Machine Learning for Multi-Access Edge Computing and Autonomous Driving

by
Dawei Chen

A dissertation submitted to the Department of Electrical and Computing Engineering,

Cullen College of Engineering

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Electrical Engineering

Chair of Committee: Zhu Han

Committee Member: Miao Pan

Committee Member: Hien Van Nguyen

Committee Member: Jiang Xie

Committee Member: BaekGyu Kim

University of Houston
December 2021

ACKNOWLEDGMENTS

Upon the completion of this thesis, I am grateful to those who have offered me encouragement and support during the course of my study.

First of all, my heartiest thanks flow to my supervisor, Professor Zhu Han, for his helpful guidance, valuable suggestions, and constant encouragement both in my study and in my life. His profound insight and accurateness about my Ph.D. research taught me so much that they are engraved on my heart. He provided me with beneficial help and offered me precious comments during the whole process of my Ph.D. study, which will be an invaluable assets for my future career.

Also, I would like to express my sincere gratitude to all the dissertation committee, Professor Miao Pan, Professor Hien Van Nguyen, Professor Jiang Xie, and Professor BarkGyu Kim. Without their precious time, esteemed guidance, and continuous support, the dissertation would not be what it is now.

My appreciation also goes to my dear colleagues Xunsheng Du, Hongliang Zhang, Kuo Chun Tsai, Saeed Ahmadian, Hao Gao, Jing Li, Wen Xie, Yuhan Kang, and Kai-Chu Tsai, whom I always having great time with in both on and off work time. I am also grateful to the visitors of our laboratory, Ruoguang Li, Chunxia Su, Xuelin Cao, Di Zhou, Chuanyin Li, Junhua Wang, Zhongyu Miao, Benedetta Picano, Yue Yu, Ziye Jia, Yan Zhu, Yu Zhang, Minghui Min, Zirui Zhuang, and Zhuang Ling, whom gives strong support both in study and life. I am grateful to all my friends who give me support during my Ph.D. life. They are Zefeng Gao, Yuhong Fang, Xiaoyu Jiang, Guihao Yan, Kai Yin, Qianwen Wu, and Haifeng Tang.

Last but by no means least, I want to thank my mother, father, grandmothers, grandfathers, uncles, aunts, and cousins, for their countless love, encouragement, and support during my life. Their loving considerations and helps are the source of my strength.

ABSTRACT

Last twenty years have seen the explosive growth of information technology, and we have stepped into the era of information explosion. These technologies significantly contribute to the generation of the unprecedented quantity of data and make it challenging for data storage and data processing. Multi-access edge computing is emerged as an effective way to relieve the pressure of data processing. One of the typical edge computing assisted applications is autonomous driving, which relies heavily on edge servers for data analysis and decision making. However, with the growth of end users, how to allocate the available edge computing resources to fulfill the requirements becomes a challenging problem.

Machine learning, as a popular and effective method for decision making, has diverse applications. Machine learning is the study of computer algorithms that can improve automatically through experience and by the use of data. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without human interference. Therefore, there is a great potential to utilize the ideas, methods, and models of machine learning to make decisions for edge computing resource allocation.

Given this background, this dissertation provides a theoretical research between machine learning, multi-access edge computing, and autonomous vehicles. Especially, different machine learning models and edge computing assisted applications are discussed. The main contribution of this dissertation are as follows.

1. The basic concepts and classifications of machine learning are provided. The architecture, characteristics, and key technologies of multi-access edge computing are given as well.

2. Applications of machine learning for edge computing resources allocation are studied. Also, except the utilization of machine learning method, the efficient machine learning framework design is also discussed.

3. Numerical results are provided to show that the proposed method can be utilized for object realization in edge computing scenario such as accurate prediction, low latency. etc.

4. The potentials of machine learning for multi-access edge computing applications in future wireless networks are discussed.

This dissertation provides a theoretical research between machine learning, multi-access edge computing, and autonomous vehicles, in which different machine learning models are utilized in different multi-access edge computing assisted applications as well as efficient machine learning mechanism design. This work places a fundamental research on edge computing resource allocation. This research has the potential to contribute to the future wireless network area and has a long term effect on problems such as edge computing resource deployment, service guaranteed wireless network design.

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

Last twenty years have seen the explosive growth of information technology, and we have stepped into the era of information explosion. We are embracing the prevalence of the Internet of Things (IoT), Cyber Physical System (CPS), Mobile Internet, wearable devices, smart home, smart grid, smart city, Vehicular Ad Hoc Networks (VANETs), large-scale wireless sensor network (LSWSN), etc., and feasting the convenience they bring in every aspect of daily life. At the same time, these technologies also significantly contribute to the generation of the unprecedented quantity of data and the consequent requirements for data storage and data processing [1]. According to the anticipation of Cisco, the number of devices that are connected to the Internet will achieve more than 50 billions by 2020. Correspondingly as a result, the total amount of yielded data can reach 500 zettabytes by 2019, 45 percent of which will be dealt with or stored in the network [2]. Such a kind of massive data bring a significant challenge to data processing and storage.

To relieve the data processing pressure, multi-access edge computing is emerged as a solution [3]. Edge computing employs a distributed and hierarchical computing model. The edge nodes, which have computational abilities as well, are geographically deployed. Those relatively lower complexity missions can be addressed in the edge nodes directly instead of uploading to the cloud center, so as to the transmission distance can be significantly minified, therewith the decreasing of latency [4]. As a consequence, the edge computing is more suitable to address those real-time missions. One of the typical edge computing assisted applications is autonomous driving, which relies heavily on edge servers for data analysis and decision making [5]. However, with the growth of end users, how to allocate the available edge computing resources to fulfill the requirements becomes a challenging problem.

Machine learning, as a popular and effective method for decision making, has diverse applications. Machine learning is the study of computer algorithms that can improve automatically through experience and by the use of data. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without human interference. Therefore, there is a great potential to utilize

the ideas, methods, and models of machine learning to make decisions for edge computing resource allocation.

In this dissertation, we mainly focus our research on how to use machine method to allocate edge computing resources. Also, we discuss the mechanism design to optimize the machine learning framework. The basics of machine learning and edge computing are provided in Section 1.1 and Section 1.2 respectively. The main organization of this dissertation is given in Section 1.3.

1.1 Machine Learning Basics

Machine learning, as a popular and effective method for decision making, has diverse applications. Machine learning is the study of computer algorithms that can improve automatically through experience and by the use of data. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without human interference. Basically, it can be categorized into three classes: supervised learning, unsupervised learning, and reinforcement learning.

1.1.1 Supervised Learning

For supervised learning, the data sample contains data features as well as desired output. So the basic method is to feed the machine a plenty of data, and in the training process, the backpropagation will be used to adjust the parameters of the machine learning model like weights and biases. Therefore, by calculating a specific piece of input data, it can gives out the desired value. So basically a supervised learning can be divided into classification problem and regression problem. Both the algorithms are used for prediction in Machine learning and work with the labeled datasets. But the difference between both is how they are used for different machine learning problems. The main difference between Regression and Classification algorithms that Regression algorithms are used to predict the continuous values such as price, salary, age, etc. and classification algorithms are used to predict/classify the discrete values, as is shown in Fig. 1.

In machine learning, classification refers to a predictive modeling problem where a

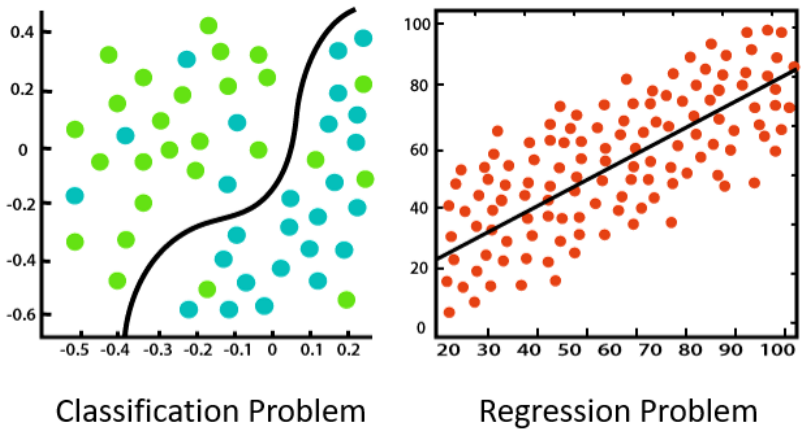


Figure 1: The difference of classification problem and regression problem.

class label is predicted for a given example of input data. The typical classification problems are image classification, identity fraud detection, customer retention, and disease diagnostics. From a modeling perspective, classification requires a training dataset with many examples of inputs and outputs from which to learn. A model will use the training dataset and will calculate how to best map examples of input data to specific class labels. As such, the training dataset must be sufficiently representative of the problem and have many examples of each class label. Class labels are often string values, e.g. "spam," "not spam," and must be mapped to numeric values before being provided to an algorithm for modeling. This is often referred to as label encoding, where a unique integer is assigned to each class label, e.g. "spam" = 0, "no spam" = 1. The objective function for two-class classification problem can be written as

$$f = -\frac{1}{K}\left[\sum_{j=1}^{K} y^j \log(\omega^T x^j) + (1-y^j)\log\left(1-\omega^T x^j\right)\right], \tag{1}$$

where j denotes the j-th data sample, ω is model weights, and K is the number of data samples. There are many different types of classification algorithms for modeling classification predictive modeling problems. But there is no good theory on how to map algorithms onto problem types; instead, it is generally recommended that a practitioner use controlled experiments and discover which algorithm and algorithm configuration results in the best performance for a given classification task.

Regression predictive modeling is the task of approximating a mapping function f

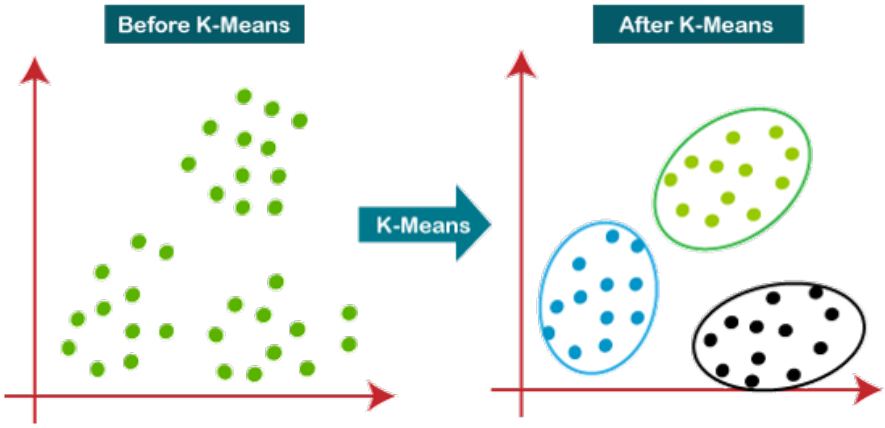


Figure 2: The k-means clustering.

from input variables x to a continuous output variable y. The typical regression problems are like population growth prediction, advertising popularity prediction, weather forecasting, market forecasting, and estimating life expectancy. The typical loss function of a regression problem can be written as

$$f = \frac{1}{2}\left(x^T\omega - y\right)^2, \tag{2}$$

where ω is model weights.

1.1.2 Unsupervised Learning

For unsupervised learning, the data are without label. Unlike supervised learning, the unsupervised learning algorithm should rely on its own to find structure in its input. In some pattern recognition problems, the training data consists of a set of input vectors x without any corresponding target values. The goal in such unsupervised learning problems may be to discover groups of similar examples within the data, where it is called clustering, as shown in Fig. 2, or to determine how the data is distributed in the space, known as density estimation. The clustering applications are like recommend systems, targeted marketing, and customer segmentation. For density estimation problems, the typical applications are like, meaningful compression, big data visualization, structure discovery, feature elicitation.

Also, in another way, unsupervised learning can be categorized into two classes:

parametric unsupervised learning and non-parametric unsupervised learning. For parametric unsupervised learning, it assumes that sample data comes from a population that follows a probability distribution based on a fixed set of parameters. Theoretically, in a normal family of distributions, all members have the same shape and are parameterized by mean and standard deviation. That means if you know the mean and standard deviation, and that the distribution is normal, you know the probability of any future observation. Parametric Unsupervised Learning involves construction of Gaussian Mixture Models and using Expectation-Maximization algorithm to predict the class of the sample in question. This case is much harder than the standard supervised learning because there are no answer labels available and hence there is no correct measure of accuracy available to check the result. In non-parameterized version of unsupervised learning, the data is grouped into clusters, where each cluster(hopefully) says something about categories and classes present in the data. This method is commonly used to model and analyze data with small sample sizes. Unlike parametric models, nonparametric models do not require the modeler to make any assumptions about the distribution of the population, and so are sometimes referred to as a distribution-free method.

1.1.3 Reinforcement Learning

Reinforcement learning is the training of machine learning models to make a sequence of decisions. The agent learns to achieve a goal in an uncertain, potentially complex environment. In reinforcement learning, an artificial intelligence faces a game-like situation. The computer employs trial and error to come up with a solution to the problem. To get the machine to do what the programmer wants, the artificial intelligence gets either rewards or penalties for the actions it performs. Its goal is to maximize the total reward. The framework is illustrated in Fig. 3.

The basic reinforcement learning algorithm, Q-learning, is to learn a policy $\pi(a|s)$ of an agent interacting with the environment over time. In this chapter, we use s and a denote current state and action at time t, respectively, and use s' and a' denote next state and next action at time $t+1$, respectively. At each time instance t, the interacting is realized by an action $a \in \mathcal{A}$, which brings a reward r_t to the agent and the feedback

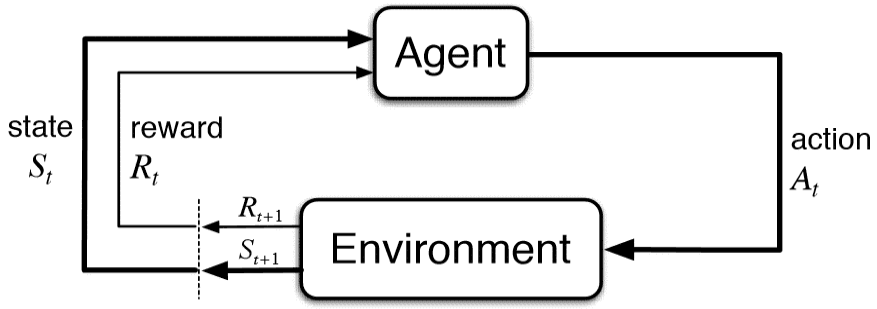


Figure 3: The framework of reinforcement learning.

can be positive as a prize or negative as a penalty. Meanwhile, through different actions, the agent will transit from the current state s to the next state s'. Intuitively, we can define the reward function as $R(s,a)$ and the state transition probability as $p(s'|s,a)$, respectively. This state transition process will not be suspended until the state achieves a terminal state. Over the time, the accumulated reward can be written as

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{3}$$

where γ is a discount factor in the range of $(0,1]$. The purpose of the agent is to maximize the expected long term reward.

The state value function is the expectation of reward from state s adopting policy π, which is $V_\pi(s) = E[R_t|s]$. In order to find the value function of each state, we need to solve the Bellman equation

$$\begin{aligned} V_\pi(s) &= E_{\pi,p}[R(s,a) + \gamma V_\pi(s'|s)] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V_\pi(s')], \end{aligned} \tag{4}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The desirable state value is $V^*(s) = \max_\pi V_\pi(s)$. Then, we define a state-action pair function $Q_\pi(s,a)$, which means the expected reward by action a in state s under the policy π. We can rewrite (4) as

$$\begin{aligned} Q_\pi(s,a) &= E_{\pi,p}\left[R(s,a) + \gamma Q_\pi(s',a')\right] \\ &= \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s') Q_\pi(s',a')\right], \end{aligned} \tag{5}$$

where, a' is the next action. Correspondingly, the objective function turns to be

$$Q^* = \max_{\pi} Q^{\pi}(s, a). \tag{6}$$

The main challenge in reinforcement learning lays in preparing the simulation environment, which is highly dependant on the task to be performed. When the model has to go superhuman in Chess, Go or Atari games, preparing the simulation environment is relatively simple. When it comes to building a model capable of driving an autonomous car, building a realistic simulator is crucial before letting the car ride on the street. The model has to figure out how to brake or avoid a collision in a safe environment, where sacrificing even a thousand cars comes at a minimal cost. Transferring the model out of the training environment and into to the real world is where things get tricky.

1.2 Edge Computing Basics

1.2.1 Background

With the development of information technology in the last twenty years, we step into an era of information explosion. The unprecedented quantity of data and the requests for data processing are generated. Internet of Things (IoT), Cyber-Physical System (CPS), Mobile Internet, etc., have a great contribution to speed data generation. According to the anticipation of Cisco, there will be more than 50 billion devices connected to the Internet by 2020. And the amount of generated data could reach 500 zettabytes, 45 percent of which will be processed or stored in the network by 2019 [2]. Such a kind of massive data bring a significant challenge to data processing and storage. In order to handle the huge workload, cloud computing emerges as a solution [1]. And its architecture is shown in Fig. 4.

The advantages of cloud computing lie in the powerful data processing ability and the huge capacity for storage. But the computation model of cloud computing is a kind of centralized model, which means every job request or task has to be uploaded to the cloud data center for processing therefore easily resulting in cloud server overloaded. In addition, from the perspective of transmission, the long distance and the limited network

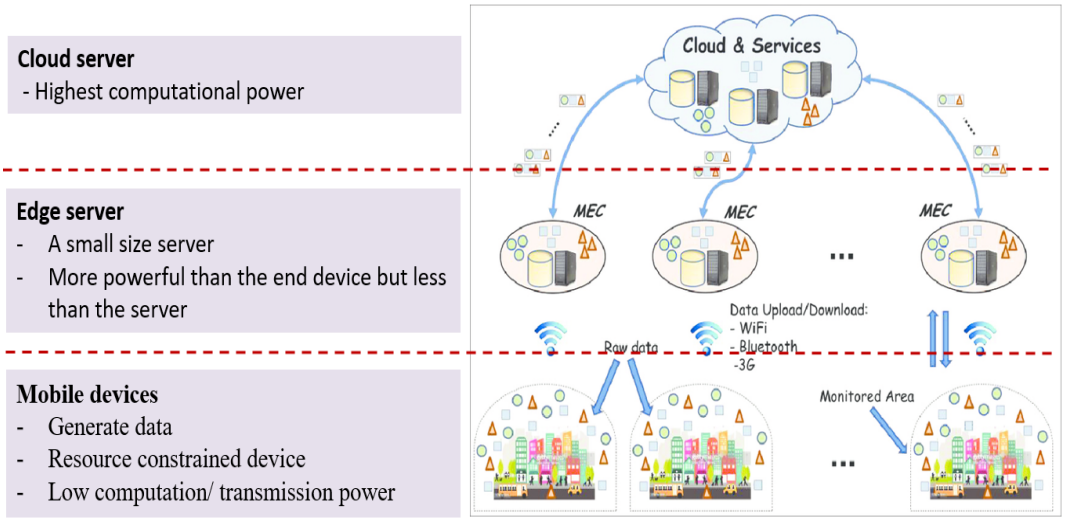


Figure 4: The architecture of edge computing.

bandwidth are still intractable problems [6]. Thus, the latency is an inevitable challenge. In order to surmount this, edge computing is proposed, which employs a hierarchical computation model. The main idea is to geographically distribute the cloud services to the edge of network rather than the cloud center [7]. Therefore, the transmission distance can be significantly reduced. Furthermore, the edge nodes themselves are possessed of data processing abilities to a certain extent. Hence, not all the tasks need to be uploaded to the cloud. Those relatively lower complexity tasks can be dealt with in edge nodes and then send back to the terminal devices so the latency can be further decreased [8]. In the aspect of technology, we can obviously find that the edge computing is more suitable for the real-time missions due to the low latency property, which is conductive to improve the quality of service (QoS).

1.2.2 Architecture of edge Computing

The reference model of edge computing architecture is a significant research topic. In recent years, a number of architectures have been proposed for edge computing. They are mostly derived from the fundamental three-layer structure. edge computing extends cloud service to the network edge by introducing edge layer between end devices and cloud. Fig. 2 shows the hierarchical architecture of edge computing.

The hierarchical architecture is composed of the following three layers:

A Terminal layer: This is the layer closest to the end user and physical environment.

It consists of various IoT devices, for example, sensors, mobile phones, smart vehicles, smart cards, readers, and so on. Specially, though the mobile phones and smart vehicles have the computing power, we only utilize them as the smart sensing devices here. These devices are widely geographically distributed in general. They are responsible for sensing the feature data of physical objects or events and transmitting these sensed data to upper layer or processing and storage [9].

B edge layer: This layer is located on the edge of the network. edge computing layer is composed of a large number of edge nodes, which generally including routers, gateways, switchers, access points, base stations, specific edge servers, etc. These edge nodes are widely distributed between the end devices and cloud, for example, cafes, shopping centers, bus terminals, streets, parks, etc. They can be static at a fixed location, or mobile on a moving carrier. The end devices can conveniently connect with edge nodes to obtain services. They have the capabilities to compute, transmit and temporarily store the received sensed data. The real-time analysis and latency-sensitive applications can be accomplished in edge layer. Moreover, the edge nodes are also connected with cloud data center by IP core network, and responsible for interaction and cooperation with cloud to obtain more powerful computing and storage capabilities.

C Cloud layer: The cloud computing layer consists of multiple high-performance servers and storage devices, and provides various application services, such as smart home, smart transportation, smart factory, etc. It has powerful computing and storage capabilities to support for extensive computation analysis and permanently storage of an enormous amount of data. However, different from traditional cloud computing architecture, not all computing and storage tasks go through the cloud. According to the demand-load, the cloud core modules are efficiently managed and scheduled by some control strategies to improve utilization of the cloud resources [10].

1.2.3 The Characteristics of Edge Computing

A Low latency and real time interactions: edge nodes at the network edge locally acquire the data generated by sensors and devices, process and store data by network edge devices in local area network. It significantly reduces data movement across the Internet and provides speedy high-quality localized services supported by endpoints. Therefore, it enables low latency and meets the demand of real time interactions, especially for latency-sensitive or time-sensitive applications.

B Save bandwidth: edge computing extends the computation and storage capabilities to the network edge to perform data processing and storing between the end nodes and traditional cloud. Some computation tasks, for example, data preprocessing, redundancy removing, data cleaning and filtering, valuable information extraction, are performed locally. Only part of useful data is transmitted to the cloud, and most of the data don't need to be transmitted over the Internet [11].

C Support for mobility: In edge computing scenarios, there are various mobile devices (e.g., smart phones, vehicles, and smart watch) so that the spatial mobility at the terminal layer is frequent, while there are also some end devices remained static, such as traffic cameras. Similarly, edge node in edge layer can also be a mobile or static computing resource platform. It can be deployed in airport and coffee shop, or on the mobile vehicles and trains.

D Geographical distribution and decentralized data analytic: Compared with the more centralized cloud computing, the services and application of edge computing advocate geographical distributed deployment. It consists of large number of widely distributed nodes, which have the ability to track and derive the locations of end devices in order to support the mobility [12]. Instead of processing and storing information in centralized data center far away from end-user, the decentralized architecture of edge computing ensures the proximity of data analytics to the customer. This characteristic can support faster analysis of big data, better location-based services, and more powerful capabilities of real-time decision making.

E Data security and privacy protection: edge computing hosts services closed to end-users. So it has particular advantages in data security and privacy protection. Firstly, it can protect data by encryption and isolation. edge nodes provide access control policy, encryption schemes, integrity check and isolation measures to protect the security of sensitive privacy data. Secondly, it can avoid the risks caused by system upgrade. The remote upgrade of traditional devices is low efficiency, and there are disadvantages such as firmware upgrade lost contact. edge computing does not need Over-the-Air Technology (OTA) firmware upgrade of system, only update the algorithms and micro applications in the edge end [13].

F Low energy consumption: In the edge computing architecture, edge nodes are dispersed geographically. So it will not generate a lot of heat due to concentration, and need not additional cooling system. In addition, short range communication mode and some optimal energy Management Policies of mobile nodes evidently reduce communication energy consumption. This will lead to reducing power consumption, saving energy and decreasing the cost. edge computing provides a greener computing paradigm [14].

1.3 Dissertation Organization

The rest of dissertation is organized as follows. In Chapter 2, we utilize the meta-learning method to predict edge computing resources demand in vehicular networks. In order to minimize the expenses to consume edge computing resources, this chapter proposed a two-stage meta-learning approach. In the first stage, a DNN is utilized to learn the experience dataset so as to figure out which machine learning model performs better in a specific situation. In the second stage, the resource amount anticipation will be conducted by the machine learning model selected by the DNN obtained in the first stage according to the meta-features. In addition, due to the fact that there is no open edge computing based vehicular network dataset, we program in the game engine Unity3D to build the 3D model of Manhattan area as in real world. Meanwhile, we adjust the factors like different roadmaps, the number of vehicles, and the randomness of task sizes for the traffic, which makes our data get closer to the practice. Eventually, we find

that out proposed meta-learning method always gives the most economical predictions, which helps save up to 39.93%, 37.15%, 5.62%, and 70.47% for multi-intersections, roundabout, highway, and bridge scenarios, respectively, compared with the fully real time requests.

In Chapter 3, the edge computing assisted high definition map updating mechanism is discussed. As a latency sensitive application, HD map transmission needs to be performed in a timely manner. This chapter investigates an information staleness minimization problem for federated analytics based HD map generation and layer-wise transmission offloading. Fortunately, emerging MEC provides a low-latency paradigm for data transmission. But the available edge computing resources may be variational practically. Therefore, this chapter discusses two cases, i.e., deterministic edge capacity case and uncertain edge capacity case. For the deterministic edge capacity case, the optimal solution is obtained analytically. And the influence of different edge server capacity, federated analytics accuracy, vehicle utility is also discussed. For the uncertain edge capacity case, the problem is reformulated into a DRCCO optimization problem and solved by the CVaR model based approximation. The experiments demonstrate the CVaR model based approximation is able to find near-optimal solutions with much less time consumption.

