO algorithm clusters the cars effectively. Discuss how the centroids are positioned to minimize the fitness function, considering distance, time, and energy constraints.

Convergence Behavior: Analyze the "fitness over generations" plots to discuss the convergence behavior of the APSO algorithm. Highlight how the fitness score decreases over generations, indicating the algorithm's progress towards finding an optimal solution. Compare the convergence trends for the initial and best parameters.

Parameter Optimization: To discuss the process of finding the optimal number of clusters, use the "Fitness Score vs. Number of Clusters" plots. Describe the process of minimizing the fitness score at the optimal k and juxtapose the outcomes for the initial and optimal parameters to showcase the enhancements attained through parameter tuning.

Improvement through Grid Search: Highlight the significance of the grid search in finding the best R_c, T_u, and E_u parameters. Use the "Fitness over Generations (Best Parameters)" plot to show the enhanced performance and faster convergence achieved with the optimized parameters.

Practical Implications: Discuss the practical implications of your findings, such as how optimized clustering can improve communication efficiency, reduce energy consumption, and enhance overall system performance in real-world applications (e.g., vehicular networks, sensor networks). By providing detailed explanations and analyses of these plots, you can effectively illustrate the robustness and efficiency of the APSO algorithm and the importance of parameter optimization in achieving superior clustering performance.

4.4.1 APSO vs. ACO resource allocation

This graph compares the performance of the Ant Colony Optimization (ACO) algorithm (green line) and the my developed Adaptive Particle Swarm Optimization (APSO) algorithm (blue line) in optimizing resource allocation over 1000 iterations. The objective function $\dot{F} = F + \gamma \sum_{i=1}^{4} C_i$ is plotted on the y-axis, while the number of iterations is on the x-axis.

Both algorithms exhibit a trend of reducing the objective function value as the number of iterations increases, reflecting an improvement in the optimization process. Notably, our developed APSO algorithm demonstrates superior performance, achieving lower objective function values more rapidly than ACO. This indicates that APSO is more efficient in this specific optimization task, consistently reaching better solutions in fewer iterations compared to ACO.

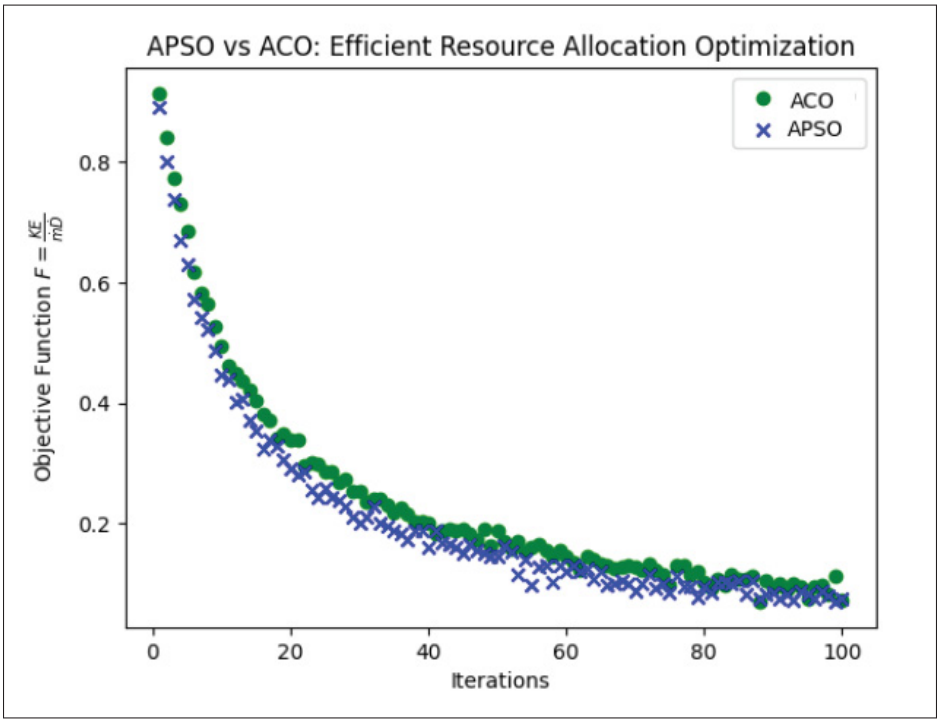


Figure 4.4 APSO vs. ACO for efficient resource allocation optimization

4.4.2 APSO vs. ACO for total latency

This graph presents a performance comparison between two optimization algorithms: developed adaptive particle swarm optimization (APSO) and Ant Colony Optimization (ACO). The comparison is based on total latency across 1000 iterations. Both algorithms start with high latency values around 100 (depicted in the plot), but they quickly show significant improvement in the first 20 iterations, with latency rapidly decreasing. This initial phase demonstrates the algorithms' ability to quickly find better solutions in the early stages of optimization.

