8-30-2022

Edge Assignment and Data Valuation in Federated Learning

Thuy T. Do
University of Massachusetts Boston

EDGE ASSIGNMENT AND DATA VALUATION IN FEDERATED LEARNING

A Dissertation Presented

by

Thuy T. Do

Submitted to the Office of Graduate Studies, University of Massachusetts Boston, in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2022

Computer Science Program

EDGE ASSIGNMENT AND DATA VALUATION IN FEDERATED LEARNING

A Dissertation Presented

by

Thuy T. Do

Approved as to style and content by:

Duc A. Tran, Associate Professor
Chairperson of Committee

Marc Pomplun, Professor
Member

Bo Sheng, Associate Professor
Member

Kenneth Fletcher, Assistant Professor
Member

Ping Chen, Associate Professor
Member

Dan Simovici, Program Director
Computer Science Program

Marc Pomplun, Chairperson
Computer Science Department

ABSTRACT

EDGE ASSIGNMENT AND DATA VALUATION IN FEDERATED LEARNING

August 2022

Thuy T. Do,
B.S. and M.S. , Viet Nam National Univerisity, Ha Noi, Viet Nam.
Ph.D., University of Massachusetts Boston

Directed by Associate Professor Duc A. Tran

Federated Learning (FL) is a recent Machine Learning method for training with private data separately stored in local machines without gathering them into one place for central learning. It was born to address the following challenges when applying Machine Learning in practice: (1) Communication cost: Most real-world data that can be useful for training are locally collected; to bring them all to one place for central learning can be expensive, especially in real-time learning applications when time is of essence, for example, predicting the next word when texting on a smartphone; and (2) Privacy protection: Many applications must protect data privacy, such as those in the healthcare field; the private data can only be seen by its local owner and as such the learning may only use a content-hiding representation of this data, which is much less informative.

To fulfill FL's promise, this dissertation addresses three important problems regarding the need for good training data, system scalability, and uncertainty robustness:

1. The effectiveness of FL depends critically on the quality of the local training data. We should not only incentivize participants who have good training data, but also minimize the effect of bad training data on the overall learning procedure. The first problem of

my research is to determine a score to value a participant's contribution. My approach is to compute such a score based on Shapley Value (SV), a concept of cooperative game theory for profit allocation in a coalition game. In this direction, the main challenge is due to the exponential time complexity of the SV computation, which is further complicated by the iterative manner of the FL learning algorithm. I propose a fast and effective valuation method that overcomes this challenge.

2. On scalability, FL depends on a central server for repeated aggregation of local training models, which is prone to become a performance bottleneck. A reasonable approach is to combine FL with Edge Computing: introduce a layer of edge servers to each serve as a regional aggregator to offload the main server. The scalability is thus improved, however at the cost of learning accuracy. The second problem of my research is to optimize this tradeoff. This dissertation shows that this cost can be alleviated with a proper choice of edge server assignment: which edge servers should aggregate the training models from which local machines. Specifically, I propose an assignment solution which is especially useful for the case of non-IID training data which is well-known to hinder today's FL performance.

3. FL participants may decide on their own what devices they run on, their computing capabilities, and how often they communicate the training model with the aggregation server. The workloads incurred by them are therefore time-varying, unpredictably. The server capacities are finite and can vary too. The third problem of my research is to compute an edge server assignment that is robust to such dynamics and uncertainties. I propose a stochastic approach to solving this problem.

ACKNOWLEDGMENTS

My PhD journey would have never been possible without the help and sacrifice of many amazing individuals. I would like to thank my advisor, Prof. Duc A. Tran, for his continuous support and patience during my PhD study. I want to thank him for always giving me his best advice. What I learned from him goes beyond doing proper research.

I want to thank my great professors in the Department of Computer Science at UMass Boston. I was inspired to study machine learning from Prof. Dan Simovici's classes and from working on the project in Prof. Kenneth Fletcher's class. I also learned a lot from Prof. Marc Pomplun's teaching in Theory of Computation.

I would like to show my deep appreciation to Lewis P. Robbins and Christopher Hull at UMass Medical School and Dr. Mai T. Tran at Viet Nam National University, Ha Noi for providing me the GPU resources to run my machine learning algorithms.

I want to thank many of my friends, my labmates and my colleagues at UMass Boston and University of Transport and Communications, Ha Noi for a cherished time spent together in the lab and outside of it.

Finally, I would like to express my greatest gratitude to my family. My parents, Son N. Do and Xuan T. Nguyen, who always have me in their thoughts; my sisters and brother, Hai T. Do, Huong T. Do and Hung M. Do, who unconditionally support me whenever I need them; most importantly, my husband and my children, Phuong M. Doan, Diep N. Doan and Phuong N. Doan, who tremendously understand and take care of me. I would have never reached this place without them.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1

INTRODUCTION

Centralized computing relies on a central server to serve clients' requests. The central server is an entity responsible for all computation in this system. Centralized computing, therefore, is a single point of failure and cannot scale. In contrast, decentralized computing refers to having multiple independent computing nodes to provide a service without reliance on a central authority. A node's failure has only a local impact and does not stop the whole system from functioning. Decentralized computing is suitable for big organizations and a trend in modern-day business environments. Edge computing, Federated Computing, and Blockchain are good examples for the shifting from centralized to decentralized computing.

This dissertation is focused on solving some optimization problems in decentralized computing environments, in particular with Edge Computing and Federated Learning. These two computing schemes will be discussed in the background section.

1.1 Background

Edge Computing (EC) is a distributed computing model that brings computation, data storage, and power closer to the source of data by deploying edge servers nearby edge devices. Thanks to the exponential growth of internet-connected devices (e.g., IoT) and new applications that require real-time computing power, EC was introduced and de-

veloped. In the EC architecture, requests from clients are primarily handled by edge servers; requests which cannot be processed at the edge servers will be passed to the cloud. Therefore, EC reduces the amount of data transmitted and stored in the cloud, and also reduces the delay in data transmission and processing. This advantage and faster networking technologies (e.g., 5G wireless) accelerate the creation and support real-time applications, including self-driving cars, artificial intelligence, robotics and video processing. According to [eca], the worldwide edge computing market will reach 274 billion U.S. dollars by 2025. EC is also extremely attractive to the research community. The number of articles related to edge computing on Google Scholar increases dramatically in the last decade: in 2010 there were only 240 articles related to edge computing on Google Scholar, whereas that number is 25,000 in 2020 [ecb].

Federated Learning is to train a machine learning algorithm in a decentralized data fashion. We will see why it is a need. A lot of data is born at the edge (not at the core) of the Internet. There were more than 10 billion active IoT devices in 2021, and the amount of data generated by IoT devices is expected to reach 73.1 ZB (zettabytes) by 2025 [Jov]. That data would be a source for data-hungry machine learning algorithms to enable better products and smarter models. However, standard machine learning (ML), requiring centralizing the training data on a server to train a model, leads to challenges in bringing ML to practice. Transmitting data is costly, especially in real-time learning applications when time is of essence, for example, predicting the next word when texting on a smartphone. Another important issue is that data can be exposed to leakage on the way. According to the ”Cost of a Data Breach Report 2021” by IBM [IBM21], the average total cost of a data breach increased by nearly 10% year over year, reaching up to $4.24 million in 2021. The average cost was $1.07 million higher in breaches where remote work was a factor in causing the breach, compared to those where remote work was not a factor. Sensitive sectors tend to experience the highest average cost of

a data breach, healthcare organizations being at the top for the eleventh year in a row. Due to data compromise, more and more strict regulations to protect data privacy are established, for example, General Data Protection Regulation implemented in 2018 in Europe. We therefore need a new machine learning mechanism which protects data privacy and works on decentralized data settings. In other words, the new mechanism should not directly access raw local data and should not require moving local data in any form to a central server for performing learning algorithms. Federated learning introduced by Google in 2017 is such a technique.

The thesis addresses three important problems in Federated Learning environment regarding the need for good training data, system scalability, and uncertainty robustness.

1.2 Practical Shapley Value for Data Valuation in Federated Learning

In the first problem, I am motivated by the following question, "How can we incentivize edge devices to provide good data for FL ecosystems?". Clearly, edge devices are willing to join the federation if their data contributions are fairly valuated. This means that we need (1) a fair metric and (2) an efficient algorithm to compute that metric for quantifying data providers. As ML is context-specific, an useful way to compute the importance of a datum should take account of the quality of that datum itself, the machine learning algorithm, the performance metric of the algorithm, and the rest of the training data. In this work, I first go over existing methods to compute the value of each datum in standard machine learning settings. I then examine approaches used to quantify clients' data contributions in federated learning scheme. Among those approaches, Shapley value [rot88], a concept in game theory to fairly distribute the profit of the game to all players, is the main focus of the research community. In Shapley value based approach,

FL is considered a game where each FL participant is a player and the performance of the model trained on the set of participants' data is the gain of the game. However, computing SVs is expensive. I propose an algorithm to compute it efficiently. My approach is to apply multi-issue decomposition to partition the grand game into subgames so that computing SV is less time-consuming. I have proven that in the worst case the time complexity of my algorithm is the same as that of the literature. However, in practice with a reasonable choice of parameters my algorithm is much faster. This work is in the preparation for submission.

1.3 Edge Assignment in Federated Learning

In the second problem, I focus on the challenge of scalability in FL. FL relies on a central server to aggregate clients' local models for a global one. Many local updates simultaneously at the central server can cause the communication bottleneck problem. Edge federated learning is introduced to overcome this challenge. That is, we deploy edge servers closer to clients to aggregate local models for regional ones; the central server aggregates regional models, which are not abundant, to form the global one. The scalability thus is no longer the problem, but the cost of learning accuracy. That cost is the consequence of adding edge servers. We will see that the learning accuracy can decrease due to the divergence of regional models. In this chapter, I aim to maximize the learning accuracy, using edge server assignment: which edge servers should aggregate the training models from which local machines.

My approach is to maximize the diversity of local data aggregated through local models at edge servers. For example, in an eFL setting with 4 participants and 2 edge servers such that participant 1's and participant 2's training data consist of mostly label A and participant 3's and participant 4's mostly label B, participants 1, 3 should be

combined on one edge server and participants 2, 4 on the other edge server. This way, each edge server has a comprehensive coverage of training data (having both labels A and B). I propose an algorithm to compute the edge assignment to maximize the global model's learning accuracy without increasing communication cost. To my knowledge, this work is the first effort on the edge assignment problem aimed to maximize learning accuracy that can work for every eFL setting. This work has been submitted to IEEE transaction on Emerging Topics in Computing in March 2022 (under review).