In Chapter 4, a low latency purposed framework design for edge assisted federated learning is described. In this chapter, we study the low latency problem for multi-task federated learning in the MEC networks. Considering in large scale IoTs scenario, it is not possible for edge nodes and end devices to obtain the complete information of the other side, which means building the complete preference list is impractical. Therefore, we propose a method to deal with the two-sided many-to-one matching with the incomplete preference list. The simulation results show that the performance of our proposed method is close to the performance of CPL, although there is small gap between them due to information missing. Besides, we also discuss the influence of number of participants, the number of edge nodes, the edge node capacity, local accuracy, energy threshold, and preference list missing rate among network latency. Evidently, the network latency is positively related to the missing rate while is negatively correlated

with number of edge nodes, capacity of edge nodes, energy threshold, and local accuracy indicator.

Finally, some possible future works and conclusions are given in Chapter 5.

2 Edge Computing Resources Reservation in Vehicular Networks: A Meta-Learning Approach

2.1 Introduction

Having stepped into the era of information technology, there are enormous artificial intelligence based autonomous devices, technologies, and services coming into being, one important branch of which is autonomous vehicles or intelligent vehicles. According to the definition of National Highway Traffic Safety Administration (NHTSA), the levels of vehicle automation can be categorized into six classes, which are distinguished by the extent of autonomy [15]. Currently, the performance of autonomous vehicles can just meet the requirements between level 2 and level 3, and both of which require the driver must be ready to take back control at any time. In other words, the artificial intelligence based autonomous driving remains much to be done before realizing the human occupants never need to be involved in driving, such as accurate prediction, precise inference, latency decreasing, etc.

For the time being, the artificial intelligence technologies of autonomous vehicles heavily rely on the data generated by the built-in devices such as an array of sensors, electronic control units, cameras, etc. According to the forecast of Intel, the data generated by one single autonomous vehicle will achieve 4TB data per day [16]. Such a kind of massive data bring inevitable challenges to data processing and storage within vehicles, especially for those real-time tasks with high computation complexity such as collaborative perception, path planning, collaborative simultaneous localization and mapping (SLAM), real-time pedestrian detection, or with high demands for storage capacity like uploading driving records. In this case, cloud computing can be an effective way to help.

Nowadays, there are two mainstream cloud computing paradigms: one is the conventional centralized cloud computing and the other one is edge computing. The superiority of traditional centralized cloud computing is founded on the powerful data processing ability and the enormous storage capacity of remote datacenter. However, as a centralized paradigm, all the data needs to be transmitted to the datacenter for storage or further processing. The latency caused by long transmission distance and

limited bandwidth is a significant challenge for vehicular networks [17]. On the opposite side, the architecture of edge computing performs better for latency alleviation and is more suitable for real-time tasks demanded by autonomous vehicles. Because in edge computing configuration, many edge nodes will be deployed in a geographically distributed manner within the network, which means the services can be provided to the users more closely. Moreover, edge nodes are capable of computation ability and storage capacity to a certain extent, which makes it feasible to perform tasks locally and feedback the results to vehicles on time. Accordingly, transmission latency can be reduced significantly and quality of service (QoS) can be remarkably improved [18].

From the commercial perspective, the investment for edge nodes deployment will increase the expenditures of services-providers correspondingly, resulting in a relatively expensive price of edge services [19]. Meanwhile, aiming at catering to the market demands and attracting more customers, the corporations carry out some marketing strategies, one of which is that the services can be sold in a reservation or subscription way with a cheaper price and a real-time requested way with an expensive price. Furthermore, different purchasing programs are supplied and the customers can decide which program to get enrolled in according to their own consumption characteristics. For instance, Amazon Web Services (AWS) provides several purchase schemes to customers, which are pay-as-you-go, save when you reserve, and pay less by using more [20]. Concretely, these programs are described as the following:

1. Pay-as-you-go: the customers can adjust the services demands at any time depending on the their own needs and only need to pay for services on an as needed basis.

2. Save when you reserve: for the service like the Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Relational Database Service (Amazon RDS), if the customers reserve in advance, a certain percent discount will be provided. In addition, if paid with upfront payments, the customers will be charged further less.

3. Pay less by using more: this is a common marketing strategy, which means for services like Amazon simple storage services (S3), if more volumes you consume,

the less you will pay per GB.

Obviously, it is a more economical way to consume services by reservation in advance and the customers can save up to 75% compared with equivalent on-demand scheme based on the AWS's description. Therefore, for the purpose of minimizing the expenditure, it is of great importance for customers to figure out how many resources, i.e., computational memory or storage memory, should be reserved. Intuitively, this expense minimization problem can be regarded as a prediction problem. However, the amount of edge resource consumption actually is closely related to many factors, such as speed, road map, task type, etc. While traditional optimization algorithms are not effective to address such high-dimensional nonlinear regression problems [21]. Fortunately, the emerging machine learning methods provide powerful tools to tackle with such prediction problems. Due to the diversity of influence factors in vehicular networks, we can hardly find one machine learning model which is suitable and performs the best in any scenarios [22]. Considering there are similarities in these scenarios, meta-learning can be an effective method to help, whose goal is to learn from the experience or prior knowledge that generalizes well to related new tasks. Therefore, we propose a two-stage meta-learning based approach to adaptively choose the appropriate machine learning algorithms according to the meta-features extracted on databases.

2.2 Related Work

As a promising distributed computation paradigm, edge computing has become a popular field of research. However, most existing papers in an edge computing scenario focus on the technology perspective of the edge computing network side, like network architecture improvements, edge nodes placement, edge-assisted tasks offloading, etc. [23] proposes a fine-grained collaborative offloading strategy with caching enhancement scheme to minimize the latency at the edge side in both femto-cloud mobile network scenario and mobile edge computing scenario. [24] proposes a method based on the Lagrangian heuristic algorithm and workload allocation scheme to optimally place the cloudlets, under the considerations of both cloudlet cost and average end-to-end delay in a mobile edge computing scenario. [25] proposes a novel communication scheme to

Table 1: Typical Edge Computing Companies and Products

Company Name	Typical Product	Main Functionalities
Amazon	AWS Greengrass	Pushing local computing, communication, caching, sync, and machine learning inference to edge devices
Clear Blade	Edge Software	Deploying and managing IoT systems on the edge, and communicating with on-premise devices
Cisco	Gateway IC3000	Providing built-in security and manageability, enabling faster decision making at the edge
Dell EMC	VMware Cloud	Extending public cloud benefits to workloads in both private datacenter and edge locations
FogHorn	Lightning	Offering rapid data ingestion, sensor fusion, and machine learning to generate actionable insights in real-time
HPE	Edgeline	Aggregating and filtering data, analyzing video streams, and translating industrial protocols
IBM	Waston IoT	IoT security, data analyzing, IoT management, and IoT machine learning application
Microsoft	Azure IoT Edge	Offloading artificial intelligence and analytics workloads to the edge
Rigado	Cascade	Edge infrastructure establishment and connecting devices in Commercial IoT environments
Saguna	vEdge	Providing virtualized resources to enable on-edge applications and real-time network visibility

enable the low-latency, robust and accurate edge node assisted self-driving service for connected autonomous driving services in a mobile edge computing scenario. Different from these literature, this chapter stands on the side of edge resources consumers and focuses on the economic aspect to minimize the expenditure for customers. Therefore, in this chapter, the specific architecture of edge computing network, the limitation of edge resources, and the wireless communications are unconcerned.

Thanks to the low-latency and distributed properties of edge computing, diverse use cases can be supported, such as connected autonomous vehicles, e-health, industrial automation, mobile gaming, smart grid, and Internet of Things (IoT) services. Correspondingly, such characteristics and capabilities offered by an edge computing platform can be translated into unique value and revenue generation [26], which also prompts

some researches on commercial aspect of edge computing. Present typical edge computing companies are summarized in Tab. 1. [27] discusses the influences of five factors, i.e., ease of use, security, cost reduction, reliability, and collaborating, on the cloud usages of micro and small businesses. [28] introduces a framework that leverages pricing aspects to enable the sharing economy vision for edge devices applied into the smart city scenario. [29] compares several pricing models, such as the time based model, volume based model, flat rate, content based model, etc., and discusses the pricing schemes from different cloud services providers including AWS EC2, AWS S3, Microsoft Azure, and AppNexus. Whereas, these works analyze and discuss problems on from the perspectives of service providers and the market operations instead of service customers as well.

As a promising machine learning approach, meta-learning has attracted considerable interests of diverse science and engineering communities recently, and is widely used in different fields. [30] proposes a meta-learning based framework to learn the online learning algorithm from offline videos so as to address an object tracking problem. [31] proposes a meta-learning based method to tackle with an automatic text classification problem through utilizing the distance-based meta-features derived from the original bag-of-words representation. [32] proposes a novel meta-learning method for domain generalization by a model agnostic training procedure so that the domain shift problem can be avoided. [33] proposes a two deep neural network architecture based meta-learning strategy to solve the clod-start problem for item recommendations when new items arrive continuously. However, to the best of our knowledge, there is no existing literature that implements a meta-learning based method to deal with an edge resource reservation problem in vehicular networks.

2.3 Scenario Description and Problem Formulation

In this chapter, we consider an edge computing platform deploying edge nodes and providing edge computing services. The computational resources or storage resources can be purchased via reservation or real-time request. The automotive company provides services to the autonomous vehicle customers, such as collaborative perception,

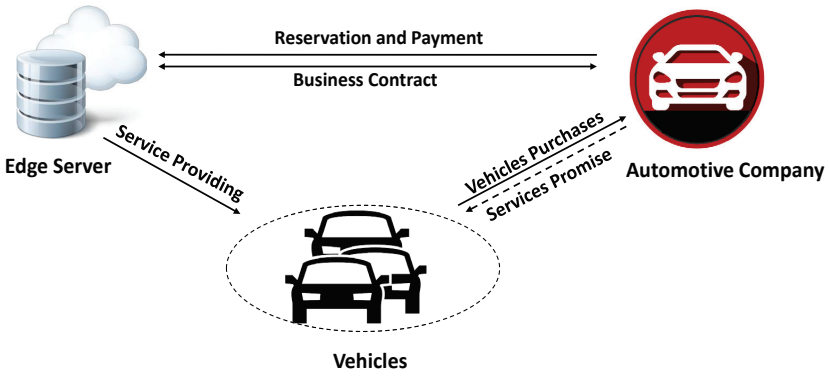


Figure 5: The scenario and procedures description.

path planning, collaborative simultaneous localization and mapping, real-time object detection, etc. Consequently, it is inevitable for automotive company to pay the rental fee or operation fee to the edge platform so as to offer the corresponding services to customers. The procedures are described in Fig. 5.

We define the unit price for reservation and real-time request as P_{rv} and P_{rt}, respectively. To perform some computation-intensive or storage-demanding tasks, the practical needed amount for edge resources is n units. While the total amount reserved by automotive company is m units. Intuitively, only when $n = m$, the consumption expenditure is optimally minimized. Because when $n > m$, the reserved amount of resources cannot meet the usage requirements and the extra real-time purchasing is inevitable. When $n < m$, the reserved amount of resources exceeds the actual demand, which actually is a waste for both customers and services provider. Therefore, the total cost of consuming edge services S can be described by the following function

$$S = \begin{cases} m \times P_{\mathrm{rv}}, & n \leq m, \\ m \times P_{\mathrm{rv}} + (n - m) \times P_{\mathrm{rt}}, & m < n. \end{cases} \tag{7}$$

Correspondingly, the total waste W can be calculated as the following piece-wise function

$$W = \begin{cases} (m - n) \times P_{\mathrm{rv}}, & n \leq m, \\ (n - m) \times P_{\mathrm{rt}}, & m < n. \end{cases} \tag{8}$$

Obviously, in order to minimize the waste, the desirable situation is $m = n$. Thereby,

the objective function can be defined by mean-square-error (MSE)

$$\min \|m - n\|_2^2 . \tag{9}$$

2.4 Methodology

In this section, we firstly introduce our proposed two-stage meta-learning approach. Then, the machine learning models utilized in this chapter are introduced as well.

2.4.1 Meta-Learning

Meta-learning is also known as learning to learn, which is not specifically defined but covers any kind of machine learning methods based on prior knowledge learned from other related tasks. One of the typical meta-learning problems is the algorithm selection problem (ASP). [34] formulates the classical ASP and proves that there is a connection between the problem characteristics and the algorithm which can be utilized to solve it. Later, [35] further demonstrates there will be no one single algorithm that can work out optimums for all the problems. In other words, the algorithm performs differently from problem to problem, or more specifically, from dataset to dataset. However, generally, ASPs are NP-hard, which are challenging to be resolved by traditional optimization methods [36]. Therefore, we propose a two-stage meta-learning approach to address this problem. Before looking into the details, some preliminary definitions and descriptions are introduced here. In this work, the meta-learning task is made up by the following components:

- The task or problem space $\mathcal{P}$: a set of problems that contains both solved and unsolved ones. Essentially, the dimension of $\mathcal{P}$ is high due to the diversity of characteristics embedded in the tasks, which is the factor that leads to the performance fluctuation of same model among different problems. In this work, $\mathcal{P}$ indicates the edge resource prediction problems in vehicular networks under different scenarios.

- The meta-feature space $\mathcal{C}$: a set of data characteristics reflecting the internal or intrinsic representations of a dataset. Basically, $\mathcal{C}$ is a multi-dimensional vector mathematically. The specific types of meta-features are decided by the problem

and the selected machine learning models, the typical forms of which include simple meta-features (such as the number of samples, the number of classes, etc.), statistical meta-features (such as skewness, kuitosis, covariance, correlation, etc.), information-theoretic meta-features (such as norm entropy, mutual information, uncertainty coefficient, etc.), complexity based meta-features (such as Fisher's discriminant, volume of overlap, data consistency, etc.), model based meta-features (like for decision tree, the number of leaves, branch length, information can be the meta-features), and landmarkers, etc [37]. The specific meta-features utilized in this work are introduced in details in Section 2.5.1.

- The machine learning model or algorithm space $\mathcal{M}$: basically it can be the universal set which contains all the existing algorithms. Whereas, practically, when considering one specific problem, $\mathcal{M}$ can only be a set of appropriately selected ones. In this work, because the tasks are time series data based prediction problems, $\mathcal{M}$ is defined as a group of long short term memory (LSTM) based machine learning models, which is detailedly introduced in Section 2.4.2.

- The performance evaluation space $\mathcal{E}$: is a set of diverse metrics to assess the performance of algorithms in $\mathcal{M}$ on a dataset. The evaluation metrics implemented in this work are discussed concretely in Section 2.5.2.

For each task $\mathcal{T}_i$ in $\mathcal{P}$, what is supposed to be learned the distribution over the dataset, which can be described as $p(\mathcal{T}_i)$. In order to find the best regression model, the traditional method is to apply all the algorithms in $\mathcal{M}$ to $\mathcal{T}_i$, and then perform ranking calculations to determine which specific algorithm should be the one [38]. However, one defect of this approach is that, each time when dealing with a new but related task, calculating over the space $\mathcal{M}$ is time-costing. Meanwhile, the prior-knowledge concealed in history experiences are not well exploitative. Therefore, we propose a two-stage meta-learning method to solve this problem, which is illustrated in Fig. 6. On the first stage, we utilize a deep neural network (named as Decider) to figure out which algorithm should be selected according to the experiences. On the second stage, the chosen machine learning model (named as Prognosticator) is implemented to perform the inference for edge resource consumption.

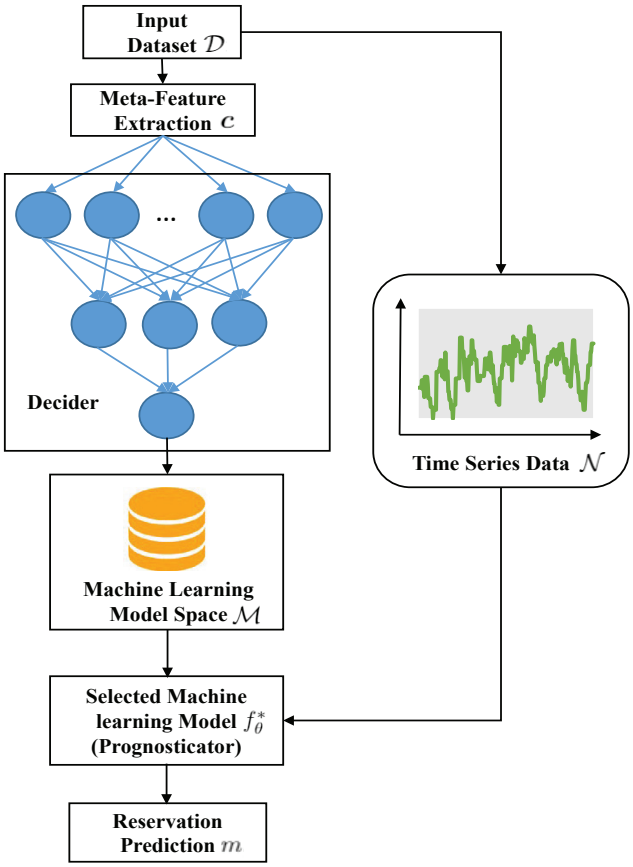


Figure 6: The proposed meta-learning framework.

Basically, each algorithm in $\mathcal{M}$ can be denoted as f_{θ_i}, where f is the function that can represent the i^{th} algorithm in $\mathcal{M}$ and θ describes the corresponding parameters determined by the machine learning model configurations. Intuitively, the objective function or loss function of the Decider can be written as

$$\begin{aligned} &\min_i \mathcal{L}_1\left(f_{\theta_i}\right) = \|f_{\theta_i}(\boldsymbol{c}) - r\|_2^2, \\ &s.t.\ f_{\theta_i} \in \mathcal{M}, \\ &\boldsymbol{c} \in \mathcal{C}, \end{aligned} \tag{10}$$

where $\boldsymbol{c}$ is the meta-feature vector and r is the suggested algorithm according to history experiences which gives the most economical scheme. The architecture of the Decider is a fully-connected deep neural network, which means the output of each hidden layer

will be the input of the following hidden layer. Thus, to optimize the network, back-propagation is inevitable operation through the network, which can be described as,

$$\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\delta}^{(l)}} = \sum \left(\frac{\partial \mathcal{L}_1}{\partial o^{(l)}} \right) \frac{\partial o^{(l)}}{\partial o^{(l-1)}} \frac{\partial o^{(l-1)}}{\boldsymbol{\delta}^{(l)}}, \tag{11}$$

where l denotes the number of hidden layers of the Decider, $\boldsymbol{\delta}^{(l)}$ describes the generalized parameter set (including weight and bias) in layer l, i.e., $\boldsymbol{\delta}^l = \{\boldsymbol{w}^{(l)}, \boldsymbol{b}^{(l)}\}$, and $o^{(l)}$ denotes the output of layer l. During each iteration, the parameters of all the layers will get updated by

$$\boldsymbol{\delta}^{'} = \boldsymbol{\delta} - \alpha \nabla_{\boldsymbol{\delta}} \mathcal{L}_1, \tag{12}$$

where α is the learning rate. When the number of iterations or the value of loss function achieves the threshold, the training phase ends and the optimal algorithm f_θ^* or the Prognosticator can be obtained. Until here, all the operations in stage one have been finished.

The second stage is to exploit the Prognosticator obtained in stage one to predict the edge resource consumption. In this stage, what needs to be done is to optimize the parameters of Prognosticator so as to fit the new dataset $\mathcal{N}$ from $\mathcal{T}_i$. Pre-requisitely, it is necessary to divide $\mathcal{N}$ into training set and testing set, which are denoted as $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Therefore, the objective function of Prognosticator can be defined as

$$\min_{\boldsymbol{\phi}} \ \mathcal{L}_2 \left(f_\theta^* \left(\boldsymbol{\phi} \right) \right) = \sum_{(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{T}_i} \| f_\theta^* \left(\boldsymbol{\phi}; \boldsymbol{x} \right) - \boldsymbol{y} \|_2^2, \tag{13}$$

where $\boldsymbol{\phi}$ is the parameters set for the machine learning model f_θ^*. To minimize the error, likewise, the Prognosticator will perform the back-propagation, i.e.,

$$\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\phi}^{(p)}} = \sum \left(\frac{\partial \mathcal{L}_2}{\partial o^{(p)}} \right) \frac{\partial o^{(p)}}{\partial o^{(p-1)}} \frac{\partial o^{(p-1)}}{\boldsymbol{\phi}^{(p)}}, \tag{14}$$

where p denotes the number of hidden layers in the Prognosticator. For step in (14), the parameters $\boldsymbol{\phi}$ will be updated by

$$\boldsymbol{\phi}^{'} = \boldsymbol{\phi} - \beta \nabla_{\boldsymbol{\phi}} \mathcal{L}_2, \tag{15}$$

Algorithm 1 The Proposed Two-Stage Meta-Learning

Input: experience data $\mathcal{D}$; meta-feature space $\mathcal{C}$; machine learning model space $\mathcal{M}$; new task $\mathcal{T}_i$; new task dataset $\mathcal{N}$; learning rates α and β; maximum iteration steps for stage one $t_{\max}$ and stage two $k_{\max}$, respectively.
Randomly initialize the parameter $\boldsymbol{\delta}$.
for t=1:$t_{\max}$ **do**
    Feed forward propagate all the samples in $\mathcal{D}$ and calculate the MSE by (10);
    Back propagate MSE through the network by applying (11) and update the values of parameter set $\boldsymbol{\delta}^{'}$ via (12);
end for
Output: the trained Decider with optimal parameters $\boldsymbol{\delta}^{*}$;
Feed the meta-features $\boldsymbol{c}_i$ from $\mathcal{T}_i$ into the Decider and the specific Prognosticator, i.e., f_{θ}^{*}, can be obtained;
Divide $\mathcal{N}$ into training set $\boldsymbol{x}$ and testing set $\boldsymbol{y}$;
Feed $\boldsymbol{x}$ into f_{θ}^{*} with randomly initialized parameters $\boldsymbol{\phi}$;
for k=1:$k_{\max}$ **do**
    Feed forward propagate all the samples through $\boldsymbol{x}$ and get the MSE by (13);
    Back propagate MSE throughout the network f_{θ}^{*} by applying (14) and update the parameter set $\boldsymbol{\phi}^{'}$ via (15);
end for
Output: the Prognosticator f_{θ}^{*} with optimal parameters $\boldsymbol{\phi}^{*}$;
Feed the testing data $\boldsymbol{y}$ into $f_{\theta}^{*}(\boldsymbol{\phi}^{*})$ and infer the amount of edge resource consumption;

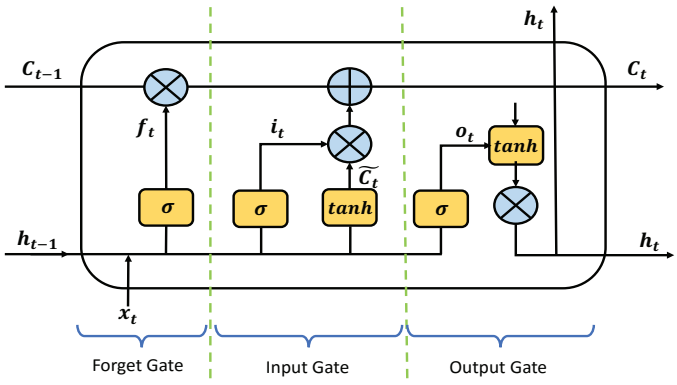


Figure 7: The structure of a LSTM cell.

where β is the learning step size. Overall, the procedures are summarized in Algorithm 1.

2.4.2 Machine Learning Model Space

In this section, the algorithm space $\mathcal{M}$ are concretely introduced. With a time series dataset, in this work, we exploit several LSTM based machine learning models, which are introduced in details as the following.

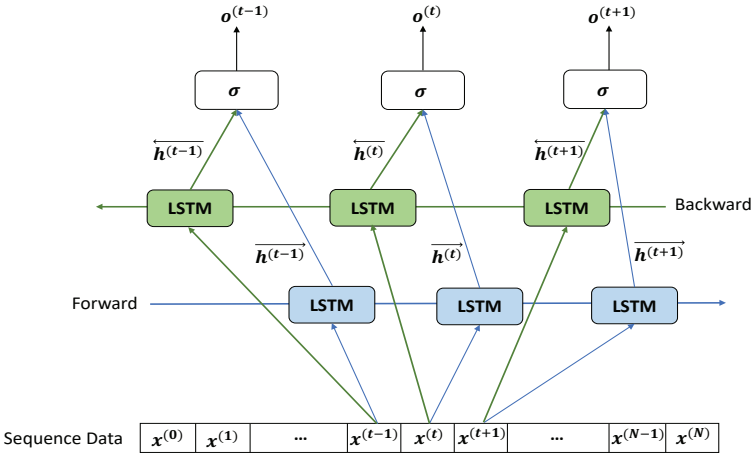


Figure 8: The architecture of BiLSTM network.