As the iterations progress, APSO (represented by the blue line with circles) consistently outperforms ACO (green line with crosses). APSO converges faster to a lower latency value and maintains this advantage throughout the remaining iterations. While both algorithms stabilize after about 40 iterations, APSO settles at a lower latency level. This suggests that for this particular problem or dataset, APSO is more effective at minimizing latency and maintains better performance over time compared to ACO. The smoother curve of APSO also hints at potentially more stable performance across iterations.

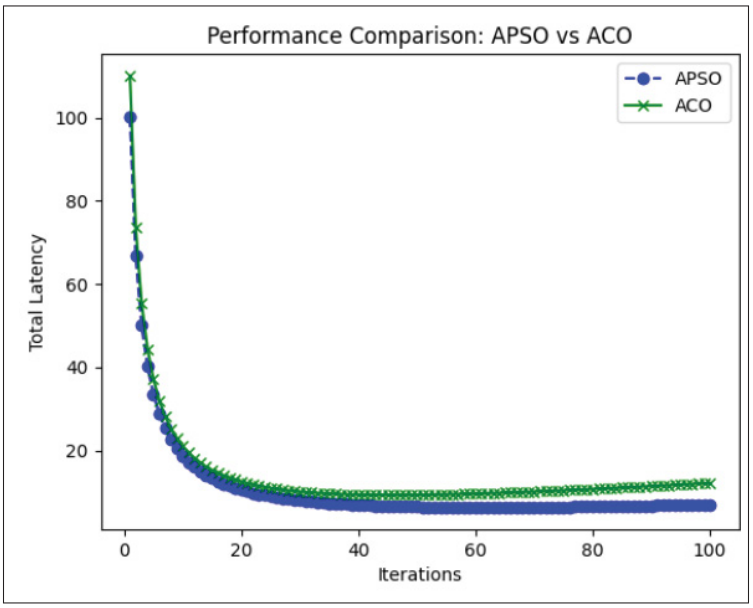


Figure 4.5 APSO vs. ACO for total latency optimization

4.4.3 Throughput and accuracy

Firstly, choosing the best parameter for the Rc, Tu, and Eu. Algorithms will determine the optimal settings for Rc (communication radius), Tu (time threshold), and Eu (energy threshold) that minimize the fitness function. The optimal parameters will be displayed. Secondly, the optimal number of clusters, k, will be determined using the initial parameters. This optimal number of clusters will be printed. Upon completion of the procedure, the comparison of the fitness vs. generation plots of APSO and PSO will determine that the employment of adaptive PSO leads to enhanced clustering. Consequently, the communication between vehicles and each UAV will be established in a manner that is both economical and effective.