1.4 A Stochastic Geo-Partitioning Problem for Mobile Edge Computing

In the final problem, I consider the edge assignment problem in the new context where the workloads created by clients change over time. For instance, mobile devices only join the federation when they are idle, charging, and on an unmetered network. The capacities of servers are finite and also time-varying. We want an assignment maximizing the utility of edge computing that is robust to these uncertainties, otherwise we have to recompute the assignment whenever changes happen. Note that computing the assignment is expensive as it is NP-hard. We also want each cluster of client to form a geographically compact coverage. In this work, I introduce a new partitioning problem: Stochastic Geographically-aware Partitioning (StoGeoPar). Its objective is to optimize a stochastic quantity formulated to incorporate all the three criteria of cost-efficiency, geo-compactness, and uncertainty-robustness. I also propose an approximate algorithm for StoGeoPar. My approach is to optimize an approximative version of its original stochastic objective by fitting the time-varying inputs with Gaussian distributions and applying a block-relaxation optimization method to solve it. Major parts of this work has been published in IEEE transaction on Emerging Topics in Computing [TDZ21].

1.5 Dissertation Organization

The remainder of the dissertation is organized as follows. The above three research problems will be presented in Chapter 2, Chapter 3, and Chapter 4, respectively. After Chapter 2, the reader can read Chapter 3 or Chapter 4 in any order. Chapter 5 concludes the dissertation with pointers to my future work.

CHAPTER 2

PRACTICAL SHAPLEY VALUE FOR DATA VALUATION IN FEDERATED LEARNING

Federated Learning (FL) is an emerging Machine Learning method for training with private data separately stored in local machines without gathering them into one place as in central learning. Despite its promise, the effectiveness of FL depends critically on the quality of the local training data. We should not only incentivize participants who have good training data, but also minimize the effect of bad training data on the overall learning procedure. It is therefore important to have a score quantifying how valuable a participant's contribution is to the system. Our approach is to compute such a score based on Shapley Value, a concept of cooperative game theory used to determine the rewards given to players of a coalition game. In this direction, the main challenge is due to the exponential time complexity of the Shapley Value computation, which is further complicated by the iterative manner of the FL learning algorithm. We propose a fast and effective valuation method that overcomes this challenge. Its performance is substantiated with an evaluation study using both real-world and synthetic data.

2.1 Introduction

Federated Learning (FL) is a Machine Learning approach recently born [MMR17] to address the following challenges when applying Machine Learning in practice: (1)

Communication cost: Most real-world data that can be useful for training are locally collected; to bring them all to one place for central learning can be expensive, especially in real-time learning applications when time is of essence, for example, predicting the next word when texting on a smartphone; and (2) Privacy protection: Many applications must protect data privacy, such as those in the healthcare and consumer markets; the private data can only be seen by its local owner and as such the learning may only use a content-hiding representation of this data, which is much less informative.

FL can compute a learning model using distributed training data that remain private and unmoved on local machines, hereafter referred to as the "participants". A conventional FL architecture consists of only the participants (owning private local training data) and a central server (called the "parameter" server). The learning is an iterative procedure as follows. In the first step, the parameter server broadcasts a global learning model, initially random, to all the participants. In the second step, each participant in parallel uses its own local data to improve this model. In the third step, the participants send their respective models to the server who in turn aggregates them to obtain a new global model. Then the first step repeats until the global model converges.

The idea of FL is simple, yet proven effective to some useful extents [KMA21, BEG19, GBM21]. In practice, more than 90% of global data is stored and processed locally [ZWB21]. Furthermore, it fits nicely in today's trend of computing toward decentralization, by crowdsourcing to local machines' computing capabilities for the training process. Having said that, to realize FL's full promise, there is much room left to explore. As FL relies on local participants' help with the training, the learning accuracy depends crucially on the quality of the training data and the cooperation of the participants. Some participants may contribute more and others less. Participants may also intentionally harm the FL system by providing bad local training model information. We should differentiate the good participants from the bad, the better from the good, so

that we can incentivize good participations and minimize the effect of the bad. It is thus critical to have a mechanism to valuate each participant's contribution fairly. This is the focus of this chapter.

The value of a participant should not solely be based on the standalone quality of its training data. Instead, an equitable valuation should see how useful it is toward the whole FL system as a coalition, i.e., how its removal from the coalition degrades the overall performance. For example, to learn if an image is of an "apple" or "orange" and the training data collection already has many "apple" samples, a participant A with few "orange" samples is more valuable than yet another participant B with many "apple" samples. These "orange" samples are needed more for training, even though A contributes more samples to the training collection set than B does.

Data valuation is a fundamental problem in machine learning in general [GZ19]. Instead of finding a way to assign an importance value to each datum that would work universally, a more useful valuation should be context-specific. The value should depend on the learning algorithm, the performance metric, and as aforementioned, the rest of the training data. A recent development is [GZ19], who proposed Data Shapley, an equitable valuation framework based on Shapley Value (SV) [Sha16]. SV is a classic concept for profit allocation in Cooperative Game Theory. By casting the machine learning task as a coalition game whose players are different sources of training data, the SV of each data source, which determines the profit allocated to this source, can be a fair representation of its contribution to the overall learning accuracy.

SV is widely influential in economics [rot88]; Lloyd Shapley received a Nobel Prize in Economics for it in 2012. It is the only payoff scheme for a cooperative game that satisfies the following properties of equitable valuation: 1) Group rationality: the total profit is fully distributed among the players; 2) Fairness: Players that add the same utility gain to any subset of coalition receive the same profit, and this profit is zero if

the performance effect is none; and 3) Additivity: when applied to multiple games, the profit given to each player equals the sum of the profits he or she receives respectively from these games.

As SV is that appealing, one would attempt to apply SV to data valuation in FL, as investigated in the works of [STW19, WRZ20, FFZ21, NN21, FFZ22, LCY21, WDZ19]. However, the practicality of this application remains questionable due to two main reasons. The first is inherited from the well-known hardness of SV computation; to compute the SV for a data contributor requires an exponential number of times to evaluate the utility function. The second reason is due to the iterative manner of the FL algorithm; its learning accuracy, which would serve as the utility for SV, can only be measured after many rounds of computation converges. We will elaborate on this later, but put simply, combining both reasons, it is impossible to directly apply SV to FL.

We propose a practical data valuation metric for FL that can be considered an approximative version of SV. Specifically, our contributions are below:

- If we have all the local data of FL gathered at one place for central training, we can compute SV for each data source. However, we show that this SV, if available, should not be used as benchmark for the FL setting, where the computation must go through many iterations, each training on a different subset of local data. This is particularly true when the training data is non-independent and identically distributed (non-iid) among the participants. This is a real-world phenomenon specific only to FL as compared to central learning [ZXL21, WKN20, SWM20].

- It would be ideal if one could customize a SV method for FL. However, not only it is impossible to apply it directly, existing efforts proposing modified versions of it to work with FL remain much theoretical. The main motivation for our proposed SV-based metric is to make it practical. It is faster than the existing SV-based

methods while still being consistent with their value distribution. Not only that, our distribution is even smoother with less deviation. This is desirable when we use these values as basis to incentivize good or penalize bad participations.

- As FL is one of the more recent approaches to Machine Learning, the research on data valuation has only just begun [LCY21, YLL20c, ZZW20, ZLR20, ZLC21, PTB20, LYN20, WDZ19, NSM20, STW19, WRZ20, FFZ21, NN21, FFZ22, XNZ21]. Some have brought the SV concept to FL [STW19, WRZ20, FFZ21, LCY21, NN21, FFZ22, WDZ19]. Our research is a timely addition to this relatively young literature. The proposed method can serve as a good benchmark for future developments.

The remainder of the chapter is organized as follows. Related work is discussed in Section 2.2. The necessary background on the FL data valuation problem is presented in Section 2.3. The proposed valuation method is proposed in Section 2.4. The results of the evaluation study are analyzed in Section 2.5. We conclude the chapter in Section 2.6.

2.2 Related works

In large-scale data analysis using Machine Learning, there is an especially strong need to prioritize samples. A straightforward approach is to measure the effect of each sample on the accuracy if we remove it from the training set. The accuracy gain (or loss) is the importance score. For a set of n samples, this so-called Leave-One-Out (LOO) approach requires re-training the dataset n times, each time with a sample omitted; hence, very expensive. A more scalable valuation method [WCZ16] proposes using an influence function called Cook's Distance [CW80], a classical technique in robust statistics

as a measure for sample influence, and applies random sampling to avoid re-training. This work is for linear models, where influence functions are well-defined thanks to the underlying loss function's convexity and are generally accurate even for estimating group influences [KAT19]. However, for deep learning with non-convex loss functions, influence functions are not well-understood, as experimentally pointed out in [BPF21]. Another approach is to learn a direct mapping between a training sample and its importance value [YAP20]. The idea is to train this mapping using Reinforcement Learning applied to a small validation set that reflects the target learning task. This method can compute the value of a data sample quickly once the mapping is learned. On the other hand, the reinforcement training is required a priori and the mapping accuracy depends on the quality of this training.

The above approaches do not satisfy the properties of an equitable valuation. For example, if the training set contains two identical samples, they will be given zero values because removing one does not change the training accuracy. This is unfair. Recently, Shapley value (SV) [Sha16] was proposed for data valuation in Machine Learning, in the name of Data Shapley [GZ19]. SV is the only scheme that satisfies all the properties of equitable valuation. Data Shapley has been experimented with good results in medical imaging of chest X-ray [TGY21].

The SV of a sample can be seen more or less as the average LOO value of this sample when evaluated on all subsets of training data. As such, SV requires re-training on all these subsets. Since this is exponential in time complexity, approximation is a must. For example, in [GZ19, CGT09], Monte-Carlo sampling is applied to select only a polynomial number of subsets to re-train. To further speed up, the procedure to compute SV can stop when a convergence threshold is reached, e.g., no more significant SV change is detected, without having to exhaust all these subsets. By these methods, it

typically takes $n \log n$ time to reach the convergence of SV approximation. It is shown in [JDW20] that one can reduce this time to $O(\sqrt{n} \log^2 n)$ with some smart sampling.

Data Shapley has been extended in different ways. A generalization of Data Shapley, called Beta Shapley, was proposed in [KZ22] arguing that the uniform averaging from all the subsets giving equal weight to each subset to compute SV is detrimental to capturing the influence of individual data; Beta Shapley provides better weights based on Beta distribution. Another generalization, called Distributional Shapley [GKZ20], extends Data Shapley to the case that the data set is not fixed, but instead follows an underlying data distribution; the (distributional) SV is the expected SV randomized over the data set distribution.