2.4.3 LSTM Cell

LSTM is developed from recurrent neural networks (RNNs), which is a kind of artificial neural networks (ANNs) with recurrent connections. For ANNs, one key assumption is that all the outputs or inputs are independent to each other. However, with the help of recurrent connections, a RNN is able to execute same task for every element from a sequence with the output being depended on the previous computations, which makes it suitable for utilizing sequential data to perform sequence recognition or prediction problem [39]. Whereas, one drawback when a RNN performs back-propagation to optimize the parameters is the gradient vanishing or gradient explosion problem due to the sequential multiplication of $\tanh'$ [40]. Aiming at solving this problem, LSTM is proposed through introducing the gates. The structure of one LSTM cell consists of three main parts: forget gate, input gate, and output gate, which is illustrated in Fig. 7. At each time t, the calculation equations are as follows:

$$f^{(t)} = \sigma\left(\omega_f\left(h^{(t-1)}, x^{(t)}\right) + b_f\right), \tag{16}$$

$$i^{(t)} = \sigma\left(\omega_i\left(h^{(t-1)}, x^{(t)}\right) + b_i\right), \tag{17}$$

$$\widetilde{C}^{(t)} = \tanh\left(\omega_c\left(h^{(t-1)}, x^{(t)}\right) + b_c\right), \tag{18}$$

$$o^{(t)} = \sigma\left(\omega_o\left(h^{(t-1)}, x^{(t)}\right) + b_o\right), \tag{19}$$

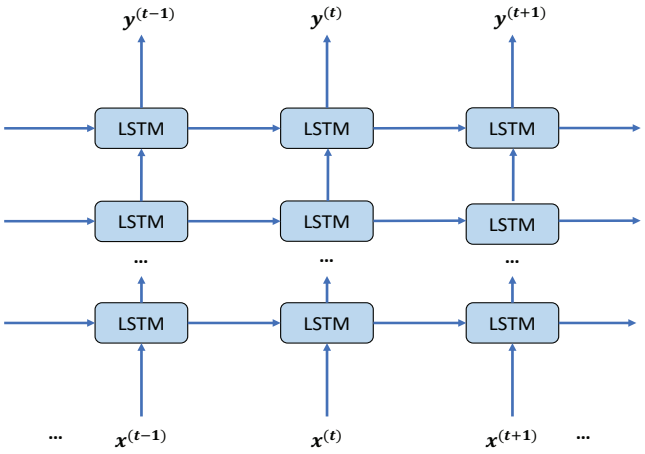


Figure 9: The structure of a stacked LSTM network.

and

$$h^{(t)} = \tanh\left(c^{(t)}\right) * o^{(t)}, \tag{20}$$

where σ is sigmoid function; $f^{(t)}$, $i^{(t)}$, $o^{(t)}$ are the value of forget gate, input gate and output gate, respectively, and the corresponding ω and b are weight and bias; $h^{(t)}$ is the hidden state. When performing back-propagation among LSTM, we can obtain

$$\begin{aligned}\frac{\partial C^{(t)}}{\partial C^{(t-1)}} &= C^{(t-1)}\sigma^{'}(\cdot)\omega_f * o^{(t-1)}\tanh^{'}\left(C^{(t-1)}\right) \\ &+ \widetilde{C}^{(t)}\sigma^{'}(\cdot)\omega_i * o^{(t-1)}\tanh^{'} \\ &+ i^{(t)}\tanh^{'}(\cdot) * o^{(t-1)}\tanh^{'}\left(C^{(t-1)}\right) + f^{(t)}.\end{aligned} \tag{21}$$

For k steps back-propagation, the derivative shown in (21) will be multiplied over k times. If the LSTM architecture is sufficiently large, e.g. $k \to \infty$, when the network starts approaching to converge to zero, we can adjust the value of $\frac{\partial C^{(t)}}{\partial C^{(t-1)}}$ closer to one, like 0.97, through controlling the output value of forget gate $f^{(t)}$ [41]. In this way, the gradient vanishing problem can be avoided, which is also the reason for why we only focus on those LSTM based models in this work.

2.4.4 BiLSTM

BiLSTM is inspired by the directional RNN and is proposed by [42]. Conventional RNN or LSTM processes series data in a forward direction or in time order. However, for BiLSTM or bidirectional RNN, both forward and backward direction information are

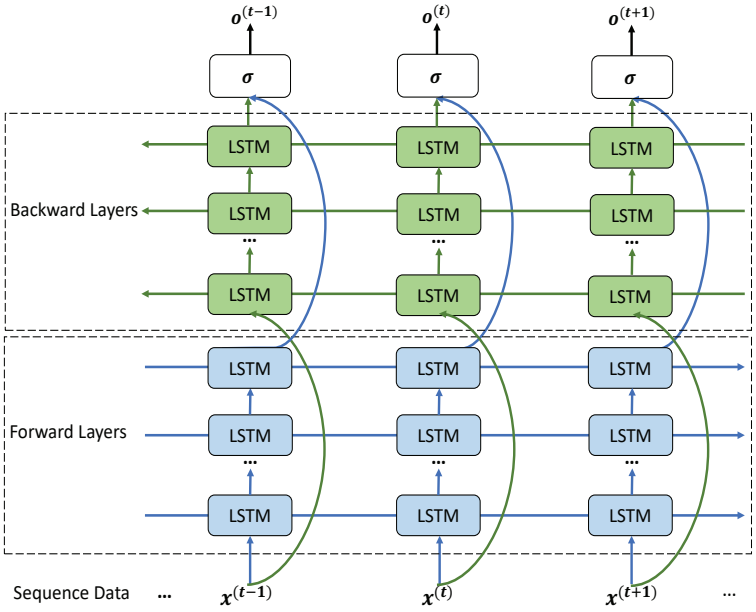


Figure 10: The structure of a Stacked BiLSTM network.

utilized to process the sequence data through two independent LSTM or RNN layers [43]. The structure of the BiLSTM network is illustrated in Fig. 8.

As we can see, the overall structure of a BiLSTM can be divided into four layers: input layer, forward layer, backward layer, and output layer. The functionality of input layer is intuitive, which feeds the series data into the network. The forward layer calculates $\overrightarrow{h^{(t)}}$ chronologically, in other words, from $t = 0$ to $t = N$. While the backward layer calculates $\overleftarrow{h^{(t)}}$ unchronologically, i.e., from $t = N$ to $t = 0$. For both forward and backward layers, LSTM is the basic element for configuration. All the formulations and structures inside these LSTM cells are totally the same as what is discussed in Subsection 2.4.3. The output layer utilizes the output of forward and backward layers to calculate the current output through a sigmoid activation function. Therefore, unlike the output of LSTM in (19), the final output of the BiLSTM network can be expressed as

$$o^{(t)} = \sigma\left(\omega_o\left(x^{(t)}, \overrightarrow{h^{(t)}}, \overleftarrow{h^{(t)}}\right) + b_o\right). \tag{22}$$

2.4.5 Stacked LSTM and Stacked BiLSTM

In the field of machine learning, there is a kind of model named as deep neural networks (DNNs), which is developed from ANNs. Generally, compared with ANNs, a DNN is possessed of a deep architecture, which has more than one hidden layers. With

multiple layers, a DNN is able to extract high-level and more essential representations so as to fitting a high-dimensional non-linear model [44]. This idea works for LSTM and BiLSTM as well. Based on the shallow structure, a deeper model can be built by adding more hidden layers. The names of deep LSTM and BiLSTM model are stacked LSTM and stacked BiLSTM, respectively, and the corresponding architectures are shown in Fig. 9 and Fig. 10, respectively.

The architecture becomes hierarchical paradigm and the output function needs some modifications accordingly. For stacked LSTM, in layer l, the input is the ouput of the last layer, which can be described as

$$h_l^{(t)} = \omega_{(l-1,l)} h_{l-1}^{(t)} + b_l. \tag{23}$$

Suppose the number of hidden layers is M, the output of the network is

$$o^{(t)} = \sigma\left(\omega_o\left(h_M^{(t)}\right) + b_o\right). \tag{24}$$

Similarly, the output of a stacked BiLSTM network can be expressed as

$$o^{(t)} = \sigma\left(\omega_o\left(\overrightarrow{h_M^{(t)}}, \overleftarrow{h_M^{(t)}}\right) + b_o\right). \tag{25}$$

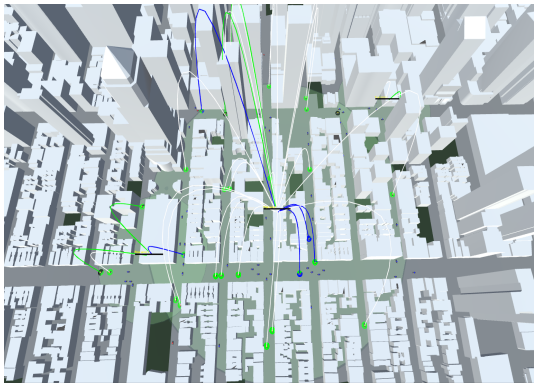
Although a deeper network can fit a complex non-linear model well, it also easily brings over-fitting problem. Over-fitting means the model is exactly suitable for the training dataset while performs poorly upon test dataset. To tackle with this problem, one effective method is to adopt the dropout strategy. The main idea of dropout is to randomly ignore some hidden LSTM units by a certain percentage during the training phase. The selected hidden units will not update the parameters in the back-propagation process. But for the other units, they will optimize the parameters normally according to the back propagated error. By this way, the model can never fit the training dataset perfectly so as to avoid the over-fitting problem [45]. Therefore, based on these two deep models, we add dropout strategy during the training phase, which are named as S-LSTM with dropout (S-LSTM-D) and S-BiLSTM with dropout (S-BiLSTM-D), respectively.

Figure 11: The 3D model of Manhattan area built in Unity.

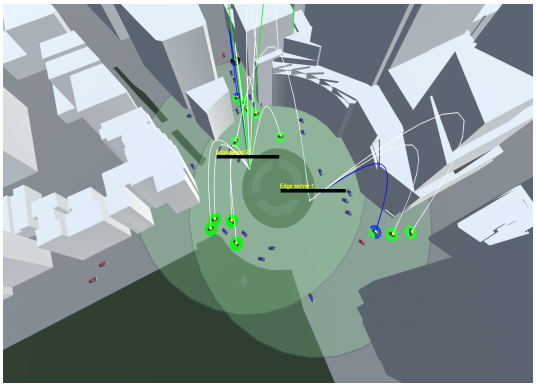
2.5 Simulation Results

2.5.1 Data Generation

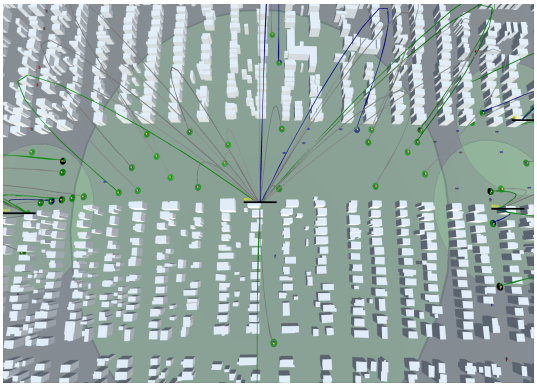
There are many different edge computing resources in reality, such as computation memory, computation power, etc. In this work, we focus on the computation memory consumption. In order to obtain the memory utilization data in diverse traffic situation, we build our own simulation environment through implementing the game engine Unity3D [46]. We build up the 3D model of real-world Manhattan area as the geographical background, which is shown in Fig. 11, where the red dots indicate the locations of edge servers and the green dashed boxes represent the simulation areas. The traffic AI package is implemented to yield traffic flows. Vehicle models are generated in specified areas and move forward following the pre-defined road network obeying traffic rules. In addition, we adjust the total number of vehicles appear in the roadside unit (RSU) coverage area to simulate peak period and off-peak period traffic. Edge server model and vehicle receiver model are built to simulate edge computing assisted computation execution behavior on both units when vehicle receiver is located within the connection range of edge server. Every offloaded computation task will consume a certain amount of memory in the edge server during the period of the task execution. Therefore, we can obtain the memory utilization data by documenting the memory consumption of the edge server. Task sizes are defined in the vehicle receiver model to simulate offloading computation with different data size. Accordingly, a larger data size will occupy more



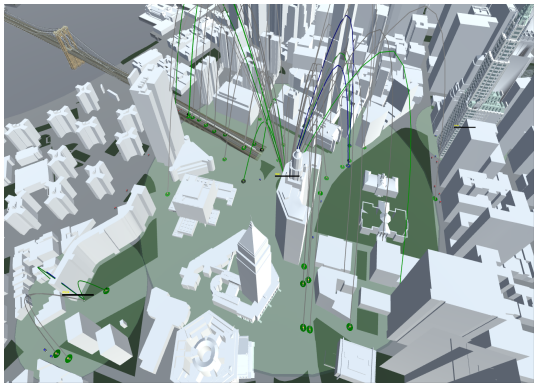
(a) The multi-intersection scenario.



(b) The roundabout scenario.



(c) The highway scenario.



(d) The bridge scenario.

Figure 12: The different scenarios built in Unity.

memory capacity on edge server, resulting in taking longer time to be computed or processed, and vice versa. Vehicle receiver model is attached to the vehicle model of the traffic AI package, and edge server models are placed in the area, where is considered to perform simulated computation offloading service for all vehicles within the range of the edge server.

For the purpose of enriching the scenarios, in this work, we choose four different road maps to simulate traffic flow, i.e., multiple intersections, roundabout, highway, and

Table 2: Parameters Setting for Different Road Maps

Road Map	Amount of Vehicles			Task Size Range	
	Small	Medium	Large	Small	Large
Multi-Intersections	30	60	100	[18,22]	[36,44]
Roundabout	20	35	50	[18,22]	[36,44]
Highway	30	60	100	[18,22]	[36,44]
Bridge	45	90	150	[18,22]	[36,44]

bridge areas, which is illustrated in Fig. 12. As is shown in Fig. 12, edge servers are located right below the horizontal black bars, surrounded by a green circle representing the coverage range of each edge server. The spline connecting vehicles and edge servers represents the connection link, with different color represents different ongoing operation (green for uploading, white for processing, and blue for downloading). The small dot on top of the vehicles is the vehicle receiver model, and the color of the dot stands for the status of the receiver (blue for task complete, yellow for task in process, and red for disconnected). Besides, the larger green or blue disk attached to the bottom of vehicles represents upload and download progress. Apart from creating different scenarios, we also vary the task size in vehicle receiver model and number of vehicles utilizing the driving AI package. The number of cars for multi-intersection, roundabout, highway, and bridge are set as {30, 60, 100}, {20, 35, 50}, {30, 60, 100}, and {45, 90, 150}, respectively. For the small data size situation, the data size will be randomly generated within the range of $[18, 22]$ (MB). And for big data size situation, the data size will be randomly generated within the rage of $[36, 44]$ (MB). Therefore, for each road scenario, there are six dataset in total with the combination of different data size and number of cars and these three factors constitute the meta-feature space $\mathcal{F}$. For convenience, we name the dataset in the format "Scenario-number of vehicles-datasize". For example, Roundabout-20-B means the data is generated in roundabout scenario with 20 vehicles and the data size is big. The details of settings for variation of situations can be summarized in Tab. 11.

2.5.2 Simulation and Evaluation

For the performance evaluation metrics space $\mathcal{E}$, we adopt the suggestions in [47] and use three measurements totally, i.e. root mean square error (RMSE), mean absolute

Table 3: Parameters Settings for $\mathcal{M}$

Models	Hidden Layer	Units	Dropout Percentage
LSTM	1	64	-
BiLSTM	1	64	-
S-LSTM	3	32-32-32	-
S-BiLSTM	2	32-32	-
S-LSTM-D	3	32-64-32	10%
S-BiLSTM-D	2	64-64	10%

percentage (MAP), and mean GEH (MGEH), whose definitions are

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - Y_i)^2}, \tag{26}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - Y_i|}{Y_i}, \tag{27}$$

and

$$\text{MGEH} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{2\,(y_i - Y_i)^2}{y_i + Y_i}}, \tag{28}$$

respectively, where y_i is the predicted value and Y_i is the ground truth. Therein RMSE and MAPE are widely used statistical metrics. While GEH is named by the initials of creator, Geoffrey E. Havers, and has no special mathematical or statistical meaning. Even so, GEH has been approved to be an effective measurement for a variety of traffic analysis purpose and a smaller GEH value indicates a better regression of observed flows[48]. In addition, as is discussed in Section 2.4.2, $\mathcal{M}$ contains six models, i.e., LSTM, BiLSTM, S-LSTM, S-BiLSTM, S-LSTM-D, and S-BiLSTM-D. The parameters settings are as shown in Tab. 3. The simulations are performed under TensorFlow framework and the GPU version is NVIDIA 1080Ti.

Among the three metrics, it is obvious that they keep consistent to each other actually. Checking throughout the table, one conclusion is that LSTM performs better in the majority cases instead of those deep models. Generally, deep network is supposed to work better than shallow networks, like the performance difference between DNNs and ANNs, however, which is not the case here. The main difference is, in DNNs or ANNs, a vital assumption is the data samples are independent to each other. Whereas,

for time series data, one significant property is time dependency, which means the information comes from previous LSTM makes more sense than the information transmitted between hidden layers. In addition, in the stacked architecture, if the previous layer has already made wrong prediction, the next layer will continue forecasting based on the incorrect results, which means errors will be transmitted and enlarged. Besides, time dependency can also explain why the dropout strategy is not suitable here. Since the information that each LSTM brings may be of great importance for cells in the later series. Dropout strategy will erase the randomly selected units so that anticipation cannot be calculated accurately. Moreover, for time series data, the input can be as few as even two dimensional, i.e., time and value. Unlike high dimensional cases, overfitting is not that common. Also, comparing the bidirectional model with unbidirectional model, generally, the performances are similar. But there are still some numerical differences between them, which is because if the forward information is different from backward information, it is easy to introduce bias in the output phase.

Based on the history results, we generate the data for model selection recommendations. The data is four-dimensional. The first three features are roadmap, the number of vehicles, and task size. And the rest one is the suggested model. With the foundation of this dataset $\mathcal{D}$, we build a two-hidden layer DNN to perform the model selection problem, or we can say, a classification problem. The hidden units in each hidden layer is 4. The loss function is defined as MSE as well and the gradient descent methods is also Adam. In addition, according to the evaluation from [49], the peak time of traffic happens averagely from 6am to 9am and 4pm to 7pm. Therefore, when we assess the performance on a roadmap basis and quantitative analysis, we add different weights to the values. The datasets with a larger number of vehicles will get 25%, as is regarded as peak time. The two hours around peak time are defined for medium number of vehicles, which is 33%. The rest 42% goes for the datasets with small amount of vehicles, which is considered as off-peak time. In addition, for the percentage of different task size, it is divided equally. Finally, the final scores obtained by our meta-learning method and the non-meta methods are summarized in Tab. 4, which is statistically calculated among the different roadmaps. Obviously, we can find that our proposed method always achieves

Table 4: Scores Obtained by Proposed Meta Method and Non-meta Methods

Roadmap	Multi-Intersections			Roundabout		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
Meta-learning	**37.91**	**9.13**	**1.51**	**19.42**	**8.56**	**1.35**
LSTM	44.26	9.56	1.68	19.76	9.03	1.37
BiLSTM	69.71	11.78	2.41	38.42	17.14	2.79
S-LSTM	111.28	17.15	3.79	38.71	17.40	2.82
S-BiLSTM	64.16	14.21	2.49	40.70	26.28	3.15
S-LSTM-D	87.57	16.20	2.89	37.87	22.64	2.72
S-BiLSTM-D	62.05	13.14	2.16	38.98	22.79	2.99
Roadmap	Highway			Bridge		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
Meta-learning	**37.23**	**11.64**	**1.63**	**46.63**	**9.31**	**1.49**
LSTM	43.92	12.65	2.72	52.63	10.03	1.72
BiLSTM	44.50	13.57	2.45	74.45	13.67	2.36
S-LSTM	60.79	20.08	3.57	78.98	22.96	2.73
S-BiLSTM	59.26	22.41	2.84	75.99	14.63	2.42
S-LSTM-D	79.20	25.55	3.58	95.27	16.08	3.06
S-BiLSTM-D	64.55	16.32	2.74	93.53	16.56	2.76

the best evaluation scores among all the roadmaps.

Besides, we conduct the quantitative analysis among all the methods as well. Here, to calculate cost defined in (7) and waste defined in (8), we take the price of the AWS as reference. According to the data and description from reference [20], the one year of all upfront price is 541 dollars and can save 43% compared with paying on demand. Here, we just do a simple normalization and define the reservation unit price as 1.48 dollars and the unit price for real-time request as 2.60 dollars, respectively. The concrete details for the cost are summarized in Tab.5. Also, we add the best case and the worst case as the benchmark values, which indicates all the volumes are purchased by reservation and real-time requests, respectively. Obviously, we can find that although there is gap between the cost of our proposed meta-learning method and the counterpart of the best case, the proposed method always gives the better price than all the other methods, which helps save up to 39.93%, 37.15%, 5.62%, and 70.47% for the multi-intersections, roundabout, highway, and bridge scenarios, respectively, compared with the fully real time requests. Apart from the cost, we also calculate the amount of waste. Actually, the total cost is made up by two parts: one is the reservation part and the other one is the real-time request when we have insufficient reservation. Intuitively, both of them can be

Table 5: Total cost for different methods

Roadmap	Multi-intersections	Roundabout	Highway	Bridge
Best Case	79,663.366	19,249.67	62,557.02	102,169.57
Meta Learning	84,068.38	21,254.02	65,474.81	105,290.85
LSTM	84,315.18	21,402.28	66,557.07	105,755.63
BiLSTM	87,637.47	23,592.48	66,347.30	106,897.90
S-LSTM	87,264.55	23,359.78	70,263.50	106,796.78
S-BiLSTM	86,383.19	23,824.74	67,596.26	107,266.89
S-LSTM-D	88,215.74	22,885.18	69,513.89	108,729.89
S-BiLSTM-D	85,955.44	22,897.63	67,618.88	108,746.32
Worst Case	139,949.49	33,816.76	69,367.75	179,487.10

Table 6: Waste for different methods

Roadmap	Multi-Intersections		Roundabout	
Model	Excess	Inadequate	Excess	Inadequate
Meta-learning	**2,897.63**	**1,507.09**	**1,346.69**	**657.67**
LSTM	2,952.71	1,698.81	1,409.11	743.5
BiLSTM	4,897.08	3,076.73	3,587.86	754.95
S-LSTM	4,485.60	3,115.29	3,398.36	711.75
S-BiLSTM	4,328.86	2,390.67	3,940.01	835.06
S-LSTM-D	5,702.95	2,849.13	2,957.22	691.74
S-BiLSTM-D	4,123.82	2,167.96	2,856.19	791.77
Roadmap	Highway		Bridge	
Model	Excess	Inadequate	Excess	Inadequate
Meta-Learning	**1,329.84**	**1,587.95**	**1,944.85**	**1,176.43**
LSTM	1,407.77	2,592.28	2,189.71	1,396.35
BiLSTM	1,565.95	2,224.33	2,955.62	1,772.71
S-LSTM	5,184.06	2,522.42	2,934.56	1,692.65
S-BiLSTM	2,251.39	2,787.85	3,163.27	1,934.05
S-LSTM-D	4,331.12	2,625.75	4,217.52	2,342.80
S-BiLSTM-D	3,048.65	2,013.21	3,947.43	2,629.32

the source of waste. If the amount of reservation is more than that of needed, it means we make excess reservation, which leads to a waste of money for the customer and a waste of computation or storage resources for edge platform as well. Similarly, if the amount of reservation is less than the demands, it indicates we make inadequate reservation, which means purchasing more resources is inevitable for meeting the demands with a relatively expensive price. Therefore, when we calculate the waste, the distinguishment for these two kinds of waste is also took into consideration, and the details are summarized in Tab.6. For the total waste, the results keep consistent with the results of cost, i.e., the method gives the most economical scheme brings the least waste. Meanwhile, no matter for the excess waste or the inadequate waste, we can find that our proposed

meta-learning gives out the least waste.

2.6 Conclusion

In order to minimize the expenses to consume edge computing resources, this chapter proposed a two-stage meta-learning approach. In the first stage, a DNN is utilized to learn the experience dataset so as to figure out which machine learning model performs better in a specific situation. In the second stage, the resource amount anticipation will be conducted by the machine learning model selected by the DNN obtained in the first stage according to the meta-features. In addition, due to the fact that there is no open edge computing based vehicular network dataset, we program in the game engine Unity3D to build the 3D model of Manhattan area as in real world. Meanwhile, we adjust the factors like different roadmaps, the number of vehicles, and the randomness of task sizes for the traffic, which makes our data get closer to the practice. Eventually, we find that out proposed meta-learning method always gives the most economical predictions, which helps save up to 39.93%, 37.15%, 5.62%, and 70.47% for multi-intersections, roundabout, highway, and bridge scenarios, respectively, compared with the fully real time requests.

3 Love of Variety based Latency Analysis for High Definition Map Updating: Age of Information and Distributional Robust Perspectives

3.1 Introduction

Having stepped into the era of information technology, there are enormous artificial intelligence based autonomous devices, technologies, and services coming into being, and one important branch of which is autonomous vehicles or intelligent vehicles. According to the definition of the National Highway Traffic Safety Administration (NHTSA), the levels of vehicle automation can be categorized into six classes, which are distinguished by the extent of autonomy [15]. Currently, the performance of autonomous vehicles can just meet the requirements between levels 2 and 3, and both of which require the driver must be ready to take back control at any time. Aimed at achieving a higher automation level, one effective way is to utilize the high definition (HD) map. Unlike the traditional map, HD map is represented with a high degree of precision and resolution, which is as fine as 10-20 centimeters or better.

To generate and maintain HD map, one key is to extract useful information from data captured by embedded sensor systems, such as object detection, lane marking detection, ranging, etc. However, since different sensor systems have essential pros and cons regarding range, resolution, sensitivity to visibility, etc. [50]. In order to extract effective information that contributes to HD map generation and maintenance, collaborative making use of diverse sensors is inevitable, which exactly can be done by *federated analytics*. The aim of federated analytics is to obtain data insights among distributed devices by applying data science methods to the analysis of raw data generated on different clients [51]. Unlike federated learning, what transmitted between clients and aggregator are data insights rather than machine learning model gradients, such as distribution, positions of detected objects, etc. In other words, to support basic data science needs is the purpose of federated analytics.

In federated mechanisms, if the number of participants (i.e., sensors here) is larger, the desirable accuracy can be achieved faster and the optimization process converges

more quickly. From the perspective of practice, this can be interpreted as well. When extracting information that contributes to HD map, if more types of sensors are utilized, the obtained information, (e.g. positions of detected objects), will be more accurate, leading to a better accuracy level of HD map. For example, in a bad weather condition (e.g., fog, heavy rains, or storm), poor visibility will degrade the performance of visual-based sensors like LiDar and camera but has no influence on Radar and ultrasonic sensors [52]. In other words, the diversity of utilized sensors is higher, the more accurate HD map can be generated. However, the challenge is how to quantify the impact of diversity among the HD map accuracy level.

To tackle this challenge, we model the diversity of sensors in the federated analytics problem into the utility of *love of variety*. Basically, love of variety is a concept from economics, which assumes that each consumer has a demand for multiple varieties of a product over a given time period [53]. Within the time period, the utility will increase if consumers use more different products. For HD map generation, the interpretation is that, during the federated analytics, the more types of sensor data are utilized, the better accuracy of HD map can be achieved. However, since HD map is typically used for autonomous driving, the accuracy level has a required threshold, such that the probability of accidents can be reduced. Therefore, in order to achieve the desired accuracy level of HD map, if more types of sensors are included in federated analytics, the accuracy of each aggregation round will be higher. Thus, the total needed number of global aggregation rounds can be decreased.