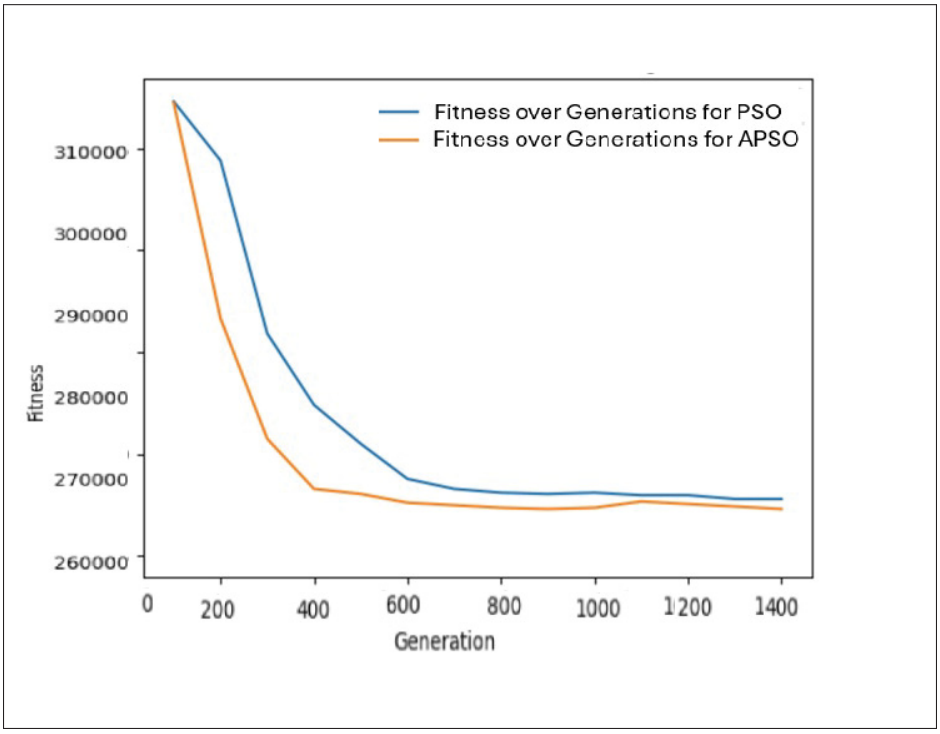


Figure 4.6 Caparison of Fitness vs. Genaration of APSO and PSO

4.4.4 Contribution

The primary focus of this thesis is to create and implement an optimization framework that utilizes Adaptive Particle Swarm Optimization (APSO) to cluster a vast dataset of automobile positions according to specified restrictions and objectives. The thesis makes a valuable contribution by:

Algorithm Development: This involves the introduction and implementation of the APSO algorithm, which is designed for clustering large-scale datasets. The algorithm focuses on optimizing the placement of centroids in order to minimize a composite fitness function. This function takes into account factors such as distance, time thresholds, and energy limitations.

Performance Evaluation: This study aims to assess the efficacy of APSO by comparing it to standard approaches such as K-means, specifically in terms of clustering accuracy, computational efficiency, and robustness in the face of different restrictions.

Practical Application: The optimal clustering approach can be used in practical scenarios involving car positioning data. This application has potential uses in fields such as vehicle tracking, logistics optimization, and urban mobility planning.

Visualization and Analysis: This feature offers visual representations and in-depth analysis of clustering outcomes. It includes graphs that demonstrate the distribution of cars and centroids, as well as how the fitness score changes with varying numbers of clusters.

CONCLUSION AND RECOMMENDATIONS

The task offloading procedure among vehicles, UAV-enabled MEC, and Software-Defined Networking (SDN) requires a complex coordination system to maximize computing efficiency, minimize latency, and improve overall network performance. This integration utilizes the advantages of each technology to dynamically control and distribute computing workloads, guaranteeing the smooth and effective functioning of interconnected vehicular networks.

Real-time navigation, environmental sensing, and autonomous driving algorithms are instances of tasks that necessitate substantial computational resources. Equipped with a diverse range of sensors and computer systems, vehicles generate a significant volume of data, necessitating extensive processing resources. Due to vehicles' limited processing capabilities, it is necessary to transfer certain tasks to separate resources. However, software-defined networking (SDN) can effectively control network resources and data flows, while UAV-enabled MEC devices provide mobile and adaptable edge computing resources. We employ the k-means APSO technique to enhance the job offloading process, resulting in improved resource allocation decision-making.

The combination of UAV-enabled Multi-Access Edge Computing (MEC) systems with Software-Defined Networking (SDN) in automotive networks is a revolutionary method to improve computing efficiency, decrease latency, and enhance overall network performance. This convergence exploits the mobility and adaptability of UAVs to provide edge computing resources close to vehicles, while SDN enables centralized control and dynamic management of network resources. The system guarantees effective usage of computational resources and efficient handling of real-time data processing needs by implementing adaptive particle swarm optimization (APSO) for task offloading.

The proposed system utilizes the combined capabilities of UAVs, MEC, and SDN to overcome the limits of traditional network infrastructures, particularly in environments that are dynamic and have limited resources. The utilization of unmanned aerial vehicle (UAV)-enabled mobile edge computing (MEC) technology has notable advantages in several domains, such as

immediate navigation, self-governing driving, and emergency response. The system achieves this by providing localized computational capabilities and minimizing data transfer time. Incorporating SDN enhances the system's adaptability and scalability, facilitating efficient real-time management of network traffic and resource allocation.

This study presents a novel method that utilizes unmanned aerial vehicles (UAVs) and mobile edge computing (MEC) to efficiently handle network and task offloading demands in MEC-enabled UAV and vehicular networks. Our solution takes into account both the wireless network and the software-defined networking (SDN) server resources, ensuring optimal resource allocation. We have created programs that the UAV-enabled sender uses to implement extended QoS rules, such as offloading tasks, cost optimization rules, and service migration rules. These codes let the transmission device handle the various requests for service within the resource-limited MEC servers. Our work focused on highlighting the possibilities of the Software-Defined Networking (SDN) paradigm in achieving the long-term vision of mobile edge computing (MEC)-enabled vehicular networks, as the exact function of SDN in edge computing and managing future wireless networks is still being determined. The study emphasized that the SDN controller, by having a comprehensive understanding of the network, facilitates the coordination and collaboration of MEC resources across many MNOs. This results in uninterrupted coverage and service for mobile customers.