In the context of Federated Learning (FL), we are interested in valuing not individual data samples, but the participants who each own a subset of the overall training data. One measure can be based on the data amount and computing resources provided by the participants [YLL20c, ZZW20, ZLR20]. Another measure considers the resulted learning performance, for example, how similar their local model updates are to the global model [ZLC21, PTB20, LYN20]. LOO and influence functions are used in [YLL20b, BCM18, WDZ19, XNZ21]. In the same line with our research, SV is used in [STW19, WRZ20, FFZ21, LCY21, NN21, FFZ22, WDZ19]. SV is appealing not only because of its equitability properties. It is useful to select good participants to join each update round [NN21] or bad data features to ignore in the local training on the participant side [FFZ22]. To compute the training accuracy, we cannot have all the training data into one place. Instead, only their local models may be communicated. Furthermore, FL must run many rounds of local model updates until convergence to obtain good training. Re-training must be avoided. This, together with the exponential complexity of SV computation makes the precise computation of SV for FL infeasible. The common SV-based approach to date works as follows [STW19]: 1) in each update

round, compute the SV of each participant based on the utility function applied to current local models, and 2) summing these round-based SVs to obtain the final SV for each participant. Further steps can apply to speed up the round-SV computation by approximation (e.g., using Monte Carlo sampling [WRZ20], matrix completion [FFZ21], guided truncation [LCY21]). To improve fairness, for example, in avoiding the case that two participants with the same training data receives are given different values, one can consider the method in [FFZ21]. Compared to the above SV-based methods for FL, our research provides another layer of scaling up the SV computation. Thanks to the additivity property of SV, a game can be decomposed into sub-games so that the overall SV can be efficiently computed from the sub-SVs [CS04]. This idea has been shown [ACC14, CD19] to be faster and more accurate than sampling-based approximation. We apply a similar divide-and-conquer approach. By dividing the participants into different subgroups, we can compute SV for their respective sub-games and take advantage of the addibility property of SV to obtain the final SV for FL. To the best of our knowledge, we are the first to pursue this direction.

2.3 Background and Motivation

Consider a typical supervised learning task of learning a function that maps an input object to an output value (called "label") based on a set of input-output pair samples (called the "training set"). The label here can be a class label in classification learning or a real-valued number in regression learning.

Let X and Y denote the input and output (compact) spaces, respectively. Suppose that we are given a training set of samples, $\mathscr{D} = \{(x_1, y_1), ..., (x_{|\mathscr{D}|}, y_{|\mathscr{D}|})\}$, such that $x_i \in X$ is the feature vector of the i^{th} input sample and $y_i \in Y$ its corresponding label. The learning task is to find a function $g : X \to Y$ so that given each new input object x we will predict that its label is $g(x)$. The formulation of g depends on the underlying learning method in

use (support vector machines, deep neural networks, etc.). We assume that g is uniquely formulated based on a model $\mathbf{w} \in \mathbb{R}^d$ which is a vector of d parameters. Hereafter, we are interested in finding $\mathbf{w}$ (since g can be derived from $\mathbf{w}$).

We quantify the prediction quality by the following empirical loss

$$F(\mathbf{w}; \mathscr{D}) \triangleq \frac{1}{|\mathscr{D}|} \sum_{(x,y)\in\mathscr{D}} l\Big(\mathbf{w}; x, y\Big) \tag{2.1}$$

where $l(\mathbf{w}; x, y)$ is a user-defined function measuring the prediction loss on sample (x, y) using model $\mathbf{w}$.

The goal of the learning task is to find $\mathbf{w}$ given $\mathscr{D}$ to minimize $F(\mathbf{w}; \mathscr{D})$. A typical way to find the optimal $\mathbf{w}$ is by the iterative method of Stochastic Gradient Descent as follows:

Algorithm 1: Stochastic Gradient Descent (SGD)

1. Start with some initial model $\mathbf{w}^{(0)}$
2. For each round t = 1, 2, ..., T:
   - Update $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \nabla_w F(\mathbf{w}^{(t-1)}; \mathscr{D})$
3. Return $\mathbf{w}^{(T)}$

Here, ∇ is the vector differential operator in math. As the number of rounds T is sufficiently large, the value $\mathbf{w}^{(T)}$ at the end of this loop should converge to the optimum $\mathbf{w}$. Parameter η is predefined and called the learning rate. The higher η is chosen, the quicker convergence is reached but there is a higher risk to miss it. On the other hand, if η is too small, the learning can be too slow to converge.

For ease of presentation, we denote this algorithm with $\texttt{SGD}(\mathbf{w}^{(0)}; \mathscr{D})$ where $\mathbf{w}^{(0)}$ is the initial model to begin the loop with and $\mathscr{D}$ the set of training samples.

2.3.1 Federated Learning (FL)

In FL, the training samples are not available all at one place, but instead they reside independently and privately on many local machines (participants). Let $\mathscr{K}$ be the set of K participants and $\mathscr{D} = \mathscr{D}_1 \cup \mathscr{D}_2 \cup ... \cup \mathscr{D}_K$ where $\mathscr{D}_k$ denotes the subset of training samples owned by participant $k \in \mathscr{K}$.

The prediction loss $F(w)$ can then be expressed as follows

$$F(\mathbf{w}; D) = \sum_{k=1}^{K} \frac{|\mathscr{D}_k|}{|\mathscr{D}|} \underbrace{\left(\frac{1}{|\mathscr{D}_k|} \sum_{(x,y)\in\mathscr{D}_k} l(\mathbf{w}; x, y) \right)}_{F(\mathbf{w};\mathscr{D}_k)} \tag{2.2}$$

$$= \sum_{k=1}^{K} \frac{|\mathscr{D}_k|}{|\mathscr{D}|} F(\mathbf{w}; \mathscr{D}_k). \tag{2.3}$$

We can think of $F(\mathbf{w}, \mathscr{D}_k)$ as the local prediction loss of participant k. In the IID case, where the set of training samples is distributed uniformly at random among the participants, we would have

$$\mathbb{E}_{\mathscr{D}_k}[F(\mathbf{w}; \mathscr{D}_k)] = F(\mathbf{w}; \mathscr{D}). \tag{2.4}$$

That is, the expectation of $F(\mathbf{w}; \mathscr{D}_k)$ over an IID-generated $\mathscr{D}_k$ as a subset of $\mathscr{D}$ would equal $F(\mathbf{w}, \mathscr{D})$. What this implies is a simple distributed learning approach: the participants can each independently solve the learning problem using their own training data and the average over all these local models provides a good approximation for the optimal model. This is the foundation for FL.

The earliest and arguably most widely-used FL algorithm is FedAvg [MMR17]. FedAvg uses SGD (presented above) as the learning method. In the simplest form it works as follows:

Algorithm 2: Federated Averaging (SGD)

1. The server starts with an initial model $\mathbf{w}^{(0)}$
2. For each round t = 1, 2, ..., T:
   (a) The server broadcasts model $\mathbf{w}^{(t-1)}$ to all the participants
   (b) In response, each participant k runs SGD on local data to compute/update its local model
   $$\mathbf{w}_k^{(t)} = \texttt{SGD}(\mathbf{w}^{(t-1)}; \mathscr{D}_k)$$
   and sends it to the server. Note that this SGD starts with the global model $\mathbf{w}^{(t-1)}$ just received from the server.
   (c) The server updates the global model by averaging the updated local models, weighted by the size of each local dataset:
   $$\mathbf{w}^{(t)} = \frac{\sum_{k=1}^{K} |\mathscr{D}_k| \mathbf{w}_k^{(t)}}{\sum_{k=1}^{K} |\mathscr{D}_k|}$$
3. Return $\mathbf{w}^{(T)}$

We denote this algorithm with $\texttt{FL}(\mathbf{w}^{(0)}; \{\mathscr{D}_1, \mathscr{D}_2, ..., \mathscr{D}_K\})$, which uses $\mathbf{w}^{(0)}$ as the initial global model and applies to a FL system of K participants with local training datasets $\mathscr{D}_1$, $\mathscr{D}_2$, ..., $\mathscr{D}_K$, respectively.

This algorithm has several variations [MMR17]. For example, instead of broadcasting the global model in each round to all the participants (step 2a of the above algorithm), we choose to send it to only a random subset of them, which will result in lower communication and computation costs. To further reduce the local computation cost of the participants, the SGD procedure can be modified to apply on a mini-batch of training samples in each gradient descent step rather than on the entire training set.

2.3.2 Shapley Value (SV)

Consider a coalition game $(\mathscr{K}, v)$ where $\mathscr{K}$ is the grand coalition of candidate players and v the characteristic function (also called the utility function). Any subset of players $S \subset \mathscr{K}$ can form a coalition to play this game, which results in a payoff value $v(S) \in \mathbb{R}$ for the coalition; in the trivial case, $v(\varnothing) = 0$. Shapley Value (SV) is a method to compute the individual payoff given to each player. Intuitively, player $k \in \mathscr{K}$ makes a marginal contribution $\Big(v(S \cup \{k\}) - v(S)\Big)$ to each subset $S \subset \mathscr{K} \setminus \{k\}$ (excluding k). The average marginal contribution to a coalition size of size $|S| = i$ is

$$a_i = \sum_{S \subset \mathscr{K} \setminus \{k\}, |S| = i} \binom{|\mathscr{K}| - 1}{i}^{-1} \Big(v(S \cup \{k\}) - v(S)\Big).$$

The SV of player k is the average of this marginal contribution over all possible size $i \in \{0, 1, ..., |\mathscr{K}| - 1\}$,

$$\phi(k; \mathscr{K}, v) = \frac{1}{|\mathscr{K}|} \Big(a_0 + a_1 + ... + a_{|\mathscr{K}|-1}\Big)$$

which is equivalent to

$$\phi(k; \mathscr{K}, v) = \frac{1}{|\mathscr{K}|} \sum_{S \subset \mathscr{K} \setminus \{k\}} \Big(v(S \cup \{k\}) - v(S)\Big) \Big/ \binom{|\mathscr{K}| - 1}{|S|} \tag{2.5}$$

Alternatively, we can express the SV of player k as the average marginal contribution it adds to a subset formed by each permutation of the grand coalition as follows,

$$\phi(k; \mathscr{K}, v) = \frac{1}{|\mathscr{K}|!} \sum_{permutation\ P} \Big(v(P_k \cup \{k\}) - v(P_k)\Big), \tag{2.6}$$

where P_k is the set of players that appear before player k in the permutation order P.