Previously, the problem of HD map generation is discussed from one single vehicle's view. But from the practical use of HD map, it should be at least in a whole city wise. Therefore, uploading individual HD map to the server needs to be considered to form an intact city-wised HD map. Roughly, the information in the HD map can be categorized into dynamic layer and static layer. Dynamic layer information includes pedestrian, adjacent vehicles, and etc. While static layer information contains lanes, traffic signs, road markings, etc. Obviously, dynamic information updating is more latency sensitive since it may vary within seconds [54, 55]. Fortunately, emerging multi-access edge computing (MEC) provides a low-latency paradigm for data transmission, which is realized by edge

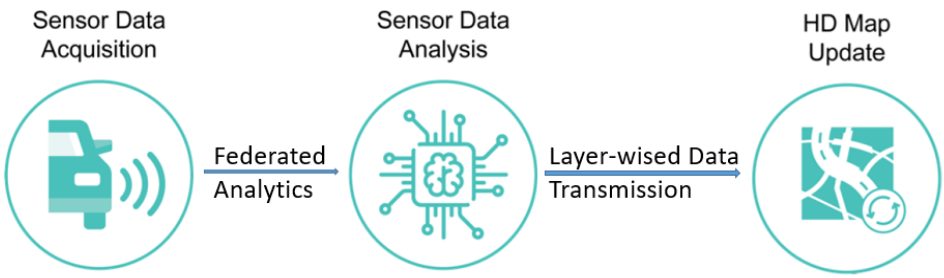


Figure 13: System Overview.

servers deployed at the edge of networks and the transmission distance can be reduced compared with traditional cloud computing [56, 57]. Because the different layers of HD map have different requirements for updating latency. In order to quantify the latency and measure the freshness of HD map, the *age of information* (AoI) is employed here. AoI is an end-to-end metric that can be used to characterize latency in status updating systems and applications. In AoI, information freshness and staleness can be quantified through the penalty function. For latency sensitive contents, the penalty will increase exponentially when the information staleness becomes larger. Therefore, considering the capacity of the edge server is limited, we propose a method to allocate the edge server capacity into different HD map layers so that the overall penalty can be minimized. Firstly, we discuss the case where the edge server capacity is deterministic. The optimal allocation scheme can be derived through Karush–Kuhn–Tucker (KKT) conditions. Then, consider the practice that an edge server provides services to multiple attached devices simultaneously, such that the available capacity for the autonomous vehicle is variational. We describe this ambiguity using Wasserstein metrics and reformulate the problem into a *distributional robust chance constrained optimization problem*, which can effectively and efficiently find a near-optimal solution. The overview of the system is illustrated in Fig. 13.

3.2 Related Work

As a key technique to realize autonomous driving, HD map has attracted interest from both industry and academia. Some existing literature focuses on how to make use of HD map. [58] uses near-term future information of HD map to propose a control scheme enabling predictive cruise control such that the overall fuel consumption can

be minimized. [59] proposes an image region of interest extraction method to improve the accuracy of traffic light recognition with the help of HD map and self-localization techniques. [60] utilizes HD map as prior information, and proposes a local motion planning method to realize the path planning and obstacles avoidance in autonomous driving scenarios. [61] proposes a sequential algorithm that can perform accurate lane-keeping or changing decisions while keeping a safe distance with the adjacent vehicle through using the position of the host vehicle and HD map. However, these works concentrates on HD map enabled applications instead of the generation and updating strategies.

While some of the existing literature focus stand on the perspective of HD map transmission. [62] proposes a collaborative vehicle to everything (V2X) transmission scheme to meet the transmission rate requirement for HD map while achieving low power consumption. [63] proposes a distributed multi-agent multi-armed bandit algorithm to maximize the accumulated cache utility for each roadside unit (RSU) by caching appropriate HD map contents in storage. [64] proposes a cluster based strategy for HD map offloading by using characteristics of HD map data and mobility of vehicles so that the energy consumption and offloading delay can be minimized. [65] proposes a fusing algorithm based on the Kalman filter to increase the position and semantic confidence of HD map such that the efficiency of HD map updating can be improved. [66] proposes a HD map data distribution mechanism such that the HD map provision task can be allocated to the selected RSU and transmit proportionate HD map data for energy efficiency purposes. But these works lack the analysis of sensor based HD map generation.

Also, there is some work focusing on the creation of HD map. [67] introduces the workflow of HD map creation and the machine learning based techniques used by industry that can minimize the amount of manual work for HD map generation. [68] proposes a crowd-sourcing framework to update the point cloud map layer in HD map from environment changes by jointly utilizing LiDar and vehicle communication. [69] proposes a semantic-based road segmentation method to address the problems of dynamic obstacles and shadows as well as the GNSS signal errors for HD map construction. [70]

proposes a Dislocation-Projection approach to create a color-pointed layer of HD map based on effective processing of LiDar, camera, and global navigation satellite system (GNSS) data. However, all the above literature only focuses on one perspective of HD map, either creation, transmission, or application. While our work jointly considers the generation and transmission processes of HD map in a layer-wised and latency-aware manner. Besides, the generation of HD map needs the contributions from different types of sensors, which is regarded as a federated analytics problem and almost no existing literature does this way.

3.3 Preliminaries

In this section, the background information is introduced. The basics and layered HD map are introduced in Subsection 3.3.1, the sensor systems are discussed in Subsection 3.3.2.

3.3.1 High Definition Map

HD map is made up of various information and resources, such as drivable paths, lane marks, the priority of lanes, traffic light and crosswalk to lane association, adjacent objects, and street furniture, which is represented in a high degree of resolution and precision, generally in the centimeter level. For practical autonomous driving use cases, HD map is the indispensable key for the advanced driver assistance system (ADAS). Intuitively, the contents of HD map can be roughly categorized into two classes: dynamic objects (such as pedestrians and vehicles) and static objects (such as traffic signs and lights). According to the definition of automotive edge computing consortium (AECC), the composition of HD map is layered, which can be represented by the highly dynamic layer, transient dynamic layer, transient static layer, and permanent static layer, as is illustrated in the upper part of Fig. 14 [71].

- In the highly dynamic layer, contents include the position, velocity, and accelerator of pedestrians, vehicles, and so on, which changes in several seconds.
- In the transient dynamic layer, contents include obstacles like fallen objects and trash, and local weather like unexpected sudden rain, which changes in several

minutes.

- In the transient static layer, contents include road work, temporary road closure, accidents, and so on, which changes in several hours.

- In the highly static layer, contents include lanes, traffic lights, road signs, and so on, which change in several days or longer.

Intuitively, in the current map, it is common to find the gaps between actual circumstances and the map, which takes days to be corrected. However, for autonomous driving, any delayed update for HD map can result in dangerous or even fatal accidents without human intervention. Therefore, HD map must be updated in a timely manner.

3.3.2 Sensor Systems in Autonomous Vehicles

HD map is inevitable for autonomous driving. How to generate and maintain this map with abundant dynamic and static information is challenging. Basically, HD map is mainly created by onboard sensor data. Typically, there are three main sensors in automotive vehicles in the present: cameras, Radar (Radio Detection And Ranging), and LiDar (Light Detection And Ranging). They allow the vehicle to see and sense everything on the road, as well as to collect the information needed in order to drive safely.

- *Camera*: Autonomous cars often have several smart cameras deployed in front and rear of the vehicle so that a 360° view of the external environment can be generated [72]. Unfortunately, these camera sensors are still far from perfect. Poor weather conditions such as rain, fog, or snow can prevent cameras from clearly seeing the obstacles in the roadway.

- *Radar*: Radar sensors send out radio waves that detect objects and gauge their distance and speed in relation to the vehicle in real-time. Unlike camera sensors, radar systems typically have no trouble at all when identifying objects during fog or rain. But since the sensors only scan horizontally, which can cause a variety of problems when driving under bridges or canyons [73].

Table 7: Pros and Cons of Sensor Systems in Autonomous Vehicles

Sensor	Strength	Weakness
Radar	Good ranging accuracy and does not rely on visibility	Cannot detect road markings
LiDar	Highly accurate ranging	Higher cost and less effective in featureless areas (like country roads)
Camera	Good object detection	Less effective in featureless roads, low visibility scenario (snow, rain, darkness. Good in tunnels/ urban canyons fog), roads without lane markings, and construction areas
GNSS	Good accuracy, Common global reference between vehicles,	Good in low visibility, featureless roads. Less reliable in high urban canyons
IMU	Good in low visibility, featureless roads	Not usable in long tunnels (a few km) due to high drift rate of IMU
Ultrasonic	Good in low visibility, featureless roads	Need close proximity and slow speeds

- *LiDar*: LiDar sensors work similar to radar systems, with the only difference being that they use lasers instead of radio waves. Apart from measuring the distances to various objects on the road, LiDar allows creating 3D images of the detected objects and mapping the surroundings [74]. The main problem for performance is the same as the camera, i.e., poor weather and visibility can sometimes block LiDar sensors.

Except for the three kinds of sensors, there are also some traditional sensor systems, such as

- *GNSS*: GNSS is the technology that uses satellites constellation to provide autonomous geo-spatial positioning, navigation, and timing services on a global or regional basis, which includes the GPS, GLONASS, Galileo, Beidou, and other regional systems.

- *Inertial motion units (IMU)*: IMU is a device including multi-axea, accelerometers, and gyroscopes that can provide an estimation of an object movement in space, such as measuring force, angular rate, altitude, and orientation.

- *Ultrasonic sensors*: Ultrasonic sensors mimic echolocation used by bats and are able to calculate distance between objects within a short range through transmitting high-frequency sound waves.

The pros and cons of different sensor systems are summarized in Table. 7.

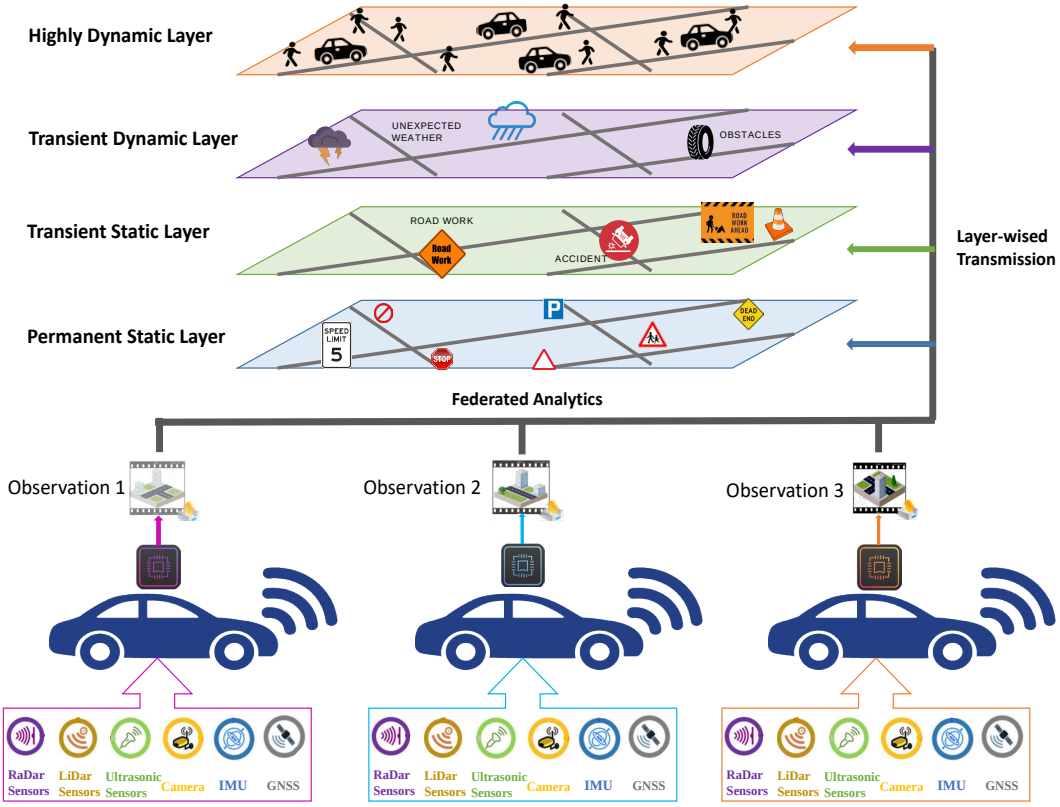


Figure 14: Illustration of federated analytics for HD map generation.

3.4 System Model and Problem Formulation

In this section, the concept of love of variety is introduced in Subsection 3.4.1. We model the utilization of multiple types of sensors for federated analytics into a love of variety based utility in Subsection 3.4.2. Then, the models of HD map layers and layer-wised transmission are introduced in Subsections 3.4.3 and 3.4.4, respectively. The transmission time based AoI is introduced in Subsection 3.4.5. Finally, since different layers of HD map have different delay requirements, we formulate the HD map transmission penalty minimization problem from the perspective of AoI in Subsection 3.4.6.

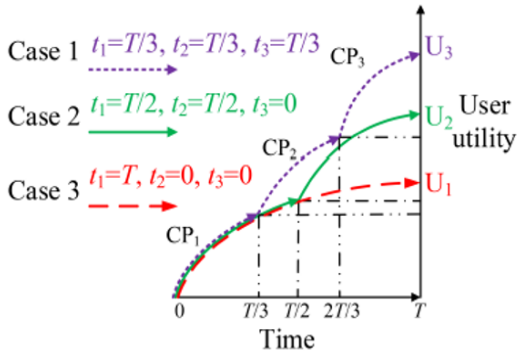


Figure 15: Illustration of RLV.

3.4.1 Love of Variety

Obviously, when performing federated analytics, as illustrated in Fig. 14, the results will be better if more types of sensors are included, e.g., the object detection and ranging can be accurate in both good visibility and poor weather scenarios. We assume the electronic control unit (ECU) can perform one type of data in a specific time slot, and a vehicle has a demand for multiple varieties of sensor data over time. Therefore, a time vector $\boldsymbol{t} = t_{i\in\mathcal{N}}$ can be used to describe the computation over different sensor systems, where t_i is denoted as ECU is executing a specific kind of sensor data during a fixed time slot. Apart from the utilization of diverse sensors, the utility of a vehicle is also related to computation time slot t_i. Therefore, the utility function of ECU can be defined as $u(t_i)$, which is a strictly increasing and concave function and meets $u(0) = 0$, as is suggested in [75]. The general utility function can be defined as

$$u(t_i) = \frac{1}{1-\rho}\left[(\alpha + t_i)^{1-\rho} - \alpha^{1-\rho}\right] + \beta t_i, \tag{29}$$

where $\alpha \geq 0$, $\beta \geq 0$, and $0 < \rho < 1$ are constant coefficients. By assigning different values for α, β, and ρ, the utility function can illustrate different vehicle characteristics for variety. Intuitively, the utilities from computation over different types of sensors are additive. Overall, the aggregated utility of an ECU or vehicle is $\sum_{i\in\mathcal{N}} u(t_i)$.

Mathematically, to involve variety, one key challenge is how to evaluate or quantify the willingness of an ECU to hand over computation from sensor data x to another sensor data y. Besides, since each vehicle needs diverse sensor data to obtain more accurate and effective information for HD map, how to quantify willingness of exchanging

among multiple sensors is also challenging. In order to solve this problem, we introduce elasticity, whose definition is shown below.

Definition 1 *For two variables x and y, the x-elasticity of y is defined as*

$$\epsilon_x^y = -\frac{\partial y}{\partial x}\frac{x}{y}. \tag{30}$$

The interpretation of elasticity is that the percentage change in y is in response to the percentage change in x. If the value of elasticity is larger, it means y is more sensitive to the change of x. To quantify the willingness of exchanging among multiple sensors, the definition of relative love of variety (RLV) is given as the following.

Definition 2 *The vehicle's relative love of variety is the elasticity of the marginal utility with respect to the computation time slot t_i, which is described by*

$$r_u(t_i) = \epsilon_{t_i}^{u'} = -\frac{u^{''}t_i}{u'} > 0. \tag{31}$$

Obviously, from Definition 2, the value of RLV reflects whether the vehicle is willing to exchange different sensor data in consecutive time slots for achieving a higher marginal utility, as is described in Fig. 15. For case 1, the user changes consumption among sensor data at the end of time slot $T/3$ and achieves the highest utility U_3. For case 2, the user consumes two types of sensor data within time T and achieves utility U_2. While the user in case 1 keeps the same type of sensor data throughout time interval and achieves the lowest utility U_1. Besides, it should be noted that the definition of RLV is with respect to a particular sensor generated data. When we need to analyze the overall RLV difference among multiple vehicles, we can utilize the average or summation of RLVs from all the types of sensor data to describe the overall RLV level, i.e., $\bar{U} = \frac{1}{N}\sum_{i=1}^{N} u(t_i)$.

3.4.2 Data Quality for Federated Analytics

Obviously, when performing federated analytics, the results will be better if more vehicles with high love of variety unity are included [76]. Therefore, we can utilize the utility value or average RLV to describe the precision or accuracy. According to

[77], for distributed optimization, if the global optimization problem is strongly convex, the general upper bound on number of global iteration is $\frac{\mathcal{O}(log(\frac{1}{\psi}))}{1-\Psi}$, where Ψ is relative accuracy level of the local subproblem. Then, the needed number of global iterations can be given as

$$G = \frac{\zeta \cdot \log(\frac{1}{\psi})}{1-\Psi}, \tag{32}$$

where $\zeta > 0$ is a constant. It can be seen that for a certain number of global iterations, the global accuracy level can be improved (i.e., ψ is close to 0) solving local subproblems towards high accuracy. However, the inverse dependence on $1-\Psi$ means that there is a limit to how much the global accuracy can gain from the high-accurate solutions of the local subproblems [78]. To obtain ψ-accuracy, the global iteration will always require $\zeta \cdot \log(\frac{1}{\psi})$ [79].

For federated mechanisms, if the number of participants is larger, higher accuracy can be achieved when the number of global aggregation is fixed. Likewise, here, when the number of types of the sensor is larger, the value of RLV or the utility U will be larger accordingly. When a desirable accuracy level ψ is given, the needed number of global aggregation will be less compared with the low average variety vehicle group. Therefore, for the vehicles with average utility level $\bar{U}$, the needed number of global federated analytics aggregation will be

$$L = \kappa G \left\lceil \frac{\log(\frac{1}{\psi})}{\bar{U}} \right\rceil, \tag{33}$$

where κ is a constant such that the given accuracy level ψ can be achieved. Obviously, if the average utility is larger, the total number of iteration will be less when the global accuracy level is fixed, as is proved in [76].

3.4.3 HD Map Components

Let $L_1(r_1)$, $L_2(r_2)$, $L_3(r_3)$, and $L_4(r_4)$ denote highly dynamic layer, transient dynamic layer, transient layer, and permanent static layer, respectively. r_i is the range that needs to be updated. Here, r_i for each layer is different, which means each HD map layer has an essential requirement for updating range. For example, for the permanent

static layer, the required info may be "on street A (like several miles long or more), the speed limit is 50mph". However, for the highly dynamic layer, the vehicle needs to know the nearest pedestrian within several meters or less, which is much more precise than the permanent static layer requirement. Therefore, the precision scale from the permanent static layer, transient static layer, transient dynamic layer, to the highly dynamic layer can be from county wide, region wide, a few blocks wide, to a crossing wide. Therefore, for each time updating, i.e., performing a federated analytics global aggregation, the size of each layer can be represented by

$$D_i = k \times r_i, \tag{34}$$

where k is a constant.

Moreover, in order to achieve ψ-accuracy, the total number of the HD map size will be

$$D = L \sum_{i=1}^{4} D_i = L \sum_{i=1}^{4} k r_i. \tag{35}$$

Here, the number of components is four, which is because the number of layers to compose HD map is four, as discussed in Subsection 3.3.1.

3.4.4 Communication Model

Intuitively, since we have different requirements (scale and time) for different layers. For those with no reporting of the real-time object layers, e.g. the transient static layer and permanent static layer, usually with relatively larger required areas, the updating can be done via the cloud through the backhaul network. While for those reporting of real-time objects layers, e.g. the transient dynamic layer and highly dynamic layer, usually with relatively smaller required regions, the HD map updating can be done through V2V communication or edge servers. For the updating part that is transmitted through the edge server, the transmission rate can be described as

$$v = B \log_2 \left(1 + \frac{ph}{N_0}\right), \tag{36}$$

where B is the bandwidth, p is transmission power, h is channel gain, and N_0 is the Gaussian noise. For the updating that is done by the backhaul network, we assume the delay to transmit a unit size map is a constant t_c, which is typically larger than the time of transmission through edge.

To update a certain layer, we assume the part transmitted via edge server is denoted by $\phi_i \in [0, 1]$. So in a time period T, the total time consumption is

$$t(\phi_i) = \sum_{i=1}^{4} \left(\frac{D_i \phi_i}{v} + D_i(1 - \phi_i) t_c \right). \tag{37}$$

3.4.5 Age of Information

AoI is an end-to-end metric that can be used to characterize latency in status updating systems and applications [80]. An update packet with timestamp a is said to have age $b - a$ at a time $b \geqslant a$. Here, timestamp a denotes the generation time of HD map and timestamp b denotes the reception of updating [81]. Therefore, the difference $b-a$ describes the transmission time of HD map, i.e., $b-a = t(\phi_i)$. For each composition in HD map, the updating age Δ_i can be written as

$$\Delta_i = t(\phi_i). \tag{38}$$

Since the delay requirement for HD map updating is very strict, when the delay becomes larger in a certain time period, the penalty increases non-linearly at the same time. Therefore, the staleness of information updating can be defined as

$$p_i \left(t_i(\phi_i) \right) = t_i(\phi_i)^{\alpha_i}, \tag{39}$$

where α_i is the constant and can describe sensitivity of different HD map layers regarding to delay. The typical cure of staleness is shown in Fig. 24.

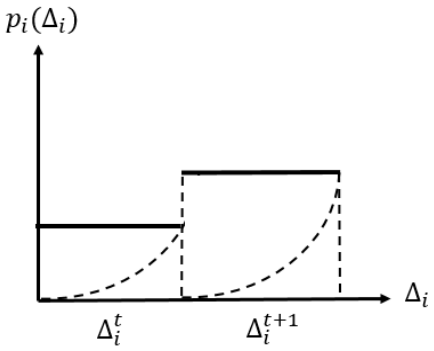


Figure 16: Penalty function.

3.4.6 Problem Formulation

For an autonomous vehicle, in order to minimize the time staleness for HD map updating, the problem can be formulated as

$$\begin{aligned} \min_{\phi_i} \quad & \sum_{j=1}^{4} p_i \\ s.t. \quad & \sum_{i=1}^{4} D_i \phi_i \leq q, \\ & \phi_i \geqslant 0, \end{aligned} \tag{40}$$

where q denotes the available computation or storage resources of edge server. Here, the constrain means the total edge transmission data of four layers cannot exceed the capacity of the edge server. However, practically, an edge server provides services to multiple attached devices simultaneously, such that the available capacity q for the autonomous vehicle is variational. Therefore, problem (12) can be rewritten as

$$\begin{aligned} \min_{\phi_i} \quad & \sum_{j=1}^{4} p_i \\ s.t. \quad & \sum_{i=1}^{4} D_i \phi_i \leq \Delta q, \\ & \phi_i \geqslant 0, \end{aligned} \tag{41}$$

where Δq denotes the uncertainty of available capacity.

3.5 Deterministic Capacity Case

In problem (13), the staleness of each layer is p_i, which can be rewritten as

$$\begin{aligned} p_i &= \left[\frac{D_i\phi_i}{v} + D_i(1-\phi_i)t_c\right]^{\alpha_i} \\ &= \left[\left(\frac{1}{v} - t_c\right) D_i\phi_i + D_i t_c\right]^{\alpha_i} \\ &= (a_i x_i + b_i)^{\alpha_i}, \end{aligned} \tag{42}$$

where $x_i = D_i\phi_i$, $a_i = \left(\frac{D_i}{v} - D_i t_c\right)$ and $b_i = D_i t_c$ are layer characteristic constants. Therefore, problem (12) can be rewritten as

$$\begin{aligned} \min_{x_i} \quad & \sum_{j=1}^{4} (a_i x_i + b_i)^{\alpha_i} \\ s.t. \quad & \sum_{i=1}^{4} x_i \leq q, \\ & x_i \geqslant 0. \end{aligned} \tag{43}$$

To address problem (15), we have the following theorem.

Theorem 1 *Let x^* be a feasible point of*

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & g_i(x) \leq 0, \quad i = 1, \cdots, m, \\ & h_i(x) = 0, \quad i = 1, \cdots, n, \end{aligned} \tag{44}$$

where f and g_i are continuously differentiable convex functions over $\mathbb{R}$, and h_i are affine functions. Suppose that there exsist multipliers $\lambda_1, \cdots, \lambda_m$ and $\mu_1, \cdots, \mu_n \in \mathbb{R}$ such that

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) + \sum_{i=1}^{n} \mu_i \nabla h_i(x^*) = 0 \tag{45}$$

and

$$\lambda_i g_i(x^*) = 0, \quad i = 1, \cdots, m. \tag{46}$$

Then x^* is an optimal solution of (16).

proof 1 *Let x be a feasible point. Define the convex function*

$$L(x) = f(x) + \sum_{i}^{m} \lambda_i g_i(x) + \sum_{i=1}^{n} \mu_i h_i(x). \tag{47}$$

A feasible point x^* is a minimizer of L, since $\nabla L(x^*) = 0$ and in particular $L(x^*) < L(x)$, we have

$$\begin{aligned} f(x^*) &= f(x^*) + \sum_{i}^{m} \lambda_i g_i(x^*) + \sum_{i=1}^{n} \mu_i h_i(x^*) = L(x^*) \\ &\leq L(x) = f(x) + \sum_{i}^{m} \lambda_i g_i(x) + \sum_{i=1}^{n} \mu_i h_i(x) \\ &\leq f(x), \end{aligned} \tag{48}$$

showing that x^* is the optimal solution. ■

To find the solution to problem (14), we introduce Lagrange multiplier $\lambda^* \in \mathbb{R}$ for the inequality constraint $x_i \geqslant 0$, and a multiplier $\mu^* \in \mathbb{R}$ for the equality constraint $\sum x_i = q$, i.e.,

$$L = \sum (a_i x_i + b_i)^{\alpha_i} - \sum \lambda_i x_i + \mu \left(\sum x_i - q \right). \tag{49}$$

Then, we can obtain the KKT conditions. Firstly, we have the following lemma.