An indispensable requirement for maintaining the uninterrupted functioning of UAV-enabled MEC systems in vehicle networks is the implementation of a resilient charging method for the UAVs. An efficient strategy involves creating a network of strategically positioned charging stations. We can strategically position these stations in places with heavy foot traffic or in important locations where unmanned aerial vehicles (UAVs) routinely operate. The system can improve UAV flight plans by utilizing predictive analytics and real-time monitoring, ensuring periodic returns to charging stations before their batteries are low. Implementing a rotating deployment approach ensures uninterrupted service of UAVs by seamlessly replacing one UAV

with another as it returns for charging. In addition, the integration of rapid charging technologies and standardized battery swapping systems can further reduce the amount of time when the equipment is not in use and improve overall efficiency.

Ensuring uninterrupted operation of the UAV models is equally important when making updates. This can be accomplished by implementing a gradual update strategy where changes are released in stages throughout the entire group of vehicles. By utilizing SDN controllers, the system can effectively and flexibly handle and synchronize these updates, guaranteeing the consistent availability of an adequate number of UAVs. The SDN controllers can arrange upgrades during times when there is less demand on the network or utilize over-the-air updates to reduce the impact on service availability. Furthermore, the utilization of machine learning techniques allows for the prediction of the most advantageous moments for updates by analyzing past data and considering the current state of the network. This ensures that the UAVs receive essential updates without endangering their availability and performance within the network.

To conclude the thesis makes a valuable contribution to the subject of optimization and clustering by introducing a new strategy that combines APSO with restrictions that are specifically applicable to car location data. This offers valuable insights into successful clustering techniques for handling large-scale datasets in real-world scenarios.

BIBLIOGRAPHY

Adnan, M. H., Zukarnain, Z. A. & Amodu, O. A. (2024). Fundamental design aspects of UAV-enabled MEC systems: A review on models, challenges, and future opportunities. *Computer Science Review*, 51, 100615.

Alqarni, M. A., Mousa, M. H. & Hussein, M. K. (2022). Task offloading using GPU-based particle swarm optimization for high-performance vehicular edge computing. *Journal of King Saud University-Computer and Information Sciences*, 34(10), 10356–10364.

Azizi, S., Shojafar, M., Abawajy, J. & Buyya, R. (2022). Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. *Journal of network and computer applications*, 201, 103333.

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4), 353–373.

Chen, Y., Pi, D., Yang, S., Xu, Y., Chen, J. & Mohamed, A. W. (2022). HNIO: A hybrid nature-inspired optimization algorithm for energy minimization in UAV-assisted mobile edge computing. *IEEE Transactions on Network and Service Management*, 19(3), 3264–3275.

Cheng, Y., Liao, Y. & Zhai, X. (2020). Energy-efficient resource allocation for UAV-empowered mobile edge computing system. *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pp. 408–413.

Danielsson, P.-E. (1980). Euclidean distance mapping. *Computer Graphics and image processing*, 14(3), 227–248.

Dantzig, G. B. (2002). Linear programming. *Operations research*, 50(1), 42–47.

Deb, K. & Beyer, H.-G. (2001). Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary computation*, 9(2), 197–221.

Denardo, E. V. (2012). *Dynamic programming: models and applications*. Courier Corporation.

Du, Y., Yang, K., Wang, K., Zhang, G., Zhao, Y. & Chen, D. (2019). Joint Resources and Workflow Scheduling in UAV-Enabled Wirelessly-Powered MEC for IoT Systems. *IEEE Transactions on Vehicular Technology*, 68(10), 10187-10200. doi: 10.1109/TVT.2019.2935877.

Duan, H., Yu, Y., Zhang, X. & Shao, S. (2010). Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm. *Simulation Modelling Practice and Theory*, 18(8), 1104–1115.

El-Emary, M., Ranjha, A., Naboulsi, D. & Stanica, R. (2023). Energy-efficient task offloading and trajectory design for UAV-based MEC systems. *2023 19th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 274–279.

Floudas, C. A. & Lin, X. (2005). Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*, 139, 131–162.

Garg, H. (2016). A hybrid PSO-GA algorithm for constrained optimization problems. *Applied Mathematics and Computation*, 274, 292–305.