SV is the only valuation method having all the following desirable properties:

- *Efficiency*: The payoff value of the grand coalition is fully distributed to the players:

$$\sum_{k \in \mathscr{K}} \phi(k; \mathscr{K}, v) = v(\mathscr{K}).$$

- *Symmetry*: Two players, k_1 and k_2, have the same SV if they make the same marginal contribution to any subset excluding them:

$$v(S\cup\{k_1\}) = v(S\cup\{k_2\})\ \forall S \subset \mathscr{K}\setminus\{k_1,k_2\} \tag{2.7}$$

$$\rightarrow \phi(k_1;\mathscr{K},v) = \phi(k_2;\mathscr{K},v). \tag{2.8}$$

- *Null Player*: A player k has zero SV if making no marginal contribution to every subset excluding it:

$$v(S\cup\{k\}) = 0\ \forall S \subset \mathscr{K}\setminus\{k\} \tag{2.9}$$

$$\rightarrow \phi(k;\mathscr{K},v) = 0. \tag{2.10}$$

- *Linearity*: If a game can be divided into two sub-games such that its characteristic function is a linear combination of the sub-game characteristic functions, SV follows the same linearity:

$$\phi(k;\mathscr{K},a_1v_1+a_2v_2) = a_1\phi(k;\mathscr{K},v_1)+a_2\phi(k;\mathscr{K},v_2).$$

A FL task can be considered a coalition game, where each participant is a player and the payoff function for a participant subset is the learning accuracy of the FL system using this subset as the participants. Specifically, the payoff value $v(S)$ for such a participant subset $S=\{s_1,s_2,...,s_l\}$ which respectively contributes training data $\{\mathscr{D}_{s_1},\mathscr{D}_{s_2},...,\mathscr{D}_{s_l}\}$ is

$$v(S) = \text{accuracy}\left\{\mathbf{w}_S = \texttt{FL}(\mathbf{w}^{(0)};\{\mathscr{D}_{s_1},\mathscr{D}_{s_2},...,\mathscr{D}_{s_l}\})\right\}.$$

Here, $\mathbf{w}_S$ is the global model resulted from applying FL to S. Obviously, to compute SV precisely this way for FL is infeasible, because it requires running FL for an exponential number of times, $O(2^{|\mathscr{K}|})$ for Eq. (2.5) or $O(|\mathscr{K}|!)$ for Eq. (2.6), each time applying

to a subset of participants. The complexity is no better than $O(T2^{|\mathscr{K}|})$, where T is the number of rounds of global update in FL.

Alternatively, to avoid re-running FL, a typical approach is to compute the following modified version of SV; this is referred to as Federated SV. Let T be the number of rounds of global update. The Federated SV of participant k is the sum of the SV of k in each round of the FL process,

$$\phi(k) = \phi(k;\mathscr{K},v_1) + \phi(k;\mathscr{K},v_2) + ... + \phi(k;\mathscr{K},v_T)$$

where in each round $t = 1,2,...,T$ we define a coalition game with the following payoff function for each participant subset $S \subset \mathscr{K}$:

$$v_t(S) = \text{acc}\left\{\mathbf{w}_S^{(t)} = \frac{\sum_{k\in S}\mathbf{w}_k^{(t)}}{|S|}\right\}. \tag{2.11}$$

Intuitively, imagining that FL stops at the end of round t, payoff function $v_t(S)$ represents the worth of subset S if the average model of its participants is used as the global model.

Federated SV can be computed on the fly as we are running FL, or later separately after FL completes if the intermediate per-round local models are saved. No re-running of FL is required, but it has the same computational complexity as the straight application of SV to FL. The time complexity of both is $O(T2^{|\mathscr{K}|})$. To reduce this time, the general approach adopted in earlier works is to apply random sampling to compute SV in each round approximately; i.e., choosing a polynomial number of random subsets to include in the computation of $\phi(k;v_t)$ in each round t).

However, to achieve a low approximation error, sampling-based SV approximation still requires prohibitively many subset samples [ACC14, CD19]. A more practical method is needed.

2.4 Practical Federated Shapley

We propose a new method to compute Federated SV based on multi-issue decomposition (MID), a divide-and-conquer theory for profit distribution in coalition games.

2.4.1 Multi-Issue Decomposition

MID was suggested for SV computation in [CS04]. The most important result is the following theorem.

Theorem 2.4.1 ([CS04])**.** *Consider a coalition game with K candidate players and utility function v. Suppose that v can be decomposed as a sum of M utility functions v_1, v_2, ..., v_M,*

$$\forall S \subset \mathscr{K} : v(S) = \sum_{m=1}^{M} v_m(S),$$

and, or each $m \in [M]$, we are given $C_m \subset \mathscr{K}$ such that v_m "concerns only" C_m. Then

$$\forall k \in \mathscr{K} : \phi(k; \mathscr{K}, v) = \sum_{m=1}^{M} \psi(k; C_m, v_m),$$

where

$$\psi(k; v_m) = \begin{cases} \phi(k; C_m, v_m), & \text{if } k \in C_m \\ 0, & \text{otherwise} \end{cases}$$

Here, the relation "concerns only" means the following.

Definition A utility function u "concerns only" a set C if $u(S_1) = u(S_2)$ for every sets S_1 and S_2 satisfying $S_1 \cap C = S_2 \cap C$. Intuitively, the utility ignores those players outside of C.

The above theorem is very useful. Solving the original SV problem directly (applying the utility function v on the grand coalition $\mathscr{K}$) takes $O(2^{|\mathscr{K}|})$ time. Instead, we will

solve M sub-problems, each problem applying utility function v_m on coalition C_m, and then sum the respective SV results to obtain the final SV. The MID approach incurs only $O(M2^{\max\{|C_m|:m\in[M]\}})$ time, much faster.

But that is just theory. The biggest question which is not easy to answer is whether there exist a decomposition $\{(v_m, C_m)\}_{m=1}^{M}$ and, if so, how to find a good one. As such, MID realizability depends on the SV utility function. [ACC14] (and [CD19] similarly) proposed a decomposition for the case that utility function v is sub-additive: $v(S_1) + v(S_2) \geq v(S_1 \cup S_2)$ for every $S_1, S_2 \subset \mathscr{K}$. First, partition the grand coalition into M clusters, $\mathscr{K} = \bigcup_{m=1}^{M} C_m$, to maximize the intra-cluster player similarity, where similarity of two players i and j is defined as $d_{ij} = v(\{i\}) + v(\{j\}) - v(\{i,j\})$; i.e.,

$$\min \sum_{m} \sum_{m' \neq m} \sum_{i \in C_m} \sum_{j \in C_{m'}} d_{ij}.$$

Then, the sub-utility functions are defined as

$$v_m(S) = v(S \cap C_m).$$

This is an approximation of MID because the utility decomposition is not precise:

$$v(S) = v\left(\bigcup_{m=1}^{M} (S \cap C_m) \right) < \sum_{m=1}^{M} v(S \cap C_m) = \sum_{m=1}^{M} v_m(S).$$

The strict ($<$) inequality holds instead of equality because the utility function is assumed sub-additive; else, it must be strictly additive. The goal of the similarity partitioning is meant to minimize the effect of this imprecision.

2.4.2 Proposed Algorithm

MID has not been explored in the FL setting. We propose that for Federated SV we apply MID to each round-local utility function v_t of Eq. (2.11). Consider a particular

round t and, for the sake of presentation, let us ignore the index t. We propose to use a linear decomposition instead of a sum decomposition:

$$v = \sum_{m=1}^{M} v_m, \tag{2.12}$$

$$v_m(S) = v(S \cap C_m). \tag{2.13}$$

Extending from Theorem 2.4.1, thanks to the linearity of SV (which is stronger than the additivity property), we can compute the SV of each participant k in round t efficiently as

$$\phi(k; \mathscr{K}, v) = \sum_{m=1}^{M} \psi(k; v_m).$$

In general, Equality (2.12) does not hold precisely and so approximation is needed. The decomposition error is

$$\varepsilon(S) = v(S) - \sum_{m=1}^{M} v_m(S) \tag{2.14}$$

$$= \mathrm{acc}\left(\frac{\sum_{k \in S} \mathbf{w}_k}{|S|}\right) - \sum_{m=1}^{M} \mathrm{acc}\left(\frac{\sum_{k \in S \cap C_m} \mathbf{w}_k}{|S \cap C_m|}\right). \tag{2.15}$$

For a good approximation, a heuristic is to form a partition $\mathscr{K} = \bigcup_m C_m$ to maximize pairwise similarity inside each cluster. We can use METIS, a very fast graph partitioner tool, for the partitioning step. To further speed up the computation, for the SV game in each round t, we do not consider the entire participant set $\mathscr{K}$. In FL, due to large number of participants, the global model update in each round is broadcast to $K_0 < |\mathscr{K}|$ random participants. Therefore, we run SV only concerning this subset, instead of applying to $\mathscr{K}$. Putting all together, we propose the following algorithm to compute a practical SV for FL.

2.5 Evaluation Study

We conducted an evaluation study comparing three valuation methods:

Algorithm 3: Practical Federated Shapley Value (FSV_proposed)

- Initialize Fed SV $\phi(k) = 0$ for every participant $k \in \mathscr{K}$.
- In each round $t = 1, 2, ..., T$ of the FL process
  1. Let $\mathbf{w}_k^{(t)}$ be the current local model of participant k and $\mathscr{K}^{(t)} \subset \mathscr{K}$ the subset of K_0 participants selected to receive the global model update.
  2. Define a coalition game with $\mathscr{K}^{(t)}$ as the grand coalition and utility function
  $$v(S) = \text{acc}\left(\frac{\sum_{k \in S} \mathbf{w}_k^{(t)}}{|S|}\right).$$
  3. Build a similarity graph G of participants in $\mathscr{K}^{(t)}$ with similarity $d_{ij} = v(\{i\}) + v(\{j\}) - v(\{i, j\})$.
  4. Apply min-cut size-balanced graph partitioning on G to obtain M clusters $C_1, C_2, ..., C_m$.
  5. Compute the SV $\phi(k; C_m, v_m)$ for each participant $k \in C_m$. Utility function v_m is defined such that $v_m(S) = v_m(S \cap C_m)$ for every set S.
  6. Let $\psi(k; v_m) = \phi(k; C_m, v_m)$ for each participant $k \in C_m$; it is zero for $k \notin C_m$.
  7. Let $\psi(k) = \sum_{m=1}^{M} \psi(k; v_m)$ for each participant $k \in \mathscr{K}^{(t)}$ and $\psi(k) = 0$ otherwise. Then normalize these values to make them sum to 1.
  8. Update $\phi(k)$ += $\psi(k)$ for each participant $k \in \mathscr{K}^{(t)}$. No update for $k \notin \mathscr{K}^{(t)}$.
- Return $\phi(k)$ as the Federated SV for each participant $k \in \mathscr{K}$.

Table 1: Evaluation parameters: datasets and learning models

Dataset	Dist.	#train	#test	#features	#labels	FL model	#participants	BS	LR	#rounds	acccuracy
MNIST	IID	60K	10K	784	10	MLP	50	10	0.01	10/450	90%
MNIST	non-IID	60K	10K	784	10	MLP	50	10	0.01	10/800	76%
CIFAR10	IID	50K	10K	1,024	10	CNN	50	20	0.15	10/600	60%
CIFAR10	non-IID	50K	10K	1,024	10	CNN	50	20	0.15	10/800	35%
SYNTH	non-IID	4.8K	960	3	regression	MLP	16	10	0.001	10/200	MSE: 0.001

- `SV_central`: Assuming that all the local data are available for central training, we compute exact SV for each participant. Although this method does not apply to FL, it serves as a benchmark for comparison.

- `FSV_optimal`: This method computes Fed SV exactly, by applying the SV formula on FL, as we presented at the beginning of Section 2.4. Although this method is not feasible in practice, it is used as the optimal benchmark which we aim to approach.