Lemma 1 *Let x^* and any (λ^*, μ^*) be any primal and dual optimal points with zero duality gap. Since x^* minimizes $L(x, \lambda^*, \mu^*)$ over x, it follows that its gradient must vanish at x^*.*

Therefore, we have the stationarity condition, which is

$$a_i \alpha_i (a_i x_i^* + b_i)^{\alpha_i - 1} - \lambda_i^* + \mu^* = 0. \tag{50}$$

The complementary slackness is

$$\lambda_i^* x_i^* = 0. \tag{51}$$

Primal feasibility is

$$x_i^* \geqslant 0 \tag{52}$$

and

$$\sum x_i^* = q. \tag{53}$$

The dual feasibility is

$$\lambda^* \geqslant 0. \tag{54}$$

We can directly solve these equations to find x_i^*, λ^*, and μ^*. We start by noting that λ^* acts as a slack variable in (17), so it can be eliminated. So we have

$$\left(\mu^* + a_i\alpha_i(a_ix_i^* + b_i)^{\alpha_i - 1}\right) x_i^* = 0 \tag{55}$$

and

$$\mu^* \geqslant -a_i\alpha_i(a_ix_i^* + b_i)^{\alpha_i - 1}. \tag{56}$$

If $\mu^* < -a_i\alpha_i b_i^{\alpha_i - 1}$, condition (22) can only hold if $x_i^* > 0$. Solving for x_i^*, we conclude that $x_i^* = \frac{\sqrt[\alpha_i - 1]{\frac{\mu^*}{a_i\alpha_i} - b_i}}{a_i}$. If $\mu^* > -a_i\alpha_i b_i^{\alpha_i - 1}$, then $x_i^* > 0$ is impossible, because it would imply $\mu^* \geqslant -a_i\alpha_i b_i^{\alpha_i - 1} > -a_i\alpha_i(a_ix_i^* + b_i)^{\alpha_i - 1}$, which violates the complementary slackness condition. Therefore, $x_i^* = 0$ if $\mu^* \geqslant -a_i\alpha_i b_i^{\alpha_i - 1}$. Thus we have

$$x_i^* = \begin{cases} \dfrac{\sqrt[\alpha_i - 1]{\frac{\mu^*}{a_i\alpha_i} - b_i}}{a_i} & \mu^* < -a_i\alpha_i b_i^{\alpha_i - 1}, \\ 0 & \mu^* \geqslant -a_i\alpha_i b_i^{\alpha_i - 1}. \end{cases} \tag{57}$$

Alternatively, we can put in a more simple way,

$$x_i^* = \max\left\{0, \frac{\sqrt[\alpha_i - 1]{\frac{\mu^*}{a_i\alpha_i} - b_i}}{a_i}\right\}. \tag{58}$$

Substituting the expression for x_i^* into the equality constraint, we have

$$\sum_{i=1}^{4} \max\left\{0, \frac{\sqrt[\alpha_i - 1]{\frac{\mu^*}{a_i\alpha_i} - b_i}}{a_i}\right\} = q. \tag{59}$$

3.6 Uncertain capacity case

The previous section discusses the solution to the deterministic edge capacity case. However, practically, an edge server provides services to multiple attached devices simultaneously, such that the available capacity q for the autonomous vehicle is variational. In this section, the uncertain capacity allocation case is discussed. Subsection 3.6.1 introduces the method using the Wasserstein distance to mathematically describe the ambiguity. Then, a distributionally robust chance constrained optimization based method is introduced to solve the uncertain capacity allocation problem in Subsection 3.6.2.

3.6.1 Wasserstein Distance Based Ambiguity Set

Assuming ambiguity set Δq follows a distribution, a natural way to hedge against the distributional ambiguity is to consider a neighborhood of the empirical probability distribution. Considering the discrepancy-based ambiguity sets: we introduce ambiguity sets based on probability distance

$$\mathcal{P} = \{P : d(\widehat{P_N}, P) \leq \epsilon\}, \tag{60}$$

where $\hat{P_N}$ is empirical probability for capacity q distribution, ϵ is radius. $d(\hat{P_N}, P)$ is the metric to measure the similarity or distance of two distributions, and one of the most commonly used methods is the Wasserstein distance, which is defined as follows.

Definition 3 *The Wasserstein metric* d_W*:* $\mathcal{M}(\Xi) \times \mathcal{M}(\Xi) \to \mathbb{R}$ *is defined by*

$$d_W(\mathbb{Q}_1, \mathbb{Q}_2) := \inf \int_{\Xi^2} \|\xi_1 - \xi_2\| \Pi(d\xi_1, d\xi_2), \tag{61}$$

for all distributions $\mathbb{Q}_1, \mathbb{Q}_2 \in \mathcal{M}(\Xi)$*, where* $\|\cdot\|$ *represents an arbitrary norm on* $\mathbb{R}^m$*,* $\Pi(\mathbb{Q}_1, \mathbb{Q}_2)$ *is the the set of all possible joint distributions of* $\mathbb{Q}_1$ *and* $\mathbb{Q}_2$*.* ξ_1 *and* ξ_2 *are the samples under joint distribution* Π*.*

With the Wasserstein metric, the Wasserstein distance based ambiguity set can be

defined as

$$\mathbb{B}_\epsilon(\widehat{P_N}) = \left\{ \mathbb{Q} \in \mathcal{M}(\Xi) : d_W(\widehat{P_N}, \mathbb{Q}) < \epsilon \right\}. \tag{62}$$

The ambiguity set Q can be viewed as a Wasserstein ball which contains all probability distributions whose Wasserstein distance to the empirical distribution $\widehat{P_N}$ is less than ϵ [82]. The Wasserstein ball contains all possible probability distributions. $\mathbb{Q}$ will include the true distribution with a higher probability. Under a common light tail assumption on the unknown data-generating distribution, this ambiguity set offers attractive performance.

3.6.2 Distributional Robust Chance Constrained Optimization

With the Wasserstein metric, problem (13) can be reformulated as the following Distributional Robust Chance Constrained Optimization (DRCCO) form:

$$\begin{aligned} \min_{x} \quad & cx \\ s.t. \quad & x_i \in S, \\ & \inf_{P \in \mathcal{P}} P\left\{ \tilde{\xi} : \tilde{\xi} \leq b(x) \right\} \geqslant 1 - \epsilon, \end{aligned} \tag{63}$$

where vector $x \in \mathbb{R}^n$ denotes the decision variables, vector $x \in \mathbb{R}^n$ is the objective function coefficients, set $S \subseteq \mathbb{S}^n$ is the deterministic constraints on x, and the last constraint is a chance constraint specified by the ambiguity set. Therefore, the second constraint is also named the distributionally robust chance constraint (DRCC). DRCC requires all of the uncertain constraints satisfied for all the probability distributions from ambiguity set P with a probability at least $(1 - \epsilon)$, where $\epsilon \in (0, 1)$ is the specified risk tolerance [83]. The feasible region induced by DRCC is defined as

$$Z := \left\{ x \in \mathbb{R}^n : \inf_{P \in \mathcal{P}} P\{ \tilde{\xi} : \tilde{\xi} \leq b(x) \} \geqslant 1 - \epsilon \right\}. \tag{64}$$

Firstly, by using the strong duality result from [84], (36) can be reformulated by the following theorem.

Theorem 2 *Set Z is equivalent to*

$$Z = \left\{ x \in \mathbb{R}^n : \begin{array}{c} \delta - \epsilon\gamma \leq \frac{1}{N} \sum\limits_{j\in[N]} z_j, \\ z_j + \gamma \leq \max\left\{b(x) - \zeta^j, \forall j \in [N], \right\}, \\ z_j \leq 0, \forall j \subseteq [N], \gamma \geqslant 0. \end{array} \right\}. \tag{65}$$

Further, we can reformulate set Z in (37) as a mixed integer program as below [85].

Lemma 2 *For DRCCO with right-hand uncertainty, suppose that there exists an $M \in \mathbb{R}_+^N$ such that*

$$\max_{i\in[I]} \max_{x\in Z} \left\{|b(x) - \zeta^j|\right\} \leq M_j, \tag{66}$$

for $\forall j \in [M]$. Then set Z is mixed integer representable, i.e.,

$$Z = \left\{ x \in \mathbb{R}^n : \begin{array}{c} \delta - \epsilon\gamma \leq \frac{1}{N} \sum\limits_{j\in[N]} z_j, \\ z_j + \gamma \leq s_j, \forall j \in [N] \\ s_j \leq b(x) - \zeta^j + M_j(1 - y_j), \forall j \in [N], \\ s_j \leq M_j y_j, \forall j \in [N], \\ \gamma \geqslant 0, z_j \leq 0, s_j \geqslant 0, y_i \in \{0, 1\}, \forall j \in [N]. \end{array} \right\}. \tag{67}$$

However, the scale of programming (39) is large, which is difficult to meet the real-time requirements of HD map updating. Therefore, to quickly find a feasible solution needs to be considered. According to [86], the feasible set Z can be reformulated as descriptions by a conditional-value-at-risk (CVaR) constrained set. Regarding a random variable $\tilde{X}$, the $(1-\epsilon)$-value at risk (VaR) of $\tilde{X}$ is

$$\mathrm{VaR}_{1-\epsilon}(\tilde{X}) = \min\left\{z : F_{\tilde{X}} \geqslant 1 - \epsilon\right\}, \tag{68}$$

where F is the cumulative distribution function of $\tilde{X}$, defined by $F_{\tilde{X}} = P\{\tilde{X} \leq z\}$. And the $(1-\epsilon)$-CVaR is defined as

$$\mathrm{CVaR}_{1-\epsilon}(\tilde{X}) = \min_{\beta} \left\{ \beta + \frac{1}{\epsilon}\mathbb{E}_P \left[\tilde{X} - \beta\right]_+ \right\}. \tag{69}$$

It is observed that, for any random variable $\tilde{X}$, we have

$$\mathrm{CVaR}_{1-\epsilon}(\tilde{X}) \leq \mathrm{CVaR}_1(\tilde{X}) := \mathrm{ess.sup}(\tilde{X}). \tag{70}$$

Then the feasible solution can be approximated by the following theorem.

Theorem 3 *Set Z can be inner approximated by*

$$Z_R = \left\{ x \in \mathbb{R}^n : \frac{\delta}{\epsilon} + \zeta^j \leq b(x), \forall j \in [N] \right\}. \tag{71}$$

proof 2 *Because $CVaR_{1-\epsilon}[-f(x,\zeta)] \leq ess.sup[-f(x,\zeta)]$, where ζ is a random vector, the feasible set Z can be inner approximated as*

$$Z_R = \left\{ x \in \mathbb{R}^n : P_\zeta \left\{ f(x,\zeta) \geqslant \frac{\delta}{\epsilon} \right\} = 1 \right\}. \tag{72}$$

By using the definition of $f(x,\zeta)$, i.e.,

$$f(x,\zeta) = \min \left\{ \min_x \max\{b(x) - \zeta, 0\}, \min_x \chi_{\{x:b(x)<0\}}(x) \right\}, \tag{73}$$

where the characteristic function $\chi_R(x) = \infty$ if $x \neq R$ and 0, otherwise. Also, because of the fact that $\frac{\delta}{\epsilon} > 0$, we can arrive (44).■

In Theorem 3, it can be proved that set Z_R is a subset of he feasible region induced by a regular chance constraint, i.e., $Z_R \subseteq Z$ [86].

3.7 Simulation results

In this section, the simulation is performed for deterministic and uncertain capacity cases in Subsections 3.7.1 and 3.7.2, respectively. The experiments are conducted on MATLAB platform. The RLV based utility function is defined as $u(t_i) = 2\times(1+t_i)^{0.5}-1$, by setting $a = 0$, $b = 1$, and $\rho = 0.5$. The constant parameter of iteration upper bound of federated analytics ζ is 20. The global accuracy index for federated analytics is 0.1. The parameters that denote HD map data size k is set as 10. The updating range for four layers are set as 2, 3, 4, and 5 MB/m^2, respectively. As for the communication model,

Table 8: Parameters Settings

Parameters	**Value**
RLV Parameter a	1
RLV Parameter b	0
RLV Parameter ρ	0.5
Global accuracy index ψ	0.1
Global accuracy parameter ζ	10
Global iteration parameter κ	10
HD map parameter k	10
HD map size parameter r_1, r_2, r_3, and r_4	2, 3, 4, and 5 MB/m^2
Bandwidth B	$10MHz$
Transmission power	23 dBm
Channel gain h	16
Gaussian Noise N_0	-96 dBm
Penalty parameters for four layers α_1, α_2, α_3, and α_4	4, 3, 2, and 1

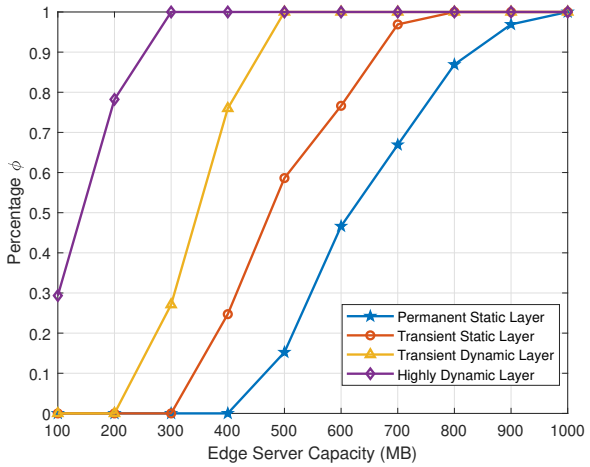


Figure 17: Edge server allocation percentage with different edge server capacity.

the bandwidth B is 10 MHz. The transmission power is 23 dBm. The channel gain is 16. The Gaussian noise power is -96 dBm. And for the penalty function parameters, α_1, α_2, α_3, and α_4 are set as 1, 2, 3, and 4, for the highly dynamic layer, transient dynamic layer, transient layer, and permanent static layer, respectively. Overall, the general model parameter settings are summarized in Table 11.

3.7.1 Deterministic Capacity Case

In this case, we firstly assume there are four vehicles in total, which will consume a random number of sensor data from the six types of sensors as introduced in Subsection 3.3.2. We increase the edge server capacity from 100 MB to 1,000 MB by step 100

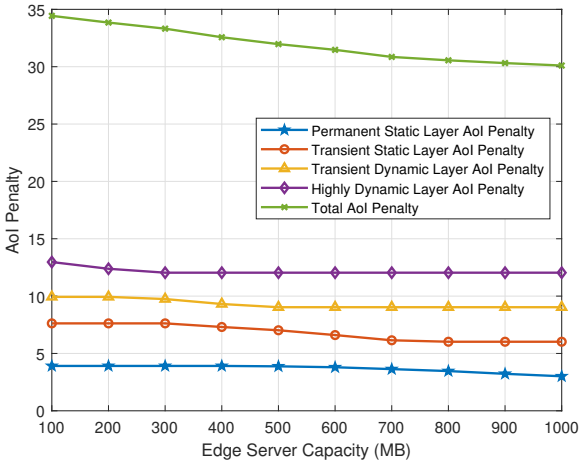


Figure 18: The AoI penalty with different edge server capacity.

MB and the results are shown in Figs. 28 and 18, respectively. Fig. 28 illustrated the variation of allocation percentage of each HD map layer to edge server. Obviously, the highly dynamic layer always has priority to consume the edge server resources and is the first one that achieves the 100% edge resources allocation at 300 MB capacity, followed by the transient dynamic layer at 500 MB, the transient static layer at 800 MB, and the permanent static layer at 1000 MB, respectively. This is understandable because each layer has different latency requirements. For example, the highly dynamic layer, which consists of surrounding vehicles and pedestrians, is much more latency sensitive than the permanent static layer which consists of road signs. Correspondingly, the penalty assigned to the highly dynamic layer will be more than the permanent static layer with regard to the same time delay. Typically, we set the penalty function to highly dynamic layer, transient dynamic layer, transient static layer, and permanent static layer as quadruplicate, cubic, quadratic, and linear functions, respectively, which gives rise to the allocation priority order as in Fig. 28. In the beginning, when the edge server resources are limited, only the highly dynamic layer is able to consume the resources because small latency will result in the fourth power penalty. Thereafter, when the resources are plentiful, all of the four HD map layers can be transmitted via the edge server.

This change can also be demonstrated by Fig. 18, which shows the overall penalty and penalty of each layer. Generally, all of the curves indicate decreasing trend with the

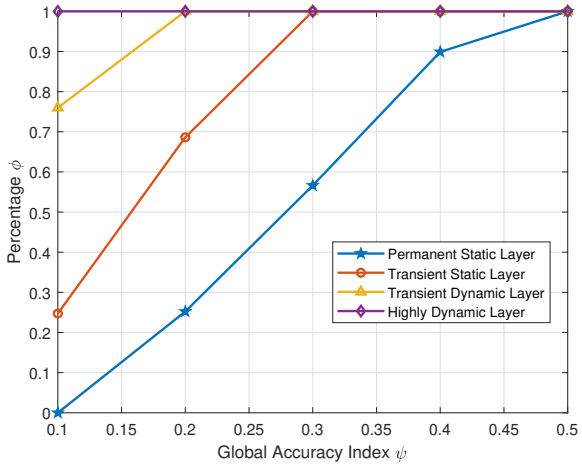


Figure 19: Edge server allocation percentage with different global accuracy index.

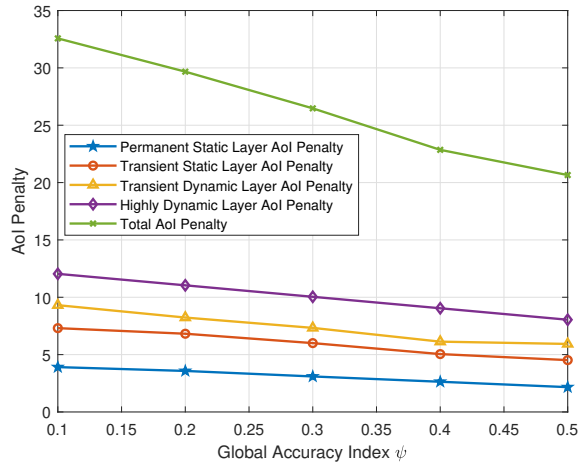


Figure 20: The AoI penalty with different global accuracy index.

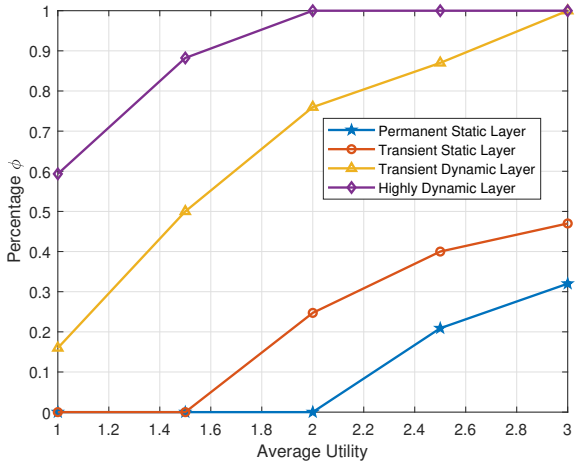


Figure 21: Edge server allocation percentage with different average utility $\bar{U}$.

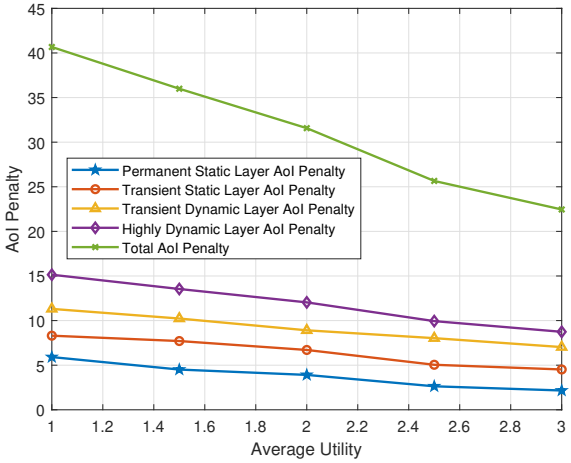


Figure 22: The AoI penalty with different average utility $\bar{U}$.

increase of edge server capacity. Because when the edge resources are more sufficient, more HD map data will be transmitted through the edge server such that the transmission delay can be reduced, resulting in the reduction of AoI penalty. In addition, we can see that the drop point of the highly dynamic layer AoI penalty is 100 MB edge capacity, 200 MB edge capacity for the transient dynamic layer, 300 MB edge capacity for the transient static layer, and 400 MB edge capacity for the permanent static layer, which also keeps consistent with the allocation priority order mentioned above.

Table 9: Results for Uncertain Capacity Case

Parameter		Optimal Solution		CVaR Model			Random		
ϵ	δ	Opt.Val	Time	Value	Gap	Time	Value	Gap	Time
0.05	0.01	32.57	4.73s	32.93	1.11%	0.05s	36.49	12.04%	0.02s
0.05	0.02	32.57	2.32s	33.02	1.38%	0.06s	33.69	3.44%	0.02s
0.1	0.01	32.57	9.16s	32.97	1.23%	0.06s	36.03	10.62%	0.02s
0.1	0.02	32.57	10.85s	33.68	3.17%	0.05s	34.87	7.06%	0.02s

Next, because the required HD map accuracy is crucial for federated analytics based HD map generation. We also explore the impact of the global accuracy index ψ on the performance. In this case, we keep the edge server capacity as 400 MB. The global accuracy index ψ increases from 0.1 to 0.5 by step 0.1, which represents local accuracy decreases gradually. The corresponding results are illustrated in Figs. 29 and 20. In Fig. 29, we can see that the overall trend is the percentage of each layer allocated to edge server transmission increases with the increase of the global accuracy index. Since a high global accuracy index indicates a low local accuracy. Therefore, to achieve to the

required precision, the needed number of global aggregation will be less due to a lower accuracy desire. Accordingly, the generated data size will be reduced, resulting in a larger percentage of HD map data allocated to the edge server with a certain volume of capacity. Hence, with only 400 MB available capacity, all of the four layers can realize fully edge server transmission with a lower accuracy expectation, i.e., ψ=0.5. In addition, similarly, because of the differences of the penalty functions, the allocation priority order still exists, where the highly dynamic layer has the highest priority, followed by transient dynamic layer, transient static layer, and permanent layers. As for the results in Fig. 20, obviously, the AoI penalty decreases with the increase of the global accuracy index for all cases. The reason is the same as explanation for Fig. 29. The number of global aggregation will be reduced due to a lower accuracy requirement. As a consequence, the generated data size will be smaller, resulting in a lower transmission latency and AoI penalty. Also, this result reveals that a trade-off can be made between accuracy and transmission delay so the decisions can be adjusted for cases with specific requirements.

Intuitively, HD map generation heavily relies on vehicle sensor data. Therefore, we also discuss the influence of different average utilities among HD map allocation and AoI penalty. In this part, we keep the available edge server capacity as 400 MB and the global accuracy index as 0.1. The average utility is increased from 1 to 3 by step 0.5. The results are shown in Figs. 21 and 22. In Fig. 10, we can see the allocation percentage increases with the increase of average utility. This is because when the average utility is larger, the needed number of federated analytics aggregation will be less regarding a given accuracy level. Thus, the data size that needs to be transmitted will be less as well, resulting in lower latency and AoI penalty. Likewise, in any case, the highly dynamic layer is always allocated the majority of edge resources, which brings into correspondence with previous experiments. Also, the penalty is illustrated in Fig. 22. The AoI penalty decreases with the increase of average utility $\bar{U}$. Since a higher average utility leads to a smaller data size, the reduction in transmission time will bring less AoI penalty.

3.7.2 Uncertain Capacity Case

In this subsection, we will use the CVaR based inner approximation to estimate the uncertain capacity case. The mean value of edge server capacity is set as 400 MB. The risk parameter ϵ is chosen from $\{0.05, 0.1\}$. And the Wasserstein distance δ is set as $\{0.1, 0.02\}$. The number of instances N is 10. In addition CVaR based inner approximation, we also use random strategy to act as a comparison method. Also, since we can refer to the optimal solution from the deterministic case, the optimality gap can be adopted as the evaluation metric, which is defined as

$$\text{GAP} = \frac{|\text{Value - Opt.Val}|}{\text{Opt.Val}}, \tag{74}$$

where Value indicates the inner approximated AoI penalty and the Opt.Val is the optimal AoI penalty based on the deterministic case. The results are shown in Table 9.

Generally, we can see that no matter for CVaR model based approximation or random strategy, they have a certain amount of accuracy gap regarding the optimal solution. The CVaR model is usually $1\% - 3\%$ away from the optimality. However, the optimality gap for random strategy varies a lot, where the least one is 3.44 % and the largest one is 12.04%, because the allocation percentage is randomly determined. Therefore, in terms of approximation accuracy, CVaR model is better. Since the existence of uncertainty, the optimality gap is inevitable but $1\% - 3\%$ difference is also acceptable. In addition, the results for CVaR model shows that the least gap is achieved when $\epsilon = 0.05$ and $\delta = 0.01$ while the largest gap comes from the case with $\epsilon = 0.1$ and $\delta = 0.02$. This is also understandable because the larger value indicates a more relaxed or larger ambiguity space. If the ϵ is smaller, the risk tolerance is smaller as well, so that high risk instances will be exempted. While for δ, if the number is larger, the radius of uncertain Wasserstein ball is larger, which increases ambiguity level.

Another thing that needs to be noted is the execution time. To obtain the optimal solution, the time consumption can be up to 10 seconds. However, the CVaR model execution time is typically less than one second. The reason behind this might be CVaR model is a second order conic programming and does not involve any binary variables.

Therefore, for time sensitive applications, the CVaR model based inner approximation can be an effective approach for the trade-off between time consumption and accuracy. Additionally, random strategy achieves the lowest execution time because it does not solve any complex programming but randomly chooses a percentage within the given range. Overall, the results in Table 9 demonstrate that the CVaR model based approximation is able to find near-optimal solutions with much less time consumption.