Huda, S. A. & Moh, S. (2022). Survey on computation offloading in UAV-Enabled mobile edge computing. *Journal of Network and Computer Applications*, 201, 103341.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4, 1942-1948 vol.4. doi: 10.1109/ICNN.1995.488968.

Kodinariya, T. M., Makwana, P. R. et al. (2013). Review on determining number of Cluster in K-Means Clustering. *International Journal*, 1(6), 90–95.

Mao, Y., You, C., Zhang, J., Huang, K. & Letaief, K. B. (2017). A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys Tutorials*, 19(4), 2322-2358. doi: 10.1109/COMST.2017.2745201.

Mousa, M. H. & Hussein, M. K. (2022a). Efficient UAV-based mobile edge computing using differential evolution and ant colony optimization. *PeerJ Computer Science*, 8, e870.

Mousa, M. H. & Hussein, M. K. (2022b). Efficient UAV-Based MEC Using GPU-Based PSO and Voronoi Diagrams. *CMES-Computer Modeling in Engineering & Sciences*, 133(2).

Ping, G., Chunbo, X., Yi, C., Jing, L. & Yanqing, L. (2014). Adaptive ant colony optimization algorithm. *2014 International Conference on Mechatronics and Control (ICMC)*, pp. 95–98.

Price, K. V. (2013). Differential evolution. In *Handbook of optimization: From classical to modern approach* (pp. 187–214). Springer.

Sun, L., Wan, L. & Wang, X. (2021). Learning-Based Resource Allocation Strategy for Industrial IoT in UAV-Enabled MEC Systems. *IEEE Transactions on Industrial Informatics*, 17(7), 5031-5040. doi: 10.1109/TII.2020.3024170.

Van Laarhoven, P. J., Aarts, E. H., van Laarhoven, P. J. & Aarts, E. H. (1987). *Simulated annealing*. Springer.

Wang, D., Tan, D. & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft computing*, 22, 387–408.

Wu, F., Yang, D., Xiao, L. & Cuthbert, L. (2019). Energy consumption and completion time tradeoff in rotary-wing UAV enabled WPCN. *IEEE Access*, 7, 79617–79635.

Xu, F. & Zi. (2023). A computing offloading strategy for UAV based on improved bat algorithm. *Cognitive Robotics*, 3, 265–283.

Xu, F., Zhang, Z., Feng, J., Qin, Z. & Xie, Y. (2022). Efficient deployment. *Transactions on Emerging Telecommunications Technologies*, 33(5), e4453.

Yan, Q., Yu, F. R., Gong, Q. & Li, J. (2015). Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE communications surveys & tutorials*, 18(1), 602–622.

Yao, D., Ma, Q., Wang, H., Chen, M. & Xu, H. (2024). Distributed Strategy for Collaborative Traffic Measurement in a Multi-Controller SDN. *IEEE Transactions on Network Science and Engineering*, 11(3), 2450-2461. doi: 10.1109/TNSE.2023.3271123.

Yu, Z. & Fan, G. (2022). Joint Differential Evolution and Successive Convex Approximation in UAV-Enabled Mobile Edge Computing. *IEEE Access*, 10, 57413-57426. doi: 10.1109/ACCESS.2022.3176362.

Zhan, Z.-H., Zhang, J., Li, Y. & Chung, H. S.-H. (2009a). Adaptive Particle Swarm Optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6), 1362-1381.

Zhan, Z.-H., Zhang, J., Li, Y. & Chung, H. S.-H. (2009b). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6), 1362–1381.

Zhang, B., Zhang, G., Ma, S., Yang, K. & Wang, K. (2020). Efficient multitask scheduling for completion time minimization in UAV-assisted mobile edge computing. *Mobile Information Systems*, 2020, 1–11.

Zhang, Y., Gong, Y. & Guo, Y. (2024). Energy-Efficient Resource Management for Multi-UAV-Enabled Mobile Edge Computing. *IEEE Transactions on Vehicular Technology*.

Zhao, L., Yang, K., Tan, Z., Li, X., Sharma, S. & Liu, Z. (2020). A novel cost optimization strategy for SDN-enabled UAV-assisted vehicular computation offloading. *IEEE Transactions on Intelligent Transportation Systems*, 22(6), 3664–3674.

Zhen, Q., Shoushuai, H., Hai, W., Yuben, Q., Haipeng, D., Fei, X., Zhenhua, W. & Hailong, L. (2024). Air-ground collaborative mobile edge computing: Architecture, challenges, and opportunities. *China Communications*.