- `FSV_sampling`: This method, presented in Section 2.3 and following the implementation in [WRZ20], is representative of the current way based on random sampling to apply SV to FL.

- `FSV_proposed`: This is our proposed method. In the partitioning step of each global update round, the participants selected for this round are partitioned in two clusters using Spectral Clustering [ACC14].

For `FSV_sampling` and `FSV_proposed`, in approximating the SV formula we use the same permutation sampling approach with the same parameters (Algorithm 2 of [WRZ20], $\varepsilon = 0.8$ and $\sigma = 0.05$).

2.5.1 Setup

The learning was applied to two real-world datasets, MNIST [Den12] (60K samples, hand-written digit recognition) and CIFAR10 [Kri09] (50K samples, object image recognition). Since the time complexity to compute exact SV in `SV_Central` and `FSV_Optimal` is exponential in terms of the number of participants, we generated a synthetic dataset, SYNTH (5760 samples, linear regression), to allow us to evaluate these methods in reasonable time. Each dataset is split into a testing set and a training set. The information about these datasets is summarized in Table 1. Note that in that table learning rate is denoted by RL, and #rounds represents number of local rounds/number of global rounds.

Two cases for the data distribution among the participants are considered: IID and non-IID. To generate the non-IID case, for the real-world datasets we have the same sample size for each participant but make the label distribution non-IID, and for the synthetic dataset we assign samples to participants uniformly at random but make the sample size for each participant follow a non-IID distribution. A good valuation method should value the participants equally in the IID case, whereas very differently in the non-IID case.

2.5.1.1 Real-World Data

In the evaluation with real-world data, the training set is distributed into 50 local training datasets corresponding to 50 participants. To generate the IID case, the training set is distributed uniformly at random such that each participant equally has for each of the 10 labels the same number of samples (120 samples in MNIST and 100 samples in CIFAR10). For the non-IID case, we make an imbalanced label distribution as usually used in non-IID FL evaluation studies [MMR17, HCP21, ZXL21, WKN20]. Specif-

ically, each participant has 90% of its training samples with only one dominant label while the remaining samples are with the other 9 labels equally likely. For example, for non-IID MNIST, a participant has 1080 samples with only one label and the remaining 120 samples are with labels equally chosen among the other 9 labels.

2.5.1.2 Synthetic Data

In the synthetic dataset, each sample is a 3-feature vector $< x_1, x_2, x_3 >$ and labeled according to a linear regression $y = w_1x_1 + w_2x_2 + w_3x_3$ where the model $< w_1, w_2, w_3 >$ is hidden. The feature values x follow a normal distribution. The label y is the ground truth added with a Gaussian noise with standard deviation 0.01. The samples are distributed among 16 participants such that the number of samples at each participant is a Zipf distribution with Zipf parameter set to 0.7 (this is typical Zipf skewness in practice). Hence, the participant with the most samples has roughly 20% of the total while the one with the fewest samples has roughly 3%. Given the sample size, the samples given to each participant are drawn uniformly at random. With this setup, the sample size can serve as a good representation for how much a participant contributes to the learning system. For example, if a valuation method results in that a participant having many samples is valued less than one having very few samples, this method is questionable.

2.5.1.3 FL Learning Models

In setting up FL, we use Multi-Layer Perceptron (MLP) as the learning method for the MNIST dataset and Convolutional Neural Network (CNN) for CIFAR10. The setup details are as follows:

- **MLP for MNIST** (203,530 model parameters): Fully connected (784, 256) $\rightarrow$ sigmoid activation $\rightarrow$ Fully connected (256, 10) $\rightarrow$ Softmax().

- **CNN for CIFAR10** (258,762 model parameters): we use the convention (in channels, out channels, kernel) for convolutional layers and ReLU activation after convolutional layers. Specifically, Conv2D(in_channels = 3, out_channels = 32, kernel = 3) $\rightarrow$ MaxPool2D(2, 2) $\rightarrow$ Dropout(p = 0.1) $\rightarrow$ Conv2D(in_channels = 32, out_channels = 64, kernel = 3) $\rightarrow$ MaxPool2D(2, 2) $\rightarrow$ Dropout(p = 0.1) $\rightarrow$ Conv2D(in_channels = 64, out_channels = 128, kernel = 3) $\rightarrow$ MaxPool2D(2, 2) $\rightarrow$ Dropout(p = 0.1) $\rightarrow$ Fully connected(128*2*2, 256) $\rightarrow$ ReLU() $\rightarrow$ Fully c onnected(256, 128) $\rightarrow$ ReLU() $\rightarrow$ Fully connected(128, 10) $\rightarrow$ Softmax().

For the synthetic dataset, we use a MLP model with three neurons for the input layer (to match with the three features) and one neuron for the output layer (to match with the ground truth). There is no hidden layer for this MLP. Mean Squared Error (MSE) is used for the loss function.

We implemented the above neural networks, MLP and CNN, with Torch. For the SGD algorithm (Algorithm 1) in FL, we employed its "mini-batch" version [MMR17] with batch size and learning rate given in Table 1; this parameter choice was chosen to fit the datasets reasonably. After each global update, it is broadcast to a random subset of $K_0 = 40$ out of $K = 50$ participants (80%). The number of local training rounds and the number of global model update rounds are also given in Table 1. These parameters are chosen so that the test accuracy of FL is in the range as specified in this table; these accuracies consistent with the benchmark in the literature.

2.5.2 Results

In this section, we first show the efficiency of our method, `FSV_proposed`. Secondly, we evaluate SVs computed in two fashions: central learning and federated learning (`SV_central` and `FSV_proposed`, respectively). We finally compare our fed-

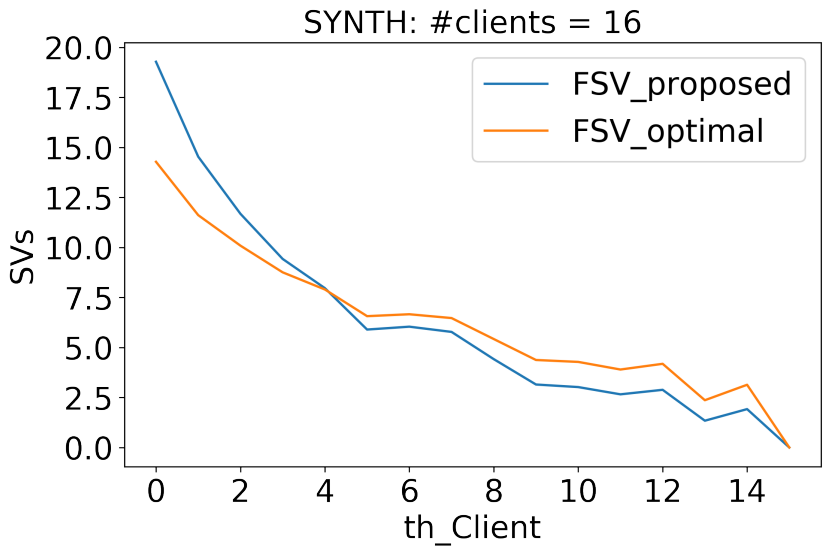


Figure 1: `FSV_proposed` vs. `FSV_optimal`: The x-axis is each participant, sorted in the decreasing order of training size. The y-axis is the importance score (SV).

erated SVs (`FSV_proposed`) with federated SVs computed by the sampling method (`FSV_sampling`).

2.5.3 The Efficiency of FSV_proposed

We can see the reliability and computation time of `FSV_proposed` in this section. Here, the SYNTH dataset is used because it is small enough to be feasible for exact computation of the proposed and optimal methods. We also run the two methods on the same CPU configuration for fair time consuming comparison.

Figure 1 reveals the FSV of all participants resulted from `FSV_optimal` and `FSV_proposed` for the SYNTH dataset. Obviously, `FSV_proposed` is quite consistent with `FSV_optimal` which serves as an optimal benchmark. The trend of `FSV_proposed` aligns well with that of `FSV_optimal` and the variation between two curves is small.

Figure 2 illustrates that `FSV_proposed` computes SVs stably over time. Clearly, each

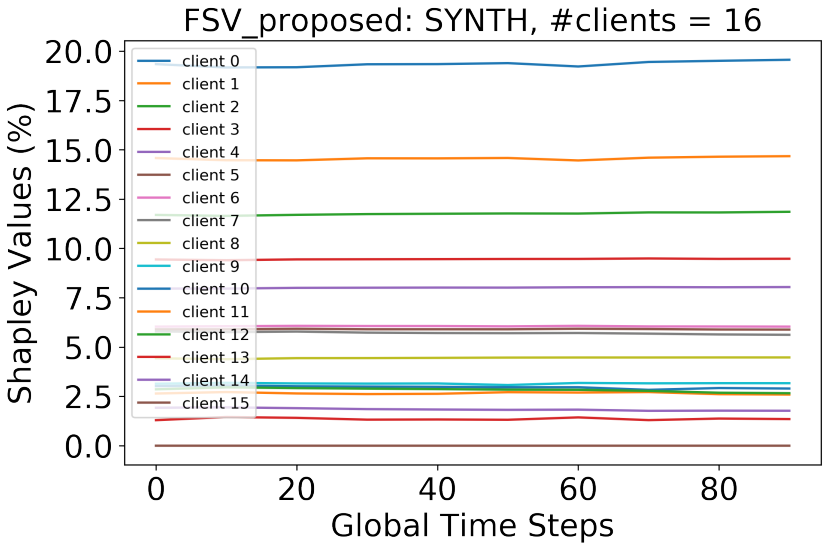


Figure 2: The stability of SVs computed by `FSV_proposed` over the time: The x-axis is each global time step to train the global model. The y-axis is the importance score (SV).

Table 2: Computation time (in seconds), SYNTH dataset

`FSV_proposed`	311.4368
`FSV_sampling`	22903.7091

participant has the same SV in all time steps. This is expected since we have all participants join the federation in each training round.

Table 2 shows the computation time to compute the FSV of `FSV_optimal` and `FSV_sampling`. As we can see `FSV_proposed` is significantly faster than `FSV_optimal`, roughly 73.5 times. This is a big achievement in making computing FSV practical.