3.8 Conclusion

As a latency sensitive application, HD map transmission needs to be performed in a timely manner. This chapter investigates an information staleness minimization problem for federated analytics based HD map generation and layer-wise transmission offloading. Fortunately, emerging MEC provides a low-latency paradigm for data transmission. But the available edge computing resources may be variational practically. Therefore, this chapter discusses two cases, i.e., deterministic edge capacity case and uncertain edge capacity case. For the deterministic edge capacity case, the optimal solution is obtained analytically. And the influence of different edge server capacity, federated analytics accuracy, vehicle utility is also discussed. For the uncertain edge capacity case, the problem is reformulated into a DRCCO optimization problem and solved by the CVaR model based approximation. The experiments demonstrate the CVaR model based approximation is able to find near-optimal solutions with much less time consumption. In the future work, we will investigate more topics on real-time HD map refreshing and maintenance mechanisms, such as adaptive resolution with communication resources constrained HD updating, efficiency or quality of service (QoS) guaranteed HD map distribution scheme, etc.

4 Matching Theory Based Low-Latency Scheme for Multi-Task Federated Learning in MEC Networks

4.1 Introduction

The last decade has witnessed an unprecedented improvement and prosperity of machine learning techniques and applications, such as face recognition, driverless vehicles, and autonomous disease diagnose, etc. On the one hand, such rapid development is heavily dependent on the tremendous available data generated by the ever-increasing number of users. According to the anticipation of International Data Corporation, the number of devices connected to the Internet will achieve 80 billions and the amount of generated data can be as much as 180 trillion gigabytes in 2025 [87]. On the other hand, thanks to the evolution of powerful computation hardware design and efficient computing architecture, like parallel high performance graphics processing units (GPUs), those computation-intensive machine learning applications finally are able to be performed on devices instead of centralized cloud data centers, which also promotes the extensive use of machine learning [88].

However, it is acknowledged that training a machine learning model relies heavily on enormous data while the data generated by one single device is limited. At the same time, due to the diversity property of individual behavior characteristics, the machine learning model obtained from one device is hard to work desirably for others [89]. Besides, for traditional machine learning, the model is trained via a centralized manner, i.e., all the training data have to be uploaded to a centralized cloud through a wire or wireless channel, which increases the risk of privacy leakage [90]. Therefore, proposing a framework that can unite multiple devices to collaboratively train a universal model and guarantee the privacy safety to a certain extent simultaneously is indispensable, which motivates federated learning coming into being.

Federated learning is firstly proposed by [91], which is a machine learning framework that allows end devices to jointly train a global machine learning model in a decentralized paradigm without sharing individual data. Typically, there will be multiple user devices involving in a federated learning tasks. Firstly, an initialized machine learning model

will be broadcast to all of the participants. Having received the naive model, each participant will optimize the model based on their own data through, for example, stochastic gradient descent method for a certain number of iterations. Then, the model updates, i.e., the calculated model parameters by each participant, will be uploaded to an aggregator. Thereupon, the aggregator performs a weighted average among all the parameters to obtain relatively optimal global model parameters, which will be fed back to all the participants again. The procedures will be conducted repetitively until the pre-defined accuracy is achieved.

Intuitively, there are several reasons to adopt federated learning in practical situation. Firstly, with the development of multi-access edge computing (MEC) networks, there will be many available edge nodes deployed at the edge, which are much closer to the user devices. Invested with adequate computing resources, edge node can be sufficiently powerful to process federated learning tasks instead of using centralized data centers so as to reduce the transmission latency [92]. Secondly, since in federated learning, what transmitted between participants and aggregator are the machine learning model parameters rather than raw data, whose size is much smaller. Therefore, the communication cost can be reduced significantly [93]. At the same time, this manner can also decrease the probability of eavesdropping to a certain extend such that the privacy can be guaranteed [94]. Thirdly, due to the diversity of individual behavior characteristics, the distributions of data generated by different devices are disparate. Fortunately, federated learning has been proved that it is effective to deal with non-identical and independent distribution (non-i.i.d.) data [95], which is suitable for large scale IoTs scenarios.

With all the alluring benefits above, federated learning is also faced with new challenges to tackle. On the one hand, the majority of existing literature make a desirable assumption that once the end devices are invited, they will unconditionally take part in the federated learning tasks that is not practical in the real world. From the perspective of participants, resources cost and willingness caused by machine learning model training have to be took into consideration. For example, when the remained power is lower than a threshold, the device can be unwilling to join any tasks. Otherwise, its normal

functions are not able to be supported. Secondly, in a MEC network, the end devices are always on the go, which means the time of devices localizing within the network is limited. Although edge computing paradigm can reduce the latency to a certain extend, how to reduce the delay and save more time needs to be discussed further. Thirdly, there are many available edge nodes in a MEC network, while each node can work as an agrregator actually. Therefore, how to parallelly perform multiple federated learning tasks, i.e., multi-task federated learning, in a low-latency purpose to augment efficiency needs to be considered as well.

4.2 Related work

As a promising distributed learning paradigm, federated learning has become a popular field of research. [96] analyzes the convergence bound for federated learning with non-i.i.d. data and proposes a control algorithm to achieve an optimal trade-off between local updates and global aggregation considering a resource budget constrain. [97] proposes a sparse ternary compression framework to reduce the communication cost for federated learning with non-i.i.d. data. [76] proposes an over-the-air computation based approach for the fast global aggregation process so as to maximize the number of participants under limited bandwidth, which can improve the accuracy of federated learning. [98] proposes a contract theory based method to build an incentive mechanism to motivate the participants with high-accuracy local training to take part in the collaborative learning process for efficient federated learning. [99] proposes a fast convergence algorithm to find an optimal trade-off between computation and communication latencies as well as overall federated learning time and user device energy consumption so as to enhance the performance of federated learning in wireless networks. [100] proposes a multi-dimensional contract-matching incentive framework to maximization the profit of model owners in a unmanned aerial vehicle (UAV) enabled (Internet of Vehicles) IoVs scenario. [101] utilizes a multi-objective evolutionary algorithm to simultaneously minimize the communication cost and maximize the global model accuracy. However, regarding these work, [96, 97, 76] make a desirable assumption that once the end devices are invited, they will unconditionally take part in the federated learning tasks, which is

not practical in the real world. Besides, for [96, 97, 76, 98, 99, 100, 101], only onefold federated learning task is discussed and multi-task federated learning is not considered.

As for matching theory, it is often used to address the combinatorial problem of players in two sets, based on the preferences of each player and the individual information [102]. [103] proposes an algorithm that combines the Markov decision process with random serial dictatorship matching to solve the UAV-assisted charging problem for energy constrained devices. [104] proposes a student project allocation game based matching method to address the joint radio and computation resource allocation problem for IoTs in the fog computing scenario. [105] proposes a two-sided matching solution for IoTs and edge nodes matching problem to reduce the average service time so that quality of service (QoS) requirements can be achieved. [106] develops an efficient task-virtual machine matching algorithm that jointly considers task execution time and energy consumption to make computation offloading decisions in ultra-dense wireless networks. [107] proposes a matching based strategy for virtual machine placement so as to minimize the system response time and requests dropping for industrial IoTs applications. [108] applies a matching theory based approach to solve the computation offloading problem utilizing parked vehicles such that the more tasks can be accomplished within a certain time range. Whereas, for the work mentioned above, the matching game is considers under the complete preference list situation, which is not practical in large-scale IoTs applications.

4.3 System model and Problem Formulation

In this section, we introduce the preliminaries for federated learning in Subsection 4.3.1. Then, the computation and communication model are discussed in Subsections 4.3.2 and 4.3.3, respectively. At the last, we describe the scenario in details and provide the corresponding formulation for multi-task federated learning latency minimization problem within the MEC network in Subsection 4.3.4. For a clear understanding of parameters and symbols in this chapter, their definitions and descriptions are provided in Table 10 in details.

Table 10: Parameter and symbol description

Notation	Definition
$\mathcal{N}$	Set of participants.
$\mathcal{E}$	Set of edge nodes.
N	Total number of participants.
E	Total number of edge nodes.
$\mathcal{N}_i$	The ith participant.
$\mathcal{E}_j$	The jth edge node.
D	Total amount of data.
D_i	The amount of data for the ith participant.
ω_i	Weights of the ith participant's machine learning model.
ω_{glob}	Weights of aggregated global machine learning model.
f_i	CPU frequency of the ith participant.
m_i	The number of instructions to process a piece of data in the ith participant.
ϵ_i	Local accuracy of the ith participant.
T_i^{cmp}	Time consumption of local training for the ith participant.
r_i	Transmission data rate.
B	Channel bandwidth.
p_i	Transmission power of the ith participant.
h_{ij}	Channel gain of the link between the ith participant and the jth edge node.
N_0	Gaussian noise.
s_i	The number of bits of the ith participant's local model parameters.
T_{ij}^{com}	Time consumption of communication between the ith participant and the jth edge node.
a_{ij}	Index of participant-edge node pair designation.
q_j	Capacity of the jth edge node.
δ	Energy threshold of willingness for participation.
c_i	Remaining battery percentage of the ith participant.
θ_i	Willingness of participation for the ith participant.
$\mathcal{M}$	A matching assignment.
$\mathcal{A}(\mathcal{E}_j)$	Acceptable participants set for the jth edge node.
$\mathcal{A}(\mathcal{N}_i)$	Acceptable edge nodes set for the ith participant.
$L(\mathcal{E}_j)$	Preference list of the jth edge node.
$L(\mathcal{N}_i)$	Preference list of the ith participant.

4.3.1 Federated Learning Preliminaries

Due to different customer expectations and daily usage habits, the generated data by each end device can be different, i.e., the data are non-i.i.d.. Therefore, the object of federated learning is to cooperatively train a global optimal machine learning model for a certain group of devices or users $\mathcal{N} = \{1, ..., N\}$, where the obtained model can be applied to any user within the network. Correspondingly, the individual dataset can be denoted as D_i, which is in a vector form (x_i, y_i), where x_i describes a diverse input

data features and y_i represents the output or label. Based on the task requirements, the devices will perform a certain local iterations to minimize the loss function l_i, i.e.,

$$\min_{\omega} l_i\left(x_i, \omega; y_i\right), \tag{75}$$

which is different according to the specific purpose. For instances, local loss function can be

$$l_i(\omega) = \frac{1}{2}\left(x_i^T \omega - y_i\right), y_i \in \mathbb{R} \tag{76}$$

for linear regression problem or

$$l_i(\omega) = \max\left\{0, 1 - y_i x_i^T \omega\right\}, y_i \in \{-1, 1\} \tag{77}$$

for a logistic regression problem using vector support machine. After a certain number of rounds, each device will upload their own model parameters to the aggregator, i.e., the matched MEC server in this work, to perform a weighted average, which can be described as

$$\omega = \frac{\sum_{i=1}^{N} D_i \omega_i}{D}, \tag{78}$$

where $D = \sum_{i=1}^{N} D_i$ is the total amount of data. Intuitively, the percentage of local parameters to form the global model is proportional to its data size.

Having finished aggregation process, the calculated parameters will be distributed to all the participated devices and perform local updates. After a certain number of similar interactions, once the maximum number of iteration or required accuracy is achieved, the whole process comes to an end. Overall, the objective function of federated learning can be written as

$$\min_{\omega \in \mathbb{R}^d} J(\omega) = \frac{1}{N} \sum_{i=1}^{N} l_i(\omega). \tag{79}$$

To summarize, each global epoch can be divided into three steps: local computation, participants-aggregator interaction, and recomputation. The corresponding process is illustrated in Fig. 23.

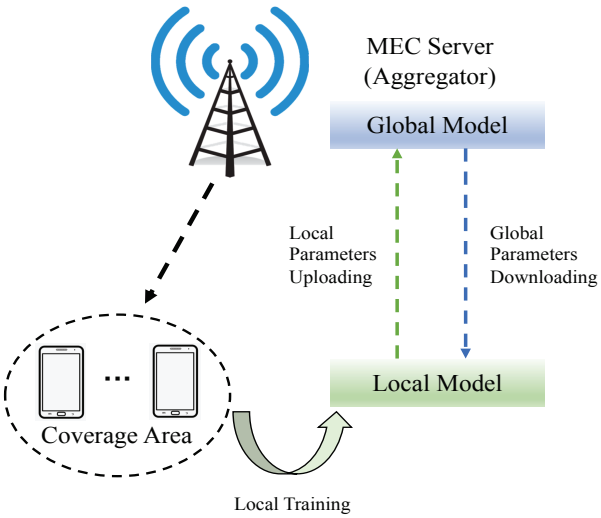


Figure 23: The federated learning procedures.

4.3.2 Local Computation Model

Generally, the involved devices can be mobile phones, IoTs, and IoVs, whose computation abilities are not as powerful as MEC or cloud servers. Hence, their computation tasks are almost accomplished through central processing units (CPUs). We define the CPU frequency for device $\mathcal{N}_i$ as f_i. The required number of CPU cycles to process a piece of data sample is m_i. We should note that the value m_i will be influenced by the model type, such as support vector machine (SVM), long-short term memory (LSTM), deep neural network (DNN), convolutional neural network (CNN), and the adopted training methods, such as stochastic gradient descent (SGD), mini-batch stochastic gradient descent (mBSGD), or Adam. Here, we assume that all the clients are optimizing the same type of machine model via the same kind of training method. Therefore, as is suggested in [109], the consumed time for local device $\mathcal{N}_i$ to perform one local iteration can be written as

$$t_i^{cmp} = \frac{D_i m_i}{f_i}. \tag{80}$$

Obviously, from the perspective of latency, the device with a higher CPU frequency is preferable for MEC server. Besides, the threshold to upload local parameters to matched server can be defined as achieving a certain local accuracy ϵ_i, where a lower ϵ_i indicates a higher prediction accuracy. To obtain the desirable accuarcy, the requisite number of local iterations can be described as $\log(\frac{1}{\epsilon_i})$ [110]. Therefore, for device $\mathcal{N}_i$,

the consumptive time for one local updates can be calculated as

$$T_i^{cmp} = \log\left(\frac{1}{\epsilon_i}\right)\frac{D_i m_i}{f_i}. \tag{81}$$

4.3.3 Communication Model

For federated learning, communication happens each time when participated devices upload local parameters and MEC servers broadcast aggregated global parameters. In this work, we adopt time-division medium access (TDMA) technology as the communication protocol. Without loss of generality, for other protocols, similar approaches can be easily extended. Besides, it is assumed that each device is allocated an orthogonal sub-channel and the interference brought by neighbor users can be ignored. For device $\mathcal{N}_i$, the transmission rate can be described as

$$r_i = B\log_2\left(1 + \frac{p_i h_{ij}}{N_0}\right), \tag{82}$$

where B is the sub-channel bandwidth allocated to device $\mathcal{N}_i$, p_i is transmission power, h_{ij} is channel gain between device $\mathcal{N}_i$ and matched MEC server $\mathcal{E}_j$, and N_0 is the Gaussion noise. Suppose for all the devices connected to the same MEC server have the same local parameter size s_i bits. Intuitively, the required time for communication can be characterized as

$$T_{ij}^{com} = \frac{s_i}{B\log_2\left(1 + \frac{p_i h_{ij}}{N_0}\right)}. \tag{83}$$

Overall, for device $\mathcal{N}_i$ associated with MEC server $\mathcal{E}_j$ in one single global iteration, the total amount of time consumed can be written as

$$T_{ij} = a_{ij}(T_{ij}^{com} + T_i^{cmp}), \tag{84}$$

where a_{ij} denotes the index for pair designation, while $a_{ij} = 1$ denotes device $\mathcal{N}_i$ is paired with edge node $\mathcal{E}_j$, and vice versa.

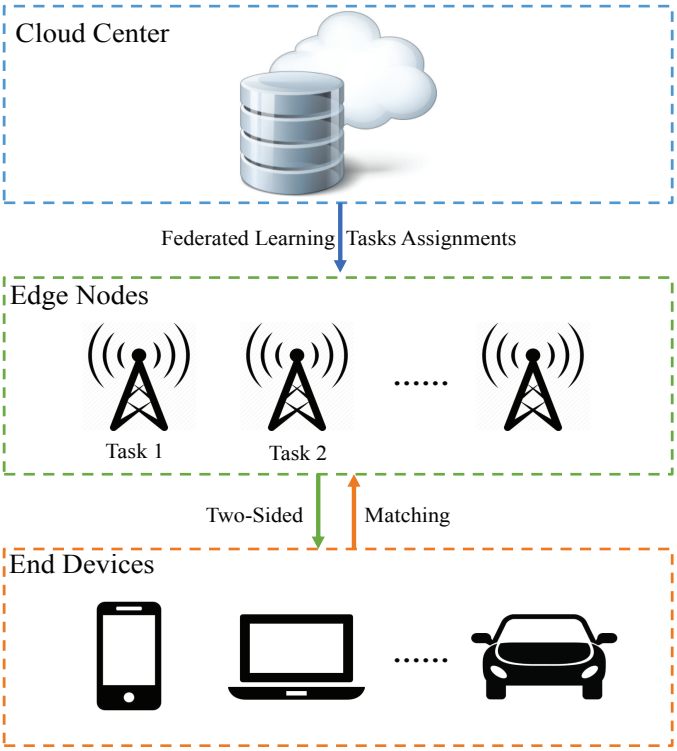


Figure 24: The multi-task federated learning framework in MEC scenario.

4.3.4 Problem Formulation

We consider the latency minimization problem for multi-task federated learning in a MEC network with many users $\mathcal{N} = \{1, ..., \mathcal{N}_i, ..., \mathcal{N}_N\}$ and several edge nodes $\mathcal{E} = \{1, ..., \mathcal{E}_j, ..., \mathcal{E}_E\}$, where each edge node $\mathcal{E}_j$ is supposed to accomplish the designated machine learning task which is different from all the other edge nodes. And each edge node will be assigned a disparate federated task. The scenario is illustrated in Fig. 24. In this work, we focus on the one global aggregation round delay minimization problem. Firstly, we consider the scenario that the mobile phones, IoTs or IoVs work as participants and edge server performs as aggregator. Due to the limitation of edge server coverage and the mobility of IoTs and IoVs, some devices can only take part in one round of federated learning in practice. Secondly, actually the proposed matching can be applied in each global aggregation round, which means if in each round the latency is minimized, then the overall latency is minimized as well. Thirdly, experiments show that if the number of local computation iterations are sufficient or local accuracy can be achieved sufficiently high, the global model can achieve the high accuracy as well even with one round global communication [111, 112]. Therefore, we only need to consider

the latency minimization problem in one communication round.

Due to the fact of diversity of devices computation abilities, from the edge nodes in the latency perspective, the data providers with relatively higher CPU frequency are desirable. On the other hand, the end devices is usually powered by batteries which is energy-constrained. Therefore, in order to proceed tasks as many as possible, providing owned data to the MEC server with better channel gain h_{ij} is preferable. Besides, when the remained power is lower than a threshold, the device can be unwilling to participate any federated learning. Otherwise, its owned function cannot be supported. From the perspective of a system or network, the aim is to find the optimal device-server pair so as to minimizing the overall time consumption. Because we consider only those device whose battery energy is higher than a certain percentage δ will take part in federated learning tasks, the parameter θ_i is introduced, which is denoted as the willingness for participation of the ith device. Intuitively, θ_i is determined by the remained energy percentage, i.e., $\theta_i = 1$ when $\delta \leq c_i$, and vice versa. Overall, the corresponding problem can be formulated as following,

$$
\begin{aligned}
&\min \sum_{i=1}^{N} \sum_{j=1}^{E} \theta_i T_{ij}, \\
&s.t. \\
&\quad C1 : a_{ij} \in \{0,1\}, \forall \mathcal{N}_i \in \mathcal{N}, \forall \mathcal{E}_j \in \mathcal{E}, \\
&\quad C2 : \sum_{j} a_{ij} \leqslant 1, \forall \mathcal{N}_i \in \mathcal{N}, \\
&\quad C3 : \sum_{i} a_{ij} \geqslant 1, \forall \mathcal{E}_j \in \mathcal{E}, \\
&\quad C4 : \sum_{i} a_{ij} \leqslant q_j, \forall \mathcal{E}_j \in \mathcal{E}.
\end{aligned}
\tag{85}
$$

Here, C1 is the designation element, where $a_{ij} = 1$ denotes device $\mathcal{N}_i$ is connected with edge node $\mathcal{E}_j$, otherwise they are not paired. C2 describes each device can only connects to one edge node. C3 represents that for a specific edge node, there will be at least one device connects to it and assists to accomplish the assigned task. C4 denotes the number of connected participants for edge node $\mathcal{E}_j$ cannot exceed its capacity q_j.

Evidently, this optimization problem is a 0-1 integer programming problem, which

is generally NP-hard to solve. Hence, we are motivated to find a feasible suboptimal solution. Therefore, we utilize a matching theory based approach: the HR problem with incomplete preference list, which will be discussed in details in the next section.

4.4 Methodology

In the previous section, we have formulated the latency minimization problem as a 0-1 integer programming problem. Because of NP-hardness, we propose a matching theory based method to find the suboptimal solution. Besides, due to the fact that the number of participants in IoTs and IoVs scenario can be pretty large, different from the traditional matching game, the preference list cannot be generated completely, which means each side cannot fully obtain all the information of the other side in practice. Therefore, we propose HR problem with incomplete preference list based solution in this section. Specifically, the HR modelling is introduced in Subsection 4.4.1, and the proposed solution is provided in Subsection 4.4.2.

4.4.1 HR Problem Allocation Modelling with Incomplete Preference List

The HR problem, also sometimes named as the college admission problem was firstly proposed by Gale and Shapley [113]. Each year, there will be many medical students looking for hospitals to do practical training. And each hospital will provide a certain number of available positions for them. From the perspective of medical students, each of them will have an order of hospitals indicating which one is more preferable to join. Simultaneously, hospitals will also review their application materials to decide the order of which ones they prefer to hire. Obviously, in the HR problem, each student can only be assigned to one hospital for practical training. But for a specific hospital, it will provide a certain number of positions for medical students. Therefore, it is actually a two-sided many-to-one matching problem.

Inspired by the HR problem, we can model the latency minimization problem for multi-task federated learning as the HR game. Because in fact, the matching problem between edge nodes and participants is also a two-sided many-to-one problem. The edges nodes can be considered as hospitals which need to hire many participants to

finish federated learning task for them. Meanwhile, the participants can be regarded as medical students, providing their data and computing resources for edge nodes, and each participant can only be assigned to a specific edge node. Intuitively, the problem involves a set of participants $\mathcal{N}$, a set of edge nodes $\mathcal{E}$, and a set of acceptable pairs $\mathcal{H} = \mathcal{N} \times \mathcal{E}$. The capacity of each edge node $\mathcal{E}_j$ should be a positive integral and every edge node has a set of acceptable participants $\mathcal{A}(\mathcal{E}_j)$, where

$$\mathcal{A}(\mathcal{E}_j) = \{\mathcal{N}_i \in \mathcal{N} : (\mathcal{N}_i, \mathcal{E}_j) \in \mathcal{H}\}. \tag{86}$$

At the same time, each participant $\mathcal{N}_i$ must be accepted by one and only one edge node. Likewise, the acceptable edge nodes options for resident $\mathcal{N}_i$ can be denoted as

$$\mathcal{A}(\mathcal{N}_i) = \{\mathcal{E}_j \in \mathcal{E} : (\mathcal{N}_i, \mathcal{E}_j) \in \mathcal{H}\}. \tag{87}$$

Let us define an agent $k \in \mathcal{N} \times \mathcal{E}$ for the HR problem, which has a preference list in which it ranks $\mathcal{A}(k)$ in a strict order, i.e. no matching candidates share the same preference. Given any participant $\mathcal{N}_i \in \mathcal{N}$ and edge nodes $\mathcal{E}_j, \mathcal{E}_p \in \mathcal{E}$, we can say $\mathcal{N}_i$ prefers $\mathcal{E}_j$ to $\mathcal{E}_p$ if $\mathcal{E}_j$ precedes p on $\mathcal{N}_i$'s preference list, under the condition that $(\mathcal{N}_i, \mathcal{E}_j) \in \mathcal{H}$ and $(\mathcal{N}_i, \mathcal{E}_p) \in \mathcal{H}$. Similarly, the preference relation can be defined for edge nodes.

A matching assignment $\mathcal{M}$ is a subset of $\mathcal{H}$. We can say $\mathcal{N}_i$ is assigned to $\mathcal{E}_j$ or $\mathcal{E}_j$ is assigned to $\mathcal{N}_i$ if the pair $(\mathcal{N}_i, \mathcal{E}_j) \in \mathcal{M}$. Once $\mathcal{N}_i$ and $\mathcal{E}_j$ is paired and the relation is no longer changeable, the matching $\mathcal{M}$ is stable. The stability notion here implies the robustness to deviations that can be beneficial to both participants and edge nodes [114]. An unstable matching indicates that participant can change the connected edge node if the altering is beneficial to both of them. However, this kind of unstability is not desirable regarding to the network operation and resources utility. For federated learning, the time of interaction between edge nodes and participants can be more than once. Reconnecting to a new edge node can make lead to the uselessness of current model. Besides, in multi-task federated learning scenario, different edge nodes have different tasks, which means the loss function can be various. Therefore, for a specific

federated learning, all the local training procedures need to be restarted, which is also a waste of time as well as computing resources. Therefore, stability is fatal for matching and here we give the formal stability definition.

Definition 4 *Let $\mathcal{M}$ be a matching in HR. A pair $(\mathcal{N}_i, \mathcal{E}_j) \in \mathcal{H} \setminus \mathcal{M}$ blocks $\mathcal{M}$, or $(\mathcal{N}_i, \mathcal{E}_j)$ is a blocking pair for $\mathcal{M}$, if the following conditions are satisfied regarding to $\mathcal{M}$:*

1. *$\mathcal{N}_i$ is unassigned or prefers $\mathcal{E}_j$ to $\mathcal{M}(\mathcal{N}_i)$;*
2. *$\mathcal{E}_j$ is under-subscribed or prefers $\mathcal{N}_i$ to at least one member of $\mathcal{M}(\mathcal{E}_j)$ (or both).*

Then $\mathcal{M}$ is said to be stable if it admits no blocking pair.