2.5.3.1 Central SV vs. Federated SV

Figure 3 compares `SV_central` and `FSV_proposed`. Again, we use the SYNTH dataset for exact computation. As seen, they are different to the extent that the former should not serve as a benchmark to represent SV in a FL setting. With respect to the training

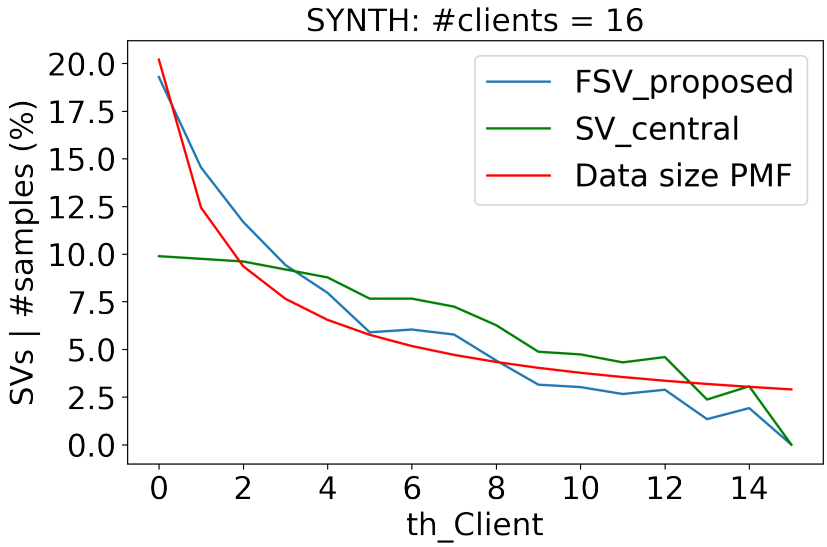
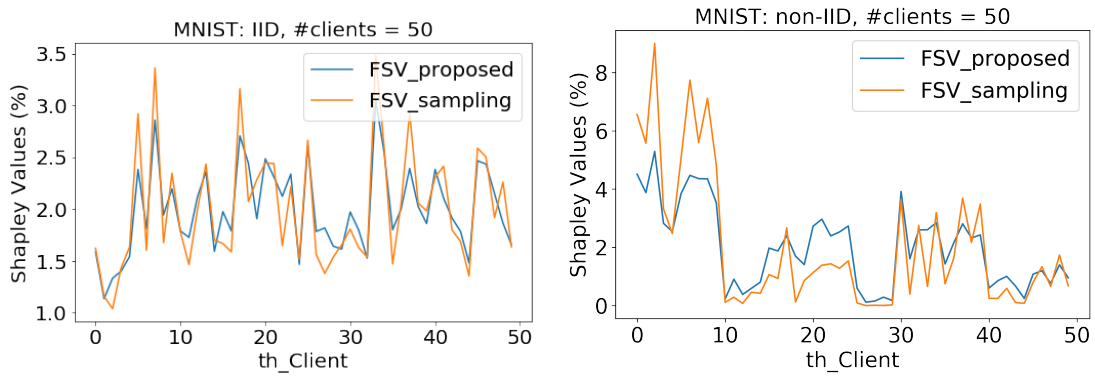


Figure 3: `SV_central` vs. `FSV_proposed`: The x-axis is each participant, sorted in the decreasing order of training size. The y-axis is the importance score (SV). To be fair, the score given to a participant should follow the pattern of its training size.

size distribution, which is best reflective of participant contributions for this dataset, to be fair the score given to a participant should follow the pattern of its training size. `FSV_proposed` is closely consistent with this fairness. In contrast, `SV_central` is not; for example, it gives the same value to participant 0 and participant 2 even though participant 0 contributes twice more training samples. This result demonstrates that 1) SV is a good measure to value participant contributions in FL, and 2) The FL setting has a different impact on the valuation of participants compared to the case they are valued in a centralized setting.

2.5.3.2 Federated SV: Proposed method vs. Sampling method

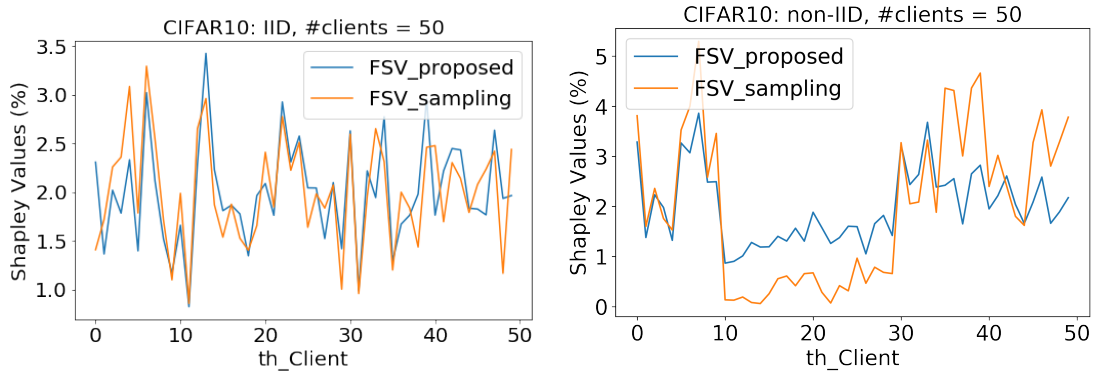
Figure 4 shows the FSV of every participant resulted from `FSV_sampling` and `FSV_proposed` for the MINIST dataset in both IID and non-IID case. The FSV variation



(a) IID (b) non-IID

Figure 4: Federated SV for MNIST dataset: `FSV_sampling` or `FSV_proposed` are consistent in the ranking order of importance values, but `FSV_proposed` offers a smoother valuation especially for the non-IID case.

among the participants is wider for the non-IID case than the IID case, which is understandable because the data distribution among them is skewed in the non-IID and uniform in the IID. Besides that triviality, we observe that `FSV_sampling` or `FSV_proposed` are consistent with each other in that the participants are valued in the same importance order in both methods. However, more interestingly, `FSV_proposed` provides a noticeably smoother valuation, especially for the non-IID case. Whereas `FSV_sampling` gives highly important participants very high values and lowly important ones very low values, `FSV_proposed` is less extreme. By having smaller gaps between them, the proposed method avoids the kind of winer-take-it-all risk. The same pattern is demonstrated for the CIFAR10 dataset (Figure 5).



(a) IID (b) non-IID

Figure 5: Federated SV for the CIFAR10 dataset: `FSV_sampling` or `FSV_proposed` are consistent in the ranking order of importance values, but `FSV_proposed` offers a smoother valuation especially for the non-IID case.

2.6 Conclusions

Valuation of participant contribution is crucially important in Federated Learning (FL). It is useful not only to recognize and incentivize good participants, but also to make the learning algorithm more efficient by emphasizing less on those participants of low influence. As Machine Learning, and FL in particular, can be cast as a kind of coalition game, it is tempting to apply Shapley Value (SV), the de facto best valuation method in cooperative game theory. Earlier attempts, which are very few, are not sufficiently efficient. We have presented a novel method, called Practical Federated Shapley Value (PFSV), which leverages the idea of multi-issue decomposition to make the SV computation for FL faster and more accurate. Its superiority compared to the literature has been substantiated in our evaluation study with both real-world and synthetic data.

CHAPTER 3

EDGE ASSIGNMENT IN EDGE FEDERATED LEARNING

Despite its promise, FL depends on a central server for repeated aggregation of local training models, which is prone to become a performance bottleneck. Therefore, one can combine FL with Edge Computing: introduce a layer of edge servers to each serve as a regional aggregator to offload the main server. The scalability is thus improved, however at the cost of learning accuracy. We show that this cost can be alleviated with a proper choice of edge server assignment: which edge servers should aggregate the training models from which local machines. Specifically, we propose an assignment solution which is especially useful for the case of non-IID training data (well-known to hinder today's FL performance). Our findings are substantiated with an evaluation study using real-world datasets.

3.1 Introduction

The reliance on the parameter server for central aggregation of local training models in FL settings is a vulnerability. For large-scale applications, the server can easily become a bottleneck. Hence, one can combine FL and Edge Computing to overcome this challenge.

Edge Computing has emerged as a viable technology for network operators to push computing resources closer to the users to avoid long-haul crossing of the network core, thus improving network efficiency and user experience. Originally initiated for cellular networks to realize the 5G vision [ETS14], Edge Computing has become mainstream with broader applicability in many types of long-distance wired or wireless networks [MYZ17, AZT18], serving various data- or computing-intensive applications; e.g., video optimization [XLT17], content delivery [SHZ17], big data analytics [NRS17], roadside assistance [PGF18], and augmented reality [LHO18].

In this chapter, we assume an Edge Computing architecture for FL. In this so-called Edge Federated Learning (eFL) architecture [LZS20, LLH20, WTS19], a middle layer of edge servers, which are commodity servers deployed at the edge of the network near the participants, serves as "regional" aggregation servers. The learning in eFL works in a hierarchical manner as follows: (1) Each regional server computes a regional model by aggregating the local models in its region; and (2) The global server (the original parameter server) updates the global model by aggregating only the regional models. Performance bottleneck is no longer an issue because the global server only needs to communicate with the edge servers, which are not many, and each edge server communicates only with the local participants in its region, which is also a much smaller group.

However, eFL incurs an inevitable tradeoff: learning accuracy. The more decentralization of the central aggregation server into distributed edge servers, the worse learning accuracy as a result. Figure 6 illustrates an example. We compared the learning accuracy of FL and eFL on a real-world dataset (MNIST [Den12]) in fair parameter settings. While FL offers an accuracy of more than 90%, eFL by deploying more edge servers results in an accuracy increasingly worse (reduced to 80%). This is understandable because a regional server in eFL gathers information only from its region whereas the

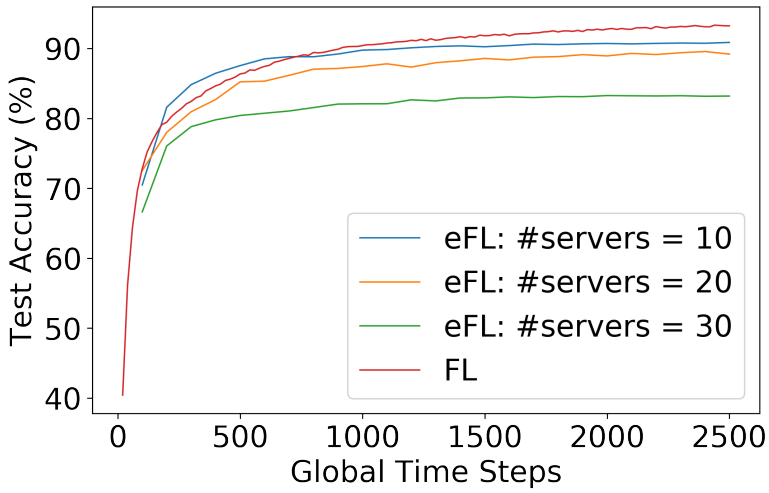


Figure 6: Tradeoff between decentralization of the central aggregation server into distributed edge servers and resulted learning accuracy. Here, the learning is applied to the real-world MNIST dataset [Den12] distributed non-iid among 300 participants. The x-axis is the global time during the algorithm running. The y-axis is the accuracy when evaluated against the test dataset.

parameter server in FL receives information from all the participants. Although the global server in eFL will eventually receive information from all the participants via edge aggregation, the learning is slower to converge.

The tradeoff can be significant and so we should minimize it, hence the focus of this chapter. We propose an approach by solving an edge server assignment problem: given a number of edge servers, choose which servers to aggregate which participants to maximize the learning accuracy. To date, it is implicitly understood in eFL designs that this assignment is given a priori. We make the following contributions:

- The proposed edge assignment problem does not require any parameters. It is purely data-driven but protects data privacy. The only assumption made is our knowing the local models after a few early rounds of local training. To our knowl-

edge, this work is the first effort on the edge assignment problem aimed to maximize learning accuracy that can work for every eFL setting.

- The problem is not trivial because there is no closed form to formulate the learning accuracy as an objective function. Due to FL's iterative manner, its accuracy can only be measured on the fly after the procedure completes. The best we could do thus is a heuristic approach. Yet, no obvious heuristic benchmark exists other than random assignment. We propose a heuristic solution that is data-driven and also useful as a better benchmark.

- A well-known hinderance to the learning accuracy of eFL, which is inherited from FL, is the non-IID (non-independent and identically distributed) nature of local training data observed in the real world [ZXL21, WKN20, SWM20]. Indeed, the edge assignment problem is not much of a need if the data is IID, because any balanced-size assignment should suffice. Our solution is noticeably effective in the non-IID case, where the assignment problem matters.

The remainder of the chapter is organized as follows. Related work is discussed in Section 3.2. The setting and objective of our server assignment problem are described in detail in Section 3.3. The proposed algorithm is proposed in Section 3.4. The results of the evaluation study are analyzed in Section 3.5. The chapter concludes in Section 3.6 with pointers to future work.

3.2 Related Work

The use of Edge Computing in eFL to address the performance bottleneck of the FL server has only recently been investigated, but gained traction quickly. Good surveys of the state of the art are provided in [LLH20, XYT21, AHS22]. In [LLH20], a compre-

hensive review of eFL categorizes the challenges in eFL into different topics, namely communication cost, resource allocation, privacy and security, and considers applications of eFL in cyberattack detection, edge caching and computation offloading, base station association, and vehicular networking. The survey in [XYT21] summarizes the research problems and methods in eFL in terms of applications, development tools, communication efficiency, security, privacy, migration, and scheduling. It also suggests open problems in eFL. The authors in [AHS22] present a report as a result of investigating more than 500 FL papers published between 2016 (the year FL was first introduced) and October 2021. This report shows that 57.3% of these papers discussed eFL. It is clear that FL and Edge Computing are well-suited for each other.

In [LZS20], the authors provide proofs that under some feasible assumptions eFL can provide a learning accuracy approaching that in a centralized learning setting where all the training data are collected in one place. One of the assumptions is the IID -ness on the distributed training datasets, which is often implicit in most machine learning algorithms. However, in a FL (and eFL) setting, the training data reside independently in different physical places and it is often the case that their distribution is non-IID, meaning that the data distribution at each place may be very statistically different from that at another place [ZXL21, WKN20, SWM20]; put another way, the participating data are strongly skewed. In a typical implementation of FL, the model aggregation in each iterative round involves only a subset of local models and a careless selection of them ignoring the non-IID of the training data may hurt the global model's convergence and learning accuracy [WKN20]. Indeed, the local models can overfit local data, leading to a poor global model.

Few studies have considered non-IID data in an eFL setting. An eFL approach is proposed in [HCP21] where edge servers can have overlapping coverages. That is, a local model may be included in the model aggregation at multiple servers; it is updated

based on model updates sent from these multiple servers. With the overlapping, the servers are closer-to-IID (or, equivalently, less non-IID) in terms of the training data they aggregate. This approach, however, incurs more communication cost because a local model may be sent to more than one server. In contrast, our work assumes non-overlapping (disjoint) servers. We maximize the learning accuracy without increasing the communication cost.

The authors in [YLL20a] focus on reducing the computation cost incurred due to model calculations at the mobile devices. They propose an eFL approach in which the edge servers do not only aggregate the local models, but also collaboratively join in the computation of the local models. Specifically, each server collaborates with a constant k of participants. The (deep learning) model is a sequence of two types of layers: the low layers and high layers. At each participant, the raw training data go through the low layers, at the end of which the output including the model weights and the ground truth is uploaded to the corresponding edge server. At each edge server, upon receipt of the low-layer outputs from its collaborating participants, will perform calculations at the high layers of the model. This approach's drawback is that local devices must share with their edge servers partial raw local data (which is the ground truth aforementioned). This violates the privacy preservation of the FL paradigm. In our eFL setting, local models are trained completely at the participant side and the job of an edge server is only to aggregate local models to obtain a regional model. Hence, no raw data is shared beyond its local owner.

To our knowledge, the only previous work sharing the same goal with our work, i.e., solving the edge server assignment problem in eFL, is [MAM22]. In this reference work, the objective is to minimize the divergence between eFL's global model and the "centralized" global model; the latter is the result of running the same learning method assuming all the data centralized in one place. The fundamental difference between this

work and ours is two-fold. First, they assume that the local data distribution at each and every local participant is known to the global server. Second, they assume that the global data distribution, combining all the local data together, is also known. These two assumptions are too strong, thus limiting the applicability of their edge assignment solution. In contrast, our work makes no such assumptions, hence suitable for any eFL setting.

3.3 Background and Motivation

Recalling the background discussed in 2.3 with the SGD algorithm [algorithm 1] and the FedAvg algorithm [algorithm 2], in this section, we will use the two algorithms to elaborate an algorithm for edge federated learning.

3.3.1 Edge Federated Learning (eFL)

Let M be the number of edge servers. Let $z_{ik} \in \{0,1\}$ denote the assignment of participant k to edge server i. The aggregation coverage of server i is the set of participants k such that $z_{ik} = 1$. Our setting assumes that each a participant is assigned to only one edge server, i.e., $\forall k \in [K] : \sum_{i=1}^{M} z_{ik} = 1$.

Intuitively, we can think of eFL as a a distributed system of FL subsystems, each running aggregation on an edge server, thus having a corresponding edge model. The job of the global/central server in this distributed system is simply to compute the global model by averaging the edge models. We can modify the FedAvg algorithm above to work for eFL as follows:

We denote this algorithm with $\texttt{eFL}(x, \mathbf{w}^{(0)}; M, \{\mathscr{D}_1, \mathscr{D}_2, ..., \mathscr{D}_k\})$ which uses $\mathbf{w}^{(0)}$ as the initial global model and applies to an eFL system with M edge servers and K par-

Algorithm 4: Edge Federated Averaging (EdgeFedAvg)

1. The server starts with an initial model $\mathbf{w}^{(0)}$
2. For each global round t = 1, 2, ..., T:
   (a) The server broadcasts model $\mathbf{w}^{(t-1)}$ to all the edge servers
   (b) In response, in parallel, each edge server i runs FL with its own subsystem, consisting of only the participants assigned, to compute/update its edge model
   $$e_i^{(t)} = \texttt{FL}(\mathbf{w}^{(t-1)}; \{\mathscr{D}_k | z_{ik} = 1\})$$
   and sends it to the server. Note that this FL starts with the global model $\mathbf{w}^{(t-1)}$ just received from the server.
   (c) The server updates the global model by the averaging the updated edge models, weighted by the size sum of the training datasets aggregated by each edge server:
   $$\mathbf{w}^{(t)} = \sum_{i=1}^{M} e_i^{(t)} \frac{\sum_{k=1}^{K} |\mathscr{D}_k| z_{ik}}{\sum_{k=1}^{K} |\mathscr{D}_k|}$$
3. Return $\mathbf{w}^{(T)}$

ticipants with local training datasets $\mathscr{D}_1$, $\mathscr{D}_2$, ..., $\mathscr{D}_K$, respectively, and $\boldsymbol{Z}$ the assignment matrix of these K participants to the M edge servers.

One can prove that when certain conditions hold, like in FL [LHY20], eFL will eventually converge to a performance comparable to that of the centralized learning setting [LZS20], which is true for both convex and non-convex loss functions.

3.3.2 Problem Statement

The implementation detail of eFL requires some parameters: the number of rounds ($T = G$) the global server updates the global model from the edge models, the number of rounds ($T = E$) each edge server updates the edge model from the local models before sending it to the global server, and the number of rounds ($T = L$) each participant performs SGD before sending the updated local model to its edge server.

In this work, we consider an eFL setting with given G, E, and L. Its learning performance then depends on the edge server assignment $\boldsymbol{Z} = \{z_{ik}\}_{M\times K}$. Our goal is to find the best $\boldsymbol{Z}$ to maximize eFL accuracy. To date, it is implicit in eFL designs that this assignment is random or given a priori without justification. Although eFL has gained a lot of traction, surprisingly the edge server assignment is rarely touched, except the work [MAM22] which is different from ours as discussed in Section 3.2.

When the training data distribution among the participants is non-IID, Equality (2.4) is untrue: averaging $F(\mathbf{w}, \mathscr{D}_k)$ over $\mathscr{D}_k$ (the local training dataset at participant k as a non-IID subset of $\mathscr{D}$) can unpredictably be bad as an approximation for $F(\mathbf{w}, \mathscr{D})$. The non-IID case remains a severe hindrance to FL accuracy. With eFL, if a proper edge assignment $\boldsymbol{Z}$ is used, we can hope to neutralize the non-IID effect.

If the IID assumption holds true, FL works effectively and the edge assignment problem for eFL does not matter much, because a random assignment solution that

balances the coverage size of each edge server would provide a good learning accuracy. We thus seek an assignment solution $\boldsymbol{Z}$ that works more effectively for the non-IID case, yet as well as the random solution for the IID case. Unfortunately there exists no closed form to formulate the learning accuracy as an objective function. Due to the iterative manner of eFL, the learning accuracy can only be measured on the fly after the procedure completes.

3.4 The Assignment Algorithm

Our heuristic is to include in the aggregation coverage of an edge server those participants whose training datasets are statistically as diverse as possible. For example, in an eFL setting with 4 participants and 2 edge servers such that participant 1's and participant 2's training datasets consist of mostly label A and participant 3's and participant 4's mostly label B, participants 1, 3 should be combined on one edge server and participants 2, 4 on the other edge server. This way, each edge server has a comprehensive coverage of training data (having both labels A and B).