When matching achieves stable, no candidate can find a more preferable partner than current one and no new matching process will be proposed. In Definition 4, $\mathcal{M}(\mathcal{N}_i)$ indicates the matching of participant $\mathcal{N}_i$ in matching $\mathcal{M}$. In this chapter, a blocking pair can be defined as a pair of participant and edge node (i, j), where participant $\mathcal{N}_i$ prefers edge node p to its current mate $\mathcal{E}_j$ and edge node $\mathcal{E}_j$ prefers participant q to its current mate $\mathcal{N}_i$. However, traditional HR matching is essentially a two-sided many-to-one matching game. And the preference list of both sides are complete, i.e., for $\forall \mathcal{E}_j \in \mathcal{E}$, each $\mathcal{N}_i \in \mathcal{N}$ has a strict order list to indicate the preference, and vice versa. However, due to the fact that the number of participants in IoTs and IoVs scenario can be very large, it is impossible for each edge node to acknowledge the computing abilities or CPU frequencies of all the participants, which leads to the incompleteness of preference list. Therefore, considering the incomplete preference list in HR game is indispensable.

HR problem with incomplete preference list is actually a variant of standard HR problem. The only difference between them is the completeness of preference list, which can be described by an example shown in Fig. 25. The majority of notations and definitions in the HR problem can be applied here directly. But for stability, we can redefine as the following.

Definition 5 *Let $\mathcal{M}$ be a matching in HR with the incomplete preference list. A pair $(\mathcal{N}_i, \mathcal{E}_j) \in \mathcal{H} \setminus \mathcal{M}$ blocks $\mathcal{M}$, or $(\mathcal{N}_i, \mathcal{E}_j)$ is a blocking pair for $\mathcal{M}$, if the following conditions are satisfied regarding to $\mathcal{M}$:*

$\mathcal{E}_1 : \mathcal{N}_1\ \mathcal{N}_3\ \mathcal{N}_2\ \mathcal{N}_4\ \mathcal{N}_5$	$\mathcal{N}_1 : \mathcal{E}_2\ \mathcal{E}_1\ \mathcal{E}_3\ \mathcal{E}_4\ \mathcal{E}_5$	$\mathcal{E}_1 : \mathcal{N}_1\ \mathcal{N}_3\ \mathcal{N}_2$	$\mathcal{N}_1 : \mathcal{E}_2\ \mathcal{E}_1\ \mathcal{E}_3\ \mathcal{E}_4\ \mathcal{E}_5$
$\mathcal{E}_2 : \mathcal{N}_3\ \mathcal{N}_1\ \mathcal{N}_5\ \mathcal{N}_2\ \mathcal{N}_4$	$\mathcal{N}_2 : \mathcal{E}_2\ \mathcal{E}_1\ \mathcal{E}_4\ \mathcal{E}_5\ \mathcal{E}_3$	$\mathcal{E}_2 : \mathcal{N}_3\ \mathcal{N}_1$	$\mathcal{N}_2 : \mathcal{E}_2\ \mathcal{E}_1$
$\mathcal{E}_3 : \mathcal{N}_2\ \mathcal{N}_1\ \mathcal{N}_5\ \mathcal{N}_4\ \mathcal{N}_3$	$\mathcal{N}_3 : \mathcal{E}_1\ \mathcal{E}_2\ \mathcal{E}_3\ \mathcal{E}_5\ \mathcal{E}_4$	$\mathcal{E}_3 : \mathcal{N}_2\ \mathcal{N}_1$	$\mathcal{N}_3 : \mathcal{E}_1\ \mathcal{E}_2$
$\mathcal{E}_4 : \mathcal{N}_3\ \mathcal{N}_2\ \mathcal{N}_4\ \mathcal{N}_5\ \mathcal{N}_1$	$\mathcal{N}_4 : \mathcal{E}_3\ \mathcal{E}_1\ \mathcal{E}_4\ \mathcal{E}_2\ \mathcal{E}_5$	$\mathcal{E}_4 : \mathcal{N}_3\ \mathcal{N}_2\ \mathcal{N}_4\ \mathcal{N}_5$	$\mathcal{N}_4 : \mathcal{E}_3\ \mathcal{E}_1\ \mathcal{E}_4$
$\mathcal{E}_5 : \mathcal{N}_3\ \mathcal{N}_4\ \mathcal{N}_2\ \mathcal{N}_5\ \mathcal{N}_1$	$\mathcal{N}_5 : \mathcal{E}_4\ \mathcal{E}_3\ \mathcal{E}_1\ \mathcal{E}_2\ \mathcal{E}_5$	$\mathcal{E}_5 : \mathcal{N}_3\ \mathcal{N}_4\ \mathcal{N}_2$	$\mathcal{N}_5 : \mathcal{E}_4\ \mathcal{E}_3$

(a) Matching with Complete Preference List. (b) Matching with Incomplete Preference List.

Figure 25: The example of matching edge nodes $\mathcal{E}_i$ with end devices $\mathcal{N}_j$ in both complete and incomplete preference list cases. The orders indicates each individual's preference list.

1. *$\mathcal{N}_i$ is unassigned or prefers $\mathcal{E}_j$ to $\mathcal{M}(\mathcal{N}_i)$;*

2. *$\mathcal{E}_j$ is unassigned or prefers $\mathcal{N}_i$ to $\mathcal{M}(\mathcal{E}_j)$.*

Then $\mathcal{M}$ is said to be stable if it admits no blocking pair.

Therefore, from the perspective of stability, let us look back to the example in Fig. 25. In Fig. 25(a), for the complete preference list case, suppose $\mathcal{E}_1$ proposes matching firstly, so it tries to match with the most desirable candidate, i.e., $\mathcal{N}_1$, according to preference list. But $\mathcal{N}_1$ prefers $\mathcal{E}_2$ to $\mathcal{E}_1$ and $\mathcal{E}_2$ is exactly available at this time. So $(\mathcal{E}_1, \mathcal{N}_1)$ is a blocking pair and $\mathcal{N}_1$ will propose to match with $\mathcal{E}_2$. Likewise, $\mathcal{E}_1$ tries to match with the next candidate $\mathcal{N}_3$. $\mathcal{N}_3$ exactly prefers $\mathcal{E}_1$ best so they will form a pair. Similarly, when all the process is done, we can see the the final stable matching results are $(\mathcal{E}_1, \mathcal{N}_3), (\mathcal{E}_2, \mathcal{N}_5), (\mathcal{E}_3, \mathcal{N}_1), (\mathcal{E}_4, \mathcal{N}_2)$, and $(\mathcal{E}_5, \mathcal{N}_4)$. Whereas, for the matching in Fig. 25(b) with incomplete preference list, the doubtless stable pairs are $(\mathcal{E}_1, \mathcal{N}_2), (\mathcal{E}_2, \mathcal{N}_3)$, and $(\mathcal{E}_4, \mathcal{N}_5)$. As for the unpaired individuals, i.e. $\mathcal{E}_3$, $\mathcal{E}_5$, $\mathcal{N}_1$, and $\mathcal{N}_4$, there exist two possibilities to combine them, i.e., $(\mathcal{E}_3, \mathcal{N}_1)$ and $(\mathcal{E}_5, \mathcal{N}_4)$ or $(\mathcal{E}_3, \mathcal{N}_4)$ and $(\mathcal{E}_5, \mathcal{N}_1)$. However, according to Definition 5, we can see that both of the options belong to stable matching. Therefore, for HR problem with the incomplete preference list, one important conclusion is that the stable matching can be partial. Besides, it has been proved that for HR with incomplete preference list problems, there may be more than one stable matching, but their size is all the same and one of them can be obtained in poly time [115][116]. In the next subsection, the algorithms proposed to solve the corresponding problem will be discussed.

4.4.2 Participants and Edge Nodes Pairing

As is discussed above, edge nodes need to hire many participants to finish federated learning task for them. Accordingly, the participants will provide their data and computing resources for edge nodes. Besides, considering the power constrain in practice, edge nodes can only choose those participants whose remaining battery capacity is larger than the pre-defined threshold δ. Therefore, the acceptable participants set for edge nodes $\mathcal{E}_j$ ($\forall \mathcal{E}_j \in \mathcal{E}$) can be defined as

$$\mathcal{A}(\mathcal{E}_j) = \{\mathcal{N}_i \in \mathcal{N} \mid \delta < c_i\}. \tag{88}$$

For participants, practically they can work for any edge nodes. Therefore, the acceptable edge node set for participants is the universe set of edge node, which can be written as

$$\mathcal{A}(\mathcal{N}_i) = \{\mathcal{E}_j \in \mathcal{E}\}. \tag{89}$$

The preference list is based on a private view and can be defined according to utility function. Making use of the computation model described in Section III, i.e., in (7), each edge node is able to calculate the time consumption for each connected participant. Therefore, a preference list of edge node $\mathcal{E}_j$ can be established based on the computation time T_i^{cmp*}, where T_i^{cmp*} indicates the least time consumption. Correspondingly, we can define the preference list of edge node $\mathcal{E}_j$ as

$$L(\mathcal{E}_j) = T_i^{cmp*}, \forall \mathcal{N}_i \in \mathcal{A}(\mathcal{E}_j). \tag{90}$$

Obviously, $L(\mathcal{E}_j)$ is in an ascending order since less time consumption is always preferable.

When it comes to the participants preference over edge nodes, the power consumption is more important. Because, participants are usually end devices with limited capacity batteries. Apart from taking part in the invited federated learning tasks, they need to consider to maintain enough power and time to perform their normal functionalities as well. Therefore, according to (9), suppose the channel between participant and

Algorithm 2 Participants and Edge Nodes Pairing with Incomplete Preference List

Input: participant set $\mathcal{N}$, edge node set $\mathcal{E}$, remaining battery energy for each participant c_i, energy threshold δ, participant CPU frequency f_i, channel gain h_{ij}, edge node capacity q_j;.
All the participants check their remaining battery capacity. Only for those devices meet the requirement $\delta < c_j$ will be the candidates for federated learning tasks.
//Participants build preference list;
for i=1:N **do**
  Build the preference list $L(\mathcal{N}_i)$;
end for
// Edge nodes build preference list;
for j=1:E **do**
  Build the preference list $L(\mathcal{E}_j)$;
end for
while $\exists \mathcal{E}_j \in \mathcal{E}$ is available and edge node has a non-empty preference list **do**
  //Participants propose to match with edge nodes
  for Unmatched $\mathcal{N}_i \in \mathcal{N}$ **do**
    Propose to the top-ranked edge node in participant $\mathcal{N}_i$'s preference list $L(\mathcal{N}_i)$;
    Remove the top-ranked edge node from participant $\mathcal{N}_i$'s preference list $L(\mathcal{N}_i)$;
  end for
  //Edge node overflow notification
  for $\mathcal{E}_j \in \mathcal{E}$ **do**
    if The number of candidates exceeds capacity q_j **then**
      Hold q_j participants based on its preference list $L(\mathcal{E}_j)$ and inform the other participants that they are get rejected from $\mathcal{E}_j$.
    else
      Hold all the participants.
    end if
  end for
end while
//Partial matching checking
for Unmatched $\mathcal{N}_i \in \mathcal{N}$ **do**
  Randomly assign them to the available edge node.
end for
Output: Stable many-to-one matching results.

edge node with a higher channel gain is more desirable. So, the corresponding preference list can be defined as

$$L(\mathcal{N}_i) = T_{ij}^{com*}, \forall \mathcal{E}_j \in \mathcal{A}(\mathcal{N}_i). \tag{91}$$

$L(\mathcal{N}_i)$ is in an ascending order because less communication time is preferable in accordance with a higher channel gain.

Based on the setting and definitions above, we can apply the many-to-one matching algorithm to find a stable matching solution for participants and edge nodes pairing problem, which is described in Algorithm 2. Firstly, every participant will evaluate its remaining power so as to decide whether it is able to involve a federated learning task. Each participant will generate its preference list based on $L(\mathcal{N}_i)$. If the number of selected participants exceeds edge node capacity, edge node $\mathcal{E}_j$ will only keep those desirable participants within capacity and reject the applications of the others. Then, the unmatched participants will continue to match with the available edge nodes. This process iterates until each participant is either matched or rejected by all the edge nodes based on its preference list, i.e., the matching arrives the stable state. However, since the preference list is incomplete, the results can be partial matching as is illustrated in Fig. 25. Therefore, when the matching arrives stable state, checking the assignments for every participant is necessary. If there are still any participants who are willing to join unassigned, they will be assigned to available edge node randomly. For Algorithm 2, we can derive the following proposition.

Proposition 1 *For the Algorithm 2, the proposed many-to-one matching method is able to converge and obtain stable matching results. Besides, as for the performance, the running time of implementation can achieve $\mathcal{O}(N \times E)$, where $(N \times E)$ denotes all the possible matching pairs.*

proof 3 *Let $\mathcal{M}$ be an instance of HR matching. The stable pairs in $\mathcal{M}$ can be found in $\mathcal{O}(N)$ time. And the stable matchings in $\mathcal{M}$ can be listed in $\mathcal{O}(E)$ time per matching, after $\mathcal{O}(N)$ pre-precessing time. Therefore, the overall running time can achieve $\mathcal{O}(N \times E)$. The corresponding proof can be found in [117] and [118] in details.*

■

Table 11: Table of Key Parameters

Simulation Parameters	**Value**
Channel bandwidth B	20MHz
Transmission power p_i	23dBm
Data size D_i	1000
The number of instructions for one piece data processing m_i	50
The size of local model parameters size s_i	5MB
Gaussian noise N_0	-96dBm
Energy threshold δ	40%
CPU frequency f_i	[10,20]MHz
Channel gain h_{ij}	[10,20]
Default local accuracy ϵ_i	0.1

4.5 Numerical results

In this section, we evaluate our proposed method for matching with the incomplete preference list (IPL) with regard to system latency. Meanwhile, we take matching with the complete preference list (CPL) and random matching as baseline methods for performance comparison. Besides, we change the number of participants, the number of edge nodes, the capacity of each edge node, local accuracy, energy threshold, and incomplete rate to see the performance varieties of the MEC network. In order to control variables, we fix some common parameters for all the participants [96, 95, 76]. The channel bandwidth B is set as 20MHz. The transmission power for each device p_i, the amount of data for each device D_i, the number of instructions to process one data sample needed by each participant m_i, and the size of local model parameters s_i are assumed the same, which are set as 23dBm, 1,000, 50, and 5MB, respectively. The Gaussian noise N_0 is set as -96dBm. The energy threshold for every participant is assumed the same as well, which is set as 40%. Remaining energy capacity is randomly assigned within $[20, 100]$%. The CPU frequency is randomly generated within the range $[10, 20]$MHz. The channel gain of the link between the ith participant and the jth participant is randomly generated within the interval [10,20]. The default local accuracy indicator is defined as 0.1, where a lower value of ϵ_i represents the higher local accuracy. The default percentage of preference missing is set as 10%. For the convenience, the parameters settings are summarized in Table 11.

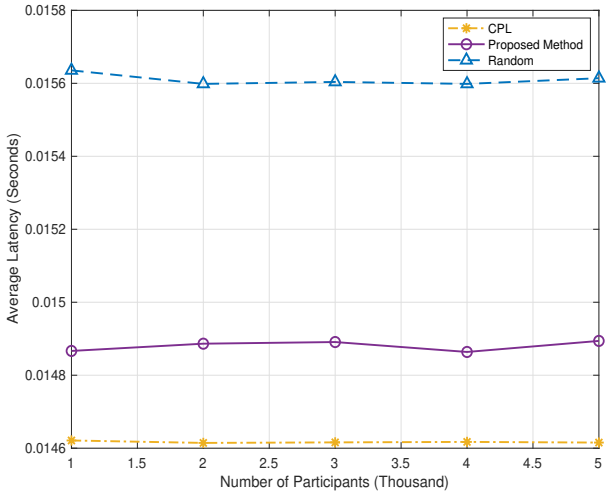


Figure 26: Average network latency with different number of users.

Firstly, we discuss the influence of different participants among participant's average latency. In this case, the number of edge node E is set as 10, where each of them can accept 100 participant at the most. The results are illustrated in Fig. 26. We increase the number of participants from 1,000 to 10,000 by step 1,000 to show the change of system latency. Among the three methods shown in Fig. 26, algorithm with CPL achieves the lowest average latency throughout the variation process. which is understandable. Because with CPL, both sides have entire knowledge to each other, i.e. channel gain and CPU frequency. Therefore, each matching step aims at pairing the higher CPU frequency for edge node and higher channel gain for participant, which approaches the lower latency until stable matching is obtained. However, for matching with IPL, due to the incomplete information, the unassigned individuals will be matched randomly with available edge node, which is not desirable and the latency is not the optimal solution. As for random strategy, it comes by random participants allocation among edge nodes, which gives the highest average latency. Besides, during the variation, the average network latency keeps almost the same for CPL but varies for IPL and random strategies. This is because random method gives different matching results each time. As for IPL, since we set the missing rate as 10%, the missing preferences introduce uncertainties, which leads to fluctuation.

Then, let us have a look at the influence of different numbers of edge nodes upon network latency. We set the number of users as 5,000 and the capacity of edge nodes

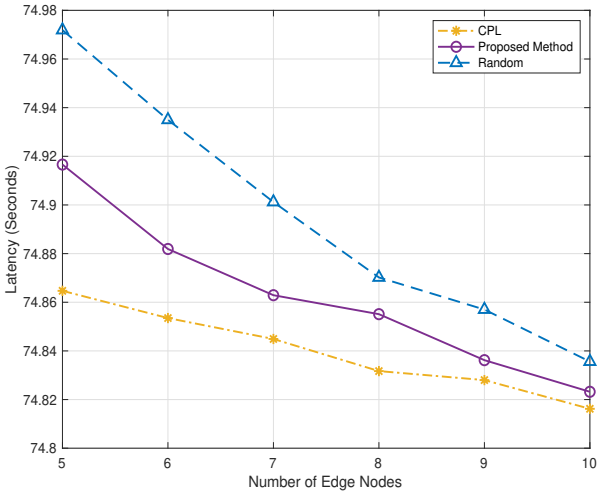


Figure 27: Network latency with different number of edge nodes.

is 1,000. We increase the number of edge nodes from 5 to 10 by step 1. The corresponding results are shown in Fig. 27. Apparently, CPL achieves the lowest latency because the complete information for preference. The performance of proposed method is in the between of random method and CPL due to a part of matching is assigned randomly instead of the utility functions. Likewise, random method generates the worst performance reflecting by the highest latency. When the edge node number is five, CPL achieves 0.1072 second less delay than random method and IPL achieves 0.0533 second less latency than random method. Besides, we can see that with the increase of number of edge nodes, the overall latency goes down, which is understandable. When the number pf edge node is larger, the elements in the acceptable sets $\mathcal{A}(\mathcal{N}_i)$ and $\mathcal{A}(\mathcal{E}_j)$ become larger accordingly, which means the options for participants get augmented. Therefore, each participant is able to find a better assignment with the higher channel gain. Correspondingly, the total number of low latency pairs increases, leading to a relatively lower network delay.

Now, we vary the edge nodes capacity to see the corresponding effect among system latency. We set the number of participants as 5,000 and the number of edge nodes as 10. The capacity of each edge node is increased from 500 to 1,000 by step 100. The corresponding results are illustrated in Fig. 28. Overall, the results keep consistent with previous experiments, i.e., random method yields the highest latency, proposed method gives less, and the CPL achieves the lowest latency. Averagely, the latency of

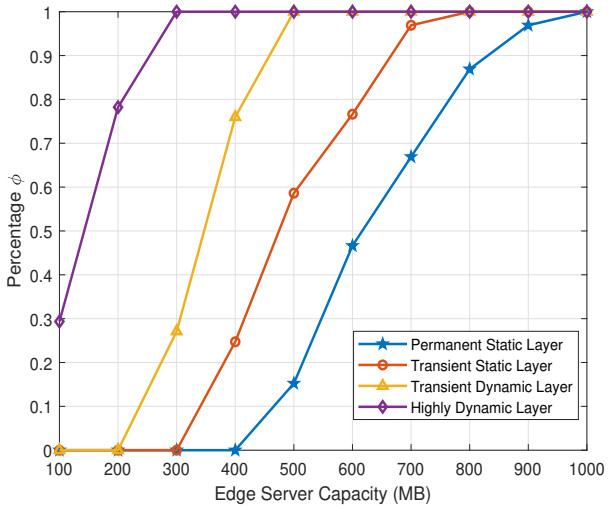


Figure 28: Network latency with different capacity of edge nodes.

random method is 0.0664 second more than CPL method, and 0.0390 second more than IPL method. Moreover, it is obvious that no matter for which method, the latency reduces with the increase of capacity of edge nodes. Because when the edge node is possessed of a larger capacity, participants can obtain a higher probability to be accepted by the desirable edge node instead of rejection, such that the overall latency can be decreased. Furthermore, for CPL, we can find that the latencies for 800, 900, and 1,000 capacity cases are the same. This is because when the edge node capacity is sufficiently large, the matching assignments achieves stable and no more optimal solution can be found. However, for IPL and random methods, due to the embedded uncertainties or randomness, i.e., random preference list missing and random participant-edge node allocation, they cannot achieve the same latency.

Next, we look into the influence of the local accuracy upon network latency. The number of participants, the number of edge nodes, the edge node capacity are set as 5,000, 10, and 500, respectively. The local model accuracy indicator is increased from 0.1 to 0.5 by step 0.1, which represents local accuracy decreases gradually. Corresponding results are illustrated in Fig. 29. Still, the performance of CPL is the best and random method gives the largest latency. Statistically, random strategy yields 2.5691 seconds more delay than CPL and 1.5724 seconds more latency than IPL. Macroscopically, with a higher accuracy, the latency is much larger as well. Actually, what is affected by local accuracy is computation time. According to (81), the total number of local computation

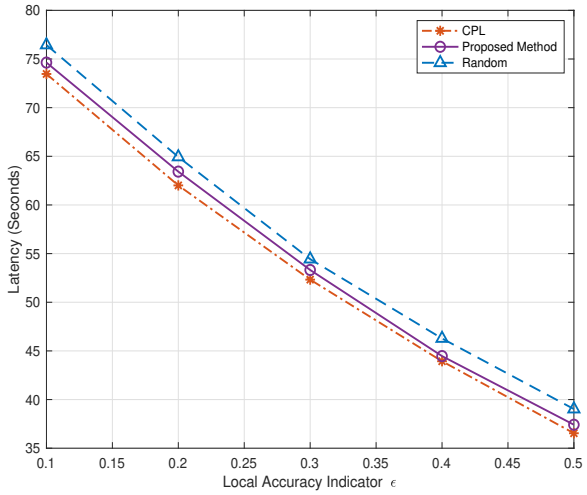


Figure 29: Network latency with different local accuracy.

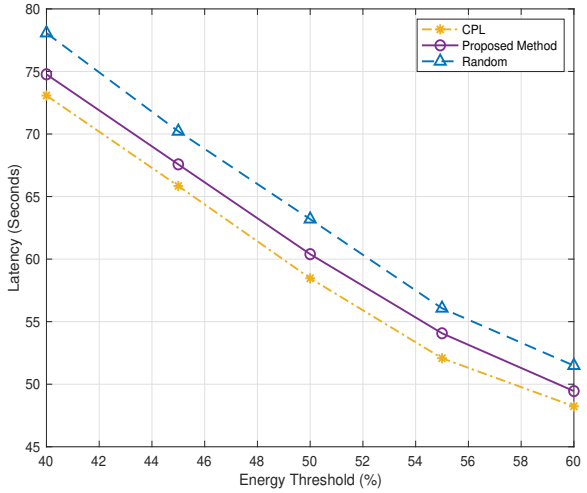


Figure 30: Network latency with different energy threshold.

iterations is in exponential growth with the decay of ϵ. Therefore, in order to achieve a relatively higher local accuracy, the required rounds of computation iterations increases explosively, resulting in longer time delay.

Also, we change the energy threshold to find the corresponding influence among system latency. The number of participants, the number of edge nodes, the edge node capacity are set as 5,000, 10, and 500, respectively. The participation willingness or energy threshold is increased from 40% to 60% by step 5%. The numerical results are shown in Fig. 30. Apparently, random method gives the highest network latency while CPL gives the lowest, which keeps consistent with previous experiments. In the

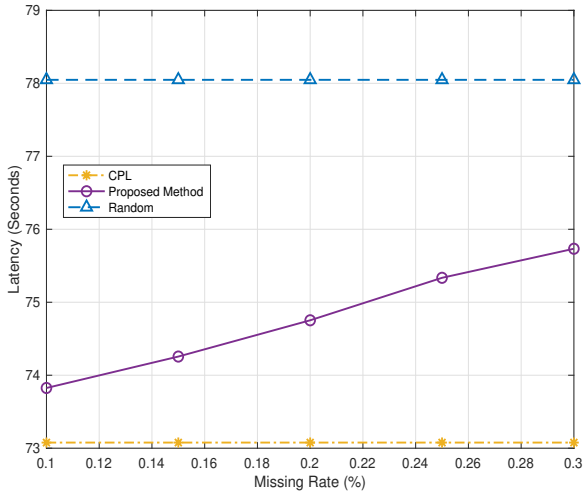


Figure 31: Network latency with different preference list missing rate.

average, random method generates 4.2652 seconds higher latency than CPL and 2.5547 seconds higher latency than IPL. In general, with the increase of energy threshold δ, the total network latency decreases accordingly. The index of willingness for participation is determined by remained energy percentage. Only when δ is less than c_i, θ_i equals to 1, i.e., $\mathcal{N}_i$ will join federated learning task. Hence, as energy threshold increases, the total number of participants involving in the learning process will decrease. Therewith, the overall network latency can be reduced correspondingly.

Finally, we adjust the parameter of preference list missing rate, which is from 10% to 30% by step 5%. The number of participants, the number of edge nodes, the edge node capacity are set as 5,000, 10, and 500, respectively. The corresponding results are illustrated in Fig. 31. The overall trend is that, with the increase of missing rate, the network latency becomes larger. And we can find that the performance approaches to CPL with small missing rate but similar to random method with a larger missing rate. This is because if the missing rate is higher, the probability of partial matching increases as well. Correspondingly, the number of unassigned individuals will be aggrandized. Since unassigned individuals will be matched with available edge node randomly, which is not based on utility function. Consequently, the obtained latency is not the optimal solution. Therefore, the performance get closer to random strategy. In an extreme case, when the missing rate is 100%, IPL will be exactly the same as random allocation. On the contrary, when the missing rate is small, which means the built preference list is

relatively complete, the system latency can achieve similar value compared with CPL. When the missing rate is sufficiently small, i.e., 0%, IPL will turn into CPL case.

4.6 Conclusion

In this chapter, we study the low latency problem for multi-task federated learning in the MEC networks. Considering in large scale IoTs scenario, it is not possible for edge nodes and end devices to obtain the complete information of the other side, which means building the complete preference list is impractical. Therefore, we propose a method to deal with the two-sided many-to-one matching with the incomplete preference list. The simulation results show that the performance of our proposed method is close to the performance of CPL, although there is small gap between them due to information missing. Besides, we also discuss the influence of number of participants, the number of edge nodes, the edge node capacity, local accuracy, energy threshold, and preference list missing rate among network latency. Evidently, the network latency is positively related to the missing rate while is negatively correlated with number of edge nodes, capacity of edge nodes, energy threshold, and local accuracy indicator.

5 Conclusion and Future Works

5.1 Conclusion Remarks

To relieve the data processing pressure, multi-access edge computing is emerged as a solution.Edge computing employs a distributed and hierarchical computing model. The edge nodes, which have computational abilities as well, are geographically deployed. Those relatively lower complexity missions can be addressed in the edge nodes directly instead of uploading to the cloud center, so as to the transmission distance can be significantly minified, therewith the decreasing of latency. However, with the growth of end users, how to allocate the available edge computing resources to fulfill the requirements becomes a challenging problem

This dissertation provides a theoretical research between machine learning, multi-access edge computing, and autonomous vehicles, in which different machine learning models are utilized in different multi-access edge computing assisted applications as well as efficient machine learning mechanism design. This work places a fundamental research on edge computing resource allocation. In this dissertation, we mainly focus our research on how to use machine learning method to allocate edge computing resources. Also, we discuss the mechanism design to optimize the machine learning framework.

In the first application, a two-stage meta-learning approach is proposed to predict the edge computing resource consumption such that the expenses to consume edge computing resources can be minimized. In the first stage, a DNN is utilized to learn the experience dataset so as to figure out which machine learning model performs better in a specific situation.In the second stage, the resource amount anticipation will be conducted by the machine learning model selected by the DNN obtained in the first stage according to the meta-features. In addition,due to the fact that there is no open edge computing based vehicular network dataset, we programming the game engine Unity3D to build the 3D model of Manhattan area as in real world. Meanwhile,we adjust the factors like different roadmaps, the number of vehicles, and the randomness of task sizes for the traffic, which makes our data get closer to the practice. Simulation results show that our proposed meta-learning method always gives the most economical predictions.

Next, the problem of latency minimization for edge assisted HD map updating

is investigated. Especially, consider the practice that the available edge computing resources may be variational. Two cases are discussed, i.e., deterministic edge capacity case and uncertain edge capacity case. For the deterministic edge capacity case, the optimal solution is obtained analytically. And the influence of different edge server capacity, federated analytics accuracy, vehicle utility is also discussed. For the uncertain edge capacity case, the problem is reformulated into a DRCCO optimization problem and solved by the CVaR model based approximation. The numerical results show that our proposed method is able to find the low latency edge computing resource allocation scheme.

Finally, the low latency problem for multi-task federated learning in the MEC networks is discussed. Considering in large scale IoTs scenario, it is not possible for edge nodes and end devices to obtain the complete information of the other side, which means building the complete preference list is impractical. Therefore, we propose a method to deal with the two-sided many-to-one matching with the incomplete preference list. The simulation results show that the performance of our proposed method is close to the performance of CPL, although there is small gap between them due to information missing.

From those works, we have seen machine learning as a powerful prediction tool is able to help arrange or optimize the edge computing resources allocation. Beyond machine learning utilization, the efficient mechanism design for machine learning, especially for distributed learning, is also necessary. This dissertation is expected to provide an accessible and holistic survey on the use of machine learning and optimization methods to address the resource allocation problem for edge computing assisted applications, and has a long term effect on problems such as edge computing resource deployment, service guaranteed wireless network design.

5.2 Future Work

As a promising transportation way, autonomous driving is in fast development, which relies heavily on the utilization of HD map. HD map is made up of various information and resources, such as drivable paths, lane marks, the priority of lanes, traffic

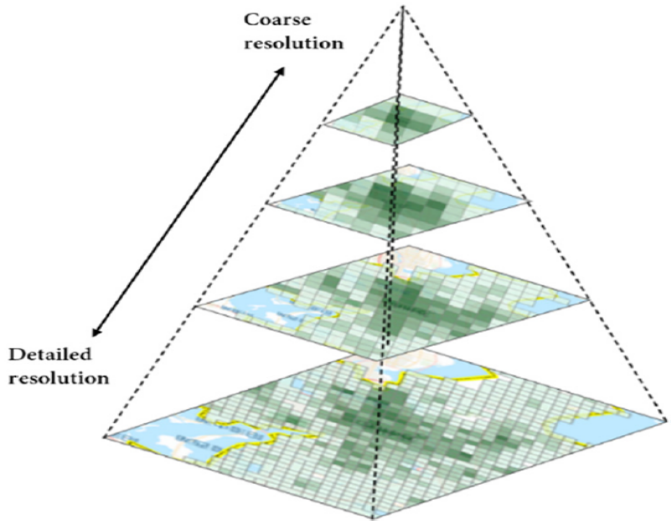


Figure 32: The scenario for adaptive resolution with communication resources constrained HD map updating.

light and crosswalk to lane association, adjacent objects, and street furniture, which is represented in a high degree of resolution and precision, generally in the centimeter level. Intuitively, in the current map, it is common to find the gaps between actual circumstances andthe map, which takes days to be corrected. However, for autonomous driving, any delayed update for HD map can result in dangerous or even fatal accidents without human intervention. Therefore, HD map must be updated in a timely manner. Thus, we see there is a great potential to do further research in edge computing assisted HD map updating, offloading, and distribution problems. The following are research directions that can be further explored in this area of research.

- Adaptive resolution with communication resources constrained HD map updating: In wireless networks, bandwidth is not always sufficient for high speed transmission. Under different wireless environments, the transmission rate can fluctuate a lot. However, HD map is a latency sensitive application. In this case, how to guarantee the HD map updating in a timely manner with communication constrains needs to be considered. To address this, an adaptive resolution scheme can be a solution. When the communication resources are limited or the transmission rate is low, a coarse resolution version of HD map will be transmitted. Since the data size becomes smaller, the latency can be reduced accordingly. When the communication resources are sufficient or the transmission rate is sufficiently high,

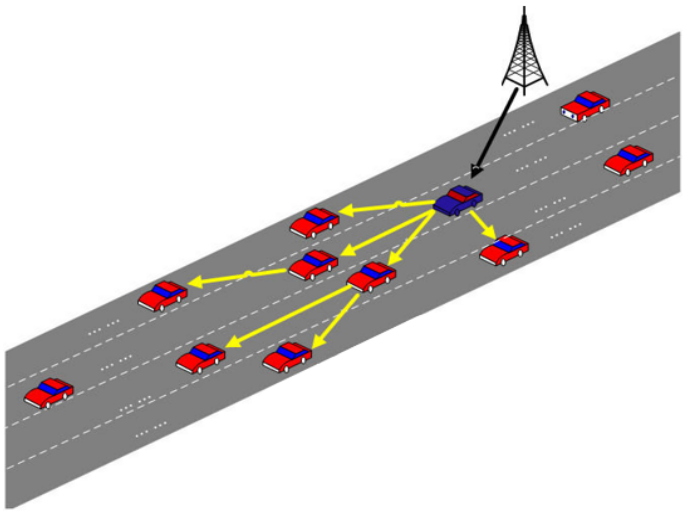


Figure 33: The scenario for HD map distribution.

a detailed resolution version of HD map will be transmitted, which is illustrated in Fig. 32.

- Efficiency or quality of service (QoS) guaranteed HD map distribution scheme: Previously we mainly discuss the HD map updating mechanism, i.e., the uplink transmission. When the information from vehicles located in different areas is gather, the complete HD map needs to be distributed to all the autonomous vehicles for real time and accurate decision making. Therefore, how to deliver the HD map to all the vehicles efficiently needs to be considered. Expect the direct transmission from server to vehicle, the other transmission methods, like vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-everything (V2X) can be take into consideration, which is described in Fig. 33.

References

[1] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *ACM Proceedings of Workshop on Mobile Big Data*, Hangzhou, China, June 2015, pp. 37–42.

[2] C. V. Networking, "Cisco global cloud index: Forecast and methodology, 2014-2019," *White Paper, Cisco*, 2013.

[3] H. Wang, T. Liu, B. Kim, C.-W. Lin, S. Shiraishi, J. Xie, and Z. Han, "Architectural design alternatives based on cloud/edge/fog computing for connected vehicles," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2349–2377, 2020.

[4] Y. Zhang and Z. Han, "Multi-dimensional payment plan in fog computing with moral hazard," in *Contract Theory for Wireless Networks*. Springer, February 2017, pp. 73–88.

[5] H. Wang and J. Xie, "User preference based energy-aware mobile ar system with edge computing," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2020, pp. 1379–1388.

[6] H. Zhang, Y. Zhang, Y. Gu, D. Niyato, and Z. Han, "A hierarchical game framework for resource management in fog computing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 52–57, August 2017.

[7] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier iot fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, October 2017.

[8] H. Wang, J. Xie, and T. Han, "A smart service rebuilding scheme across cloudlets via mobile ar frame feature mapping," in *Proc. IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[9] H. Wang, B. Kim, J. Xie, and Z. Han, "Energy drain of the object detection processing pipeline for mobile devices: Analysis and implications," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 1, pp. 41–60, 2020.

[10] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: a green computing paradigm to support iot applications," *Iet Networks*, vol. 5, no. 2, pp. 23–29, March 2016.

[11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, June 2016.

[12] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2516–2529, February 2015.

[13] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big data and internet of things: A roadmap for smart environments*, 2014, pp. 169–186.

[14] M. Chen, Y. Zhang, Y. Li, S. Mao, and V. C. Leung, "Emc: Emotion-aware mobile cloud computing in 5g," *IEEE Network*, vol. 29, no. 2, pp. 32–38, March 2015.

[15] National Highway Traffic Safety Administration (NHTSA). Automated vehicles for safety. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety#issue-road-self-driving.

[16] Intel. Data is the new oil in the future of automated driving. [Online]. Available: https://newsroom.intel.com/editorials/krzanich-the-future-of-automated-driving/#gs.jpzzzs.

[17] D. Chen, X. Zhang, L. L. Wang, and Z. Han, "Prediction of cloud resources demand based on hierarchical pythagorean fuzzy deep neural network," *IEEE Transactions on Services Computing*, to appear 2019.

[18] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Transactions on Mobile Computing*, to appear 2019.

[19] B. Antonio, F. Stefano, and I. Ahmad, "Deploying fog applications how much does it cost, by the way?" in *the 7th International Conference on Cloud Computing and Services Science (CLOSER)*, Porto, Portugal, April 2017.

[20] Amazon Web Services, "AWS Pricing - How does AWS pricing work?" [Online]. Available: https://aws.amazon.com/pricing/?nc1=h_ls.

[21] J. Xu, R. Rahmatizadeh, L. Bölöni, and D. Turgut, "Real-time prediction of taxi demand using recurrent neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2572–2581, August 2018.

[22] C. Lemke, M. Budka, and B. Gabrys, "Metalearning: a survey of trends and technologies," *Artificial intelligence review*, vol. 44, no. 1, pp. 117–130, June 2015.

[23] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11 098–11 112, November 2018.

[24] Q. Fan and N. Ansari, "On cost aware cloudlet placement for mobile edge computing," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 926–937, July 2019.

[25] H. Wang, B. Kim, J. Xie, and Z. Han, "E-auto: A communication scheme for connected vehicles with edge-assisted autonomous driving," in *IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019.

[26] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *European Telecommunications Standards Institute (ETSI) white paper*, vol. 11, no. 11, pp. 1–16, September 2015.

[27] P. Gupta, A. Seetharaman, and J. R. Raj, "The usage and adoption of cloud computing by small and medium businesses," *International Journal of Information Management*, vol. 33, no. 5, pp. 861–874, October 2013.

[28] J. M. García, P. Fernández, A. Ruiz-Cortes, S. Dustdar, and M. Toro, "Edge and cloud pricing for the sharing economy," *IEEE Internet Computing*, vol. 21, no. 2, pp. 78–84, March 2017.

[29] A. Mazrekaj, I. Shabani, and B. Sejdiu, "Pricing schemes in cloud computing: an overview," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, pp. 80–86, February 2016.

[30] B. Li, W. Xie, W. Zeng, and W. Liu, "Learning to update for object tracking with recurrent meta-learner," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3624–3635, February 2019.

[31] S. Canuto, D. X. Sousa, M. A. Gonçalves, and T. C. Rosa, "A thorough evaluation of distance-based meta-features for automated text classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2242–2256, March

2018.

[32] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *the 32th AAAI Conference on Artificial Intelligence*, New Orleans, LA, 2018.

[33] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *the 31th Conference on Neural Information Processing Systems*, Long Beach, CA, 2017.

[34] J. R. Rice, "The algorithm selection problem," in *Advances in computers*. Elsevier, 1976, vol. 15, pp. 65–118.

[35] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[36] D. G. Ferrari and L. N. De Castro, "Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods," *Information Sciences*, vol. 301, pp. 181–194, April 2015.

[37] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint arXiv:1810.03548*, 2018.

[38] M. Tripathy and A. Panda, "A study of algorithm selection in data mining using meta-learning." *Journal of Engineering Science & Technology Review*, vol. 10, no. 2, March 2017.

[39] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," *arXiv preprint arXiv:1801.01078*, 2017.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, November 1997.

[41] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, Atlanta, GA, June 2013, pp. 1310–1318.

[42] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, July 2005.

[43] Z. Cui, R. Ke, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," *arXiv preprint arXiv:1801.02143*, 2018.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, June 2014.

[46] Unity. Unity for games: create a world with more play. [Online]. Available: https://unity.com/solutions/game.

[47] J. Mackenzie, J. F. Roddick, and R. Zito, "An evaluation of htm and lstm for short-term arterial traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1847–1857, August 2019.

[48] Transport for London. Traffic modeling guidelines version 3.0. [Online]. Available: http://content.tfl.gov.uk/traffic-modelling-guidelines.pdf.

[49] T. D. Wemegah, S. Zhu, and C. Atombo, "Modeling the effect of days and road type on peak period travels using structural equation modeling and big data from radio frequency identification for private cars and taxis," *European Transport Research Review*, vol. 10, no. 2, p. 39, June 2018.

[50] R. Liu, J. Wang, and B. Zhang, "High definition map for automated driving: Overview and analysis," *The Journal of Navigation*, vol. 73, no. 2, pp. 324–341, March 2020.

[51] D. Chen, D. Wang, Y. Zhu, and Z. Han, "Digital twin for federated analytics using a bayesian approach," *to be pubulished in IEEE Internet of Things Journal, doi:10.1109/JIOT.2021.3098692*, 2021.

[52] F. Jomrich, J. Schmid, S. Knapp, A. Höß, R. Steinmetz, and B. Schuller, "Analysing communication requirements for crowd sourced backend generation of hd maps used in automated driving," in *IEEE Vehicular Networking Conference (VNC)*, Taipei, Taiwan, China, December 2018.

[53] S. Kokovin, J.-F. Thisse, and E. Zhelobodko, "Monopolistic competition: Beyond the ces," *CEPR Discussion Paper No. DP7947*, August 2010.

[54] X. Xu, S. Gao, and M. Tao, "Distributed online caching for high-definition maps in autonomous driving systems," *IEEE Wireless Communications Letters*, vol. 10, no. 7, pp. 1390–1394, July 2021.

[55] F. Zhang, J. Zhai, X. Shen, O. Mutlu, and X. Du, "Poclib: A high-performance framework for enabling near orthogonal processing on compression," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 2, pp. 459–475, June 2021.

[56] D. Chen, C. S. Hong, L. Wang, Y. Zha, Y. Zhang, X. Liu, and Z. Han, "Matching-theory-based low-latency scheme for multitask federated learning in mec networks," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 415–11 426, July 2021.

[57] G. Lloret-Talavera, M. Jorda, H. Servat, F. Boemer, C. Chauhan, S. Tomishima, N. N. Shah, and A. J. Pena, "Enabling homomorphically encrypted inference for large dnn models," *IEEE Transactions on Computers*, Early access, 2021.

[58] H. Chu, L. Guo, B. Gao, H. Chen, N. Bian, and J. Zhou, "Predictive cruise control using high-definition map and real vehicle implementation," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 11 377–11 389, September 2018.

[59] M. Hirabayashi, A. Sujiwo, A. Monrroy, S. Kato, and M. Edahiro, "Traffic light recognition using high-definition map features," *Robotics and Autonomous Systems*, vol. 111, pp. 62–72, January 2019.

[60] Z. Jian, S. Zhang, S. Chen, X. Lv, and N. Zheng, "High-definition map combined local motion planning and obstacle avoidance for autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV)*, Paris, France, June 2019, pp. 2180–2186.

[61] Y. Yoon, H. Chae, and K. Yi, "High-definition map based motion planning, and control for urban autonomous driving," *SAE Technical Paper*, 2021.

[62] F. Wang, D. Guan, L. Zhao, and K. Zheng, "Cooperative v2x for high definition map transmission based on vehicle mobility," in *IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, Kuala Lumpur, Malaysia, April 2019.

[63] X. Xu, S. Gao, and M. Tao, "Distributed online caching for high-definition maps in autonomous driving systems," *IEEE Wireless Communications Letters*, vol. 10, no. 7, pp. 1390–1394, July 2021.

[64] Y. Wu, Y. Shi, Z. Li, and S. Chen, "A cluster-based data offloading strategy for high definition map application," in *IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, May 2020.

[65] Y. Liu, M. Song, and Y. Guo, "An incremental fusing method for high-definition map updating," in *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Bari, Italy, October 2019, pp. 4251–4256.

[66] J. Xie, J. Tang, and S. Liu, "An energy-efficient high definition map data distribution mechanism for autonomous driving," *arXiv preprint arXiv:2010.05233*, 2020.

[67] J. Jiao, "Machine learning assisted high-definition map creation," in *IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, Yokyo, Japan, July 2018, pp. 367–373.

[68] C. Kim, S. Cho, M. Sunwoo, P. Resende, B. Bradaï, and K. Jo, "Updating point cloud layer of high definition (hd) map based on crowd-sourcing of multiple vehicles installed lidar," *IEEE Access*, vol. 9, pp. 8028–8046, 2021.

[69] H. Zhuang, C. Wang, Y. Qian, and M. Yang, "Semantic-based road segmentation for high-definition map construction," in *International Conference on Cognitive Systems and Signal Processing*, Zhuhai, China, December 2020, pp. 527–537.

[70] Y. Li, C. Wang, F. Chen, Z. Su, and X. Wu, "A method for generating large-scale high definition color-point map," in *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, Irkutsk, Russia, August 2019, pp. 487–492.

[71] Automotive Edge Computing Consortium (AECC) White Paper. (2020, May) Operational behavior of a high definition map application. [Online]. Available: https://aecc.org/wp-content/uploads/2020/07/Operational_Behavior_of_a_High_Definition_Map_Application.pdf

[72] Y. B. Can, A. Liniger, O. Unal, D. Paudel, and L. Van Gool, "Understanding bird's-eye view semantic hd-maps using an onboard monocular camera," *arXiv preprint arXiv:2012.03040*, 2020.

[73] W. Feng, J.-M. Friedt, G. Cherniak, and M. Sato, "Batch compressive sensing for passive radar range-doppler map generation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 6, pp. 3090–3102, February 2019.

[74] F. Ghallabi, G. El-Haj-Shhade, M.-A. Mittet, and F. Nashashibi, "Lidar-based road signs detection for vehicle localization in an hd map," in *IEEE Intelligent Vehicles Symposium (IV)*, Paris, France, June 2019, pp. 1484–1490.

[75] Y. Zhao, H. Wang, H. Su, L. Zhang, R. Zhang, D. Wang, and K. Xu, "Understand love of variety in wireless data market under sponsored data plans," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 4, pp. 766–781, February 2020.

[76] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, January 2020.

[77] C. Ma, J. Konečnỳ, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takáč, "Distributed optimization with arbitrary local solvers," *Optimization Methods and Software*, vol. 32, no. 4, pp. 813–848, February 2017.

[78] R. Han, M. Si, J. Demmel, and Y. You, "Dynamic scaling for low-precision learning," in *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Virtual Event Republic of Korea, February 2021, pp. 480–482.

[79] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in

IEEE Conference on Computer Communications (INFOCOM), Paris, France, April 2019, pp. 1387–1395.

[80] S. Zhang, H. Zhang, Z. Han, H. V. Poor, and L. Song, "Age of information in a cellular internet of uavs: Sensing and communication trade-off design," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6578–6592, October 2020.

[81] F. Wu, H. Zhang, J. Wu, Z. Han, H. V. Poor, and L. Song, "Uav-to-device underlay communications: Age of information minimization by multi-agent deep reinforcement learning," *IEEE Transactions on Communications*, vol. 69, no. 7, pp. 4461–4475, July 2021.

[82] Z. Chen, D. Kuhn, and W. Wiesemann, "Data-driven chance constrained programs over wasserstein balls," *arXiv preprint arXiv:1809.00210*, 2018.

[83] F. Luo and S. Mehrotra, "Decomposition algorithm for distributionally robust optimization using wasserstein metric with an application to a class of regression models," *European Journal of Operational Research*, vol. 278, no. 1, pp. 20–35, October 2019.

[84] J. Blanchet and K. Murthy, "Quantifying distributional model risk via optimal transport," *Mathematics of Operations Research*, vol. 44, no. 2, pp. 565–600, May 2019.

[85] W. Xie and S. Ahmed, "Distributionally robust chance constrained optimal power flow with renewables: A conic reformulation," *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 1860–1867, July 2017.

[86] W. Xie, "On distributionally robust chance constrained programs with wasserstein distance," *Mathematical Programming*, vol. 186, pp. 115–155, November 2019.

[87] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world from edge to core," *IDC White Paper*, 2018.

[88] J. Ren, G. Yu, and G. Ding, "Accelerating dnn training in wireless federated edge learning system," *arXiv preprint arXiv:1905.09712*, 2019.

[89] Y. Zhang and Q. Yang, "A survey on multi-task learning," *arXiv preprint arXiv:1707.08114v2*, 2018.

[90] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, October 2019.

[91] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[92] D. Chen, Y.-C. Liu, B. Kim, J. Xie, C. S. Hong, and Z. Han, "Edge computing resources reservation in vehicular networks: A meta-learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5634–5646, March 2020.

[93] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, January 2019.

[94] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, Dallas, TX, October 2017.

[95] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[96] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, March 2019.

[97] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, Early access, 2019.

[98] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," in *IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, Signapore, August 2019.

[99] C. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *arXiv preprint arXiv:1910.13067*, March 2020.

[100] W. Y. B. Lim, J. Huang, Z. Xiong, J. Kang, D. Niyato, X.-S. Hua, C. Leung, and C. Miao, "Towards federated learning in uav-enabled internet of vehicles: A multi-dimensional contract-matching approach," *arXiv preprint arXiv:2004.03877*, April 2020.

[101] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1310–1322, June 2019.

[102] D. Manlove, *Algorithmics of matching under preferences*. World Scientific, 2013, vol. 2.

[103] C. Su, F. Ye, L.-C. Wang, L. Wang, Y. Tian, and Z. Han, "Uav-assisted wireless charging for energy-constrained iot devices using dynamic matching," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4789–4800, June 2020.

[104] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in iot fog computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7475–7484, August 2018.

[105] N. Sharghivand, F. Derakhshan, L. Mashayekhy, and L. M. Khanli, "An edge computing matching framework with guaranteed quality of service," *IEEE Transactions on Cloud Computing*, Early access, 2020.

[106] S. Seng, C. Luo, X. Li, H. Zhang, and H. Ji, "User matching on blockchain for computation offloading in ultra-dense wireless networks," *IEEE Transactions on Network Science and Engineering*, Early access, 2020.

[107] R. Fantacci and B. Picano, "A matching game with discard policy for virtual machines placement in hybrid cloud-edge architecture for industrial iot systems," *IEEE Transactions on Industrial Informatics*, Early access, 2020.

[108] X. Huang, R. Yu, S. Xie, and Y. Zhang, "Task-container matching game for computation offloading in vehicular edge computing and networks," *IEEE Transactions on Intelligent Transportation Systems*, Early access, 2020.

[109] L. U. Khan, M. Alsenwi, Z. Han, and C. S. Hong, "Self organizing federated learning over wireless networks: A socially aware clustering approach," in *2020 International Conference on Information Networking (ICOIN)*, Barcelona, Spain, January 2020.

[110] N. H. Tran, W. Bao, A. Zomaya, N. M. N.H, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE Conference on Computer Communications (INFOCOM)*, Paris, France, April 2019.

[111] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Transactions on Neural Networks and Learning Systems*, no. 10, October 2020.

[112] D. Chen, L. J. Xie, B. Kim, L. Wang, C. S. Hong, L.-C. Wang, and Z. Han, "Federated learning based mobile edge computing for augmented reality applications," in *International Conference on Computing, Networking and Communications (ICNC)*, Big Island, HI, Feburary 2020.

[113] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly*, vol. 120, no. 5, pp. 386–391, 2013.

[114] N. Raveendran, Y. Gu, C. Jiang, N. H. Tran, M. Pan, L. Song, and Z. Han, "Cyclic three-sided matching game inspired wireless network virtualization," *IEEE Transactions on Mobile Computing*, Early access, 2019.

[115] G. M. Sotomayor, "Ms. machiavelli and the stable matching problem," *American Mathematical Monthly*, vol. 92, no. 4, pp. 261–268, 1985.

[116] D. Gale and M. Sotomayor, "Some remarks on the stable matching problem," *Discrete Applied Mathematics*, vol. 11, no. 3, pp. 223–232, July 1985.

[117] I. Gent and P. Prosser, "An empirical study of the stable marriage problem with ties and incomplete lists," in *the 15th European Conference on Artificial Intelligence*, Amsterdam, Netherlands, August 2002, pp. 141–145.

[118] K. Iwama and S. Miyazaki, "A survey of the stable marriage problem and its variants," in *International conference on informatics education and research for knowledge-circulating society (ICKS 2008)*, Tokyo, Japan, March 2008, pp. 131–136.