3.4.1 The rationale

We elaborate this heuristic mathematically as follows. Think of each edge server as "virtual" participant with a virtual training set $\mathbb{D}_i \triangleq \bigcup_{k:z_{ik}=1} \mathscr{D}_k$. The loss $F(.)$ can be

re-written as follows

$$F(\mathbf{w};D) = \frac{1}{|\mathscr{D}|}\sum_{(x,y)\in\mathscr{D}} l\Big(\mathbf{w};x,y\Big) \tag{3.1}$$

$$= \sum_{i=1}^{M}\frac{|\mathbb{D}_i|}{|\mathscr{D}|}\underbrace{\left(\frac{1}{|\mathbb{D}_i|}\sum_{(x,y)\in\mathbb{D}_i} l(\mathbf{w};x,y)\right)}_{F(\mathbf{w};\mathbb{D}_i)} \tag{3.2}$$

$$= \sum_{i=1}^{M}\underbrace{\frac{|\mathbb{D}_i|}{|\mathscr{D}|}}_{\beta_i} F(\mathbf{w};\mathbb{D}_i) = \sum_{i=1}^{M}\beta_i F(\mathbf{w};\mathbb{D}_i). \tag{3.3}$$

Let $Q_i(x,y)$ be the ground-truth (hidden) distribution representing the samples in $\mathbb{D}_i$. Denote $E_i(\mathbf{w}) \triangleq \mathbb{E}_{Q_i}[F(\mathbf{w};\mathbb{D}_i)]$ and $E(\mathbf{w}) \triangleq \sum_{i=1}^{M}\beta_i E_i(\mathbf{w})$. Applying Proposition 2 of [ZWB21], the global optimum $\mathbf{w}^* \triangleq \arg\min_{\mathbf{w}} F(\mathbf{w};\mathscr{D})$ introduces the following error bound when minimizing $E_i(\mathbf{w})$, with probability $1-\delta$:

$$E_i(\mathbf{w}^*) - \min_{\mathbf{w}} E_i(\mathbf{w}) \leq \frac{A}{|\mathscr{D}|} + 2\left\langle Q_i - \sum_{j=1}^{M}\beta_j Q_j \right\rangle \tag{3.4}$$

where $\langle f\rangle \triangleq \int_{x,y}|f(x,y)|dxdy$ and A is a constant only depending on the definition of the loss function l, the data dimensionality, and the threshold δ (which can be made arbitrarily small).

Summing this over all edge servers i's, we have

$$E(\mathbf{w}^*) - \min_{\mathbf{w}} E(\mathbf{w}) \tag{3.5}$$

$$= \sum_{i=1}^{M} \beta_i E_i(\mathbf{w}^*) - \min_{\mathbf{w}} \sum_{i=1}^{M} \beta_i E_i(\mathbf{w}) \tag{3.6}$$

$$\leq \sum_{i=1}^{M} \beta_i E_i(\mathbf{w}^*) - \sum_{i=1}^{M} \beta_i \min_{\mathbf{w}} E_i(\mathbf{w}) \tag{3.7}$$

$$= \sum_{i=1}^{M} \beta_i \left(E_i(\mathbf{w}^*) - \min_{\mathbf{w}} E_i(\mathbf{w}) \right) \tag{3.8}$$

$$\leq \sum_{i=1}^{M} \beta_i \frac{A}{|\mathscr{D}|} + 2 \sum_{i=1}^{M} \beta_i \left\langle Q_i - \sum_{j=1}^{M} \beta_j Q_j \right\rangle \tag{3.9}$$

$$\leq \frac{A}{|\mathscr{D}|} + 2 \sum_{i=1}^{M} \beta_i \left\langle Q_i - \sum_{j=1}^{M} \beta_j Q_j \right\rangle. \tag{3.10}$$

Although the left hand side of this inequality does not depend on the edge assignment $\mathbf{Z}$, its right-hand-side upper bound does. Our heuristic is to compute $\mathbf{Z}$ that minimizes this upper bound. This happens when $Q_i(.) = Q_j(.) = Q(.)$ for all i, j, because that leads to

$$\langle Q_i - \sum_{j=1}^{M} \beta_j Q_j \rangle = \langle Q - Q \sum_{j=1}^{M} \beta_j \rangle = 0.$$

In other words, ideally, we want the aggregate probability distribution of the samples virtually belonging to each server is identical. This mathematically justifies the intuition that the data distribution at the edge level should be made IID to minimize the global learning loss. However, we do not know the underlying distribution of each participant. Therefore, we apply two heuristics: 1) participants belonging to a server should be statistically diverse, and 2) we use the diversity of the local models observed empirically to represent this statistical diversity.

Consider participant k and let $P_k(x,y)$ denote its data probability distribution. From the strong law of large numbers, when the training data size $|\mathscr{D}_k|$ is sufficiently large, the loss $F(\mathbf{w};\mathscr{D}_k)$ should converge to the true risk $F_k(\mathbf{w})$ of model $\mathbf{w}$ for participant k,

which is

$$F(\mathbf{w};\mathscr{D}_k) \approx F_k(w) \triangleq \mathbb{E}_{P_k}[l(\mathbf{w};x,y)] = \int l(\mathbf{w};x,y)dP_k.$$

Participant k's local model after one round of SGD is

$$\mathbf{w}_k^{(1)} = \mathbf{w}^{(0)} - \eta\nabla_w F(\mathbf{w}^{(0)};\mathscr{D}_k) \approx \mathbf{w}^{(0)} - \eta\nabla_w F_k(\mathbf{w}^{(0)}).$$

Since $\nabla_w F_k(w) = \int \nabla_w l(\mathbf{w};x,y)dP_k$, considering two participants k and k', their model divergence after one SGD round is

$$\|\mathbf{w}_k^{(1)} - \mathbf{w}_{k'}^{(1)}\| = \eta\left\|\nabla_w F_k(\mathbf{w}^{(0)}) - \nabla_w F_{k'}(\mathbf{w}^{(0)})\right\| \tag{3.11}$$

$$= \eta\left\|\int \nabla_w l(\mathbf{w}^{(0)};x,y)d(P_k - P_{k'})\right\| \tag{3.12}$$

$$\leq \eta C\left|\int d(P_k - P_{k'})\right|, \tag{3.13}$$

where constant C denotes the upper-bound $\max_{x,y}\|\nabla_w l(\mathbf{w}^{(0)};x,y)\|$ (the maximal norm of the gradient of the per-sample loss at $\mathbf{w}^{(0)}$). This bound exists given the typically assumed convexity and smoothness of the loss function (standard assumptions in FL to guarantee its convergence [LHY20, ZWB21]); for example, this holds when l is the 2-norm.

This inequality implies that if participants k and k' have similar data distributions P_k and $P_{k'}$, they should have small model divergence, and vice versa, if they have large model divergence, their data distributions should be largely different. Therefore, the model divergence after the first SGD round (even better if more rounds take place) is a good representation for the statistical data distribution difference between two participants.

3.4.2 The algorithm

We define the "distance" between two participants to be their 1st-round model divergence: $dist(k,k') = \|\mathbf{w}_k^{(1)} - \mathbf{w}_{k'}^{(1)}\|$. To compute this distance, all the participants only need to send the server their 1st-round local model (local data privacy remains intact).

Once these distances are computed, we represent the participants as vertices in a graph whose edge weight between vertex k and vertex k' is the corresponding distance $dist(k,k')$. We (the server) then apply k-balanced min-cut graph partitioning (e.g., using METIS algorithm [KK98]) to divide the graph into M clusters of participants, each cluster being assigned to an edge server. Since each cluster consists of vertices with heavy-weight edges (correspondingly, long distance), we achieve the goal of grouping participants as statistically distant as possible into an edge server. See Algorithm 5 for a full description.

3.5 Evaluation Study

We conducted an evaluation study comparing three learning algorithms: 1) `eFL_metis`: the `eFL` algorithm (Algorithm 4) using our assignment solution (Algorithm 5) with METIS [KK98] for graph partitioning; 2) `eFL_rnd`: the `eFL` algorithm using a random assignment; and 3) `FL`: the FedAvg algorithm (Algorithm 2).

3.5.1 Evaluation setup

The learning was applied to two real-world classification datasets: MNIST [Den12] (hand-written digit recognition) and CIFAR10 [Kri09] (object image recognition). Each dataset is split into a testing set and a training set. The training set is distributed into 300 local training datasets (300 participants). The number of classes, features, training

Algorithm 5: Edge Server Assignmemt (EdgeFedAvg). Let $\mathbf{w}^{(0)}$ be the (arbitrary) initial global model.

1. The global server sends $\mathbf{w}^{(0)}$ to all participants.
2. Each participant k:
   (a) Perform $\mathbf{w}_k^{(1)} = \texttt{SGD}(\mathbf{w}^{(0)}; \mathscr{D}_k)$; or better if running it for more than 1 round.
   (b) Send $\mathbf{w}_k^{(1)}$ to the global server.
3. The server computes the assignment matrix $\boldsymbol{Z}$ as follows:
   (a) Construct an undirected edge-weighted graph G consisting of K vertices, each representing a participant, such that the weight of an edge connecting two vertices k and k' is $dist(k,k') = \|\mathbf{w}_k^{(1)} - \mathbf{w}_{k'}^{(1)}\|$.
   (b) Apply a k-balanced min-cut graph partitioning algorithm to divide G in M clusters and assign the participants in each cluster to an edge server.

Table 3: Real-world datasets used in evaluation

Dataset	train size	test size	#features	#classes
MNIST [Den12]	60,000	10,000	784	10
CIFAR10 [Kri09]	50,000	10,000	1,024	10

samples, and test samples are given in Table 3. The three aforementioned algorithms are compared in terms of the learning accuracy when applied to the testing set.

The local data distribution can be IID or non-IID. For the IID case, the training set is distributed uniformly at random such that each participant equally has for each of the 10 labels the same number of samples (20 samples in MNIST and 16 samples in CIFAR10). For the non-IID case, we make an imbalanced label distribution as usually used in non-IID FL evaluation studies [MMR17, HCP21, ZXL21, WKN20]. Specifically, each participant has 90% of its training samples with only one dominant label while the remaining samples are with the other 9 labels equally likely. For example, for non-IID MNIST, a participant has 180 samples with only one label and the remaining 20 samples are with labels equally chosen among the other 9 labels.

For the eFL setting, we vary number of edge servers in the set $M \in \{10, 20, 30\}$ to explore the effect of decentralization in eFL on the learning performance of the global model. Respectively, each edge server aggregates 30, 15 and 10 participants on average. Two configurations for the numbers of edge rounds and local rounds are considered: (E = 2, L = 10) and (E = 5, L = 20). The former configuration represents the case that edge and global aggregation updates are more frequent and the latter represents the case less frequent. It is well-known in FL for the non-IID case that when L is higher (longer local SGD computation before aggregation) can cause more divergence between local models, thus slowing down the convergence speed of the global model [MMR17]. In total, we have 24 model cases for each eFL algorithm.

For the FL setting, M and E are irrelevant; the only parameter is L, set to $L = 20$ or $L = 100$ so that the number of global model updates resulted is the same for both FL and eFL (corresponding to cases (E = 2, L = 10) and (E = 5, L = 20, respectively). In total, we have 4 model cases for FL.

Because training a model on CIFAR10 is expensive, we only have FL results for MNIST. Multi-Layer Perceptron (MLP) was used as the learning method for the MNIST dataset while Convolutional Neural Network (CNN) was used for CIFAR-10. The setup details are as follows:

- **MLP for MNIST** (203,530 model parameters): Fully connected (784, 256) $\rightarrow$ sigmoid activation $\rightarrow$ Fully connected (256, 10) $\rightarrow$ Softmax().
- **CNN for CIFAR10** (258,762 model parameters): Conv2D(in_channels = 3, out_channels = 32, kernel = 3) $\rightarrow$ MaxPool2D(2, 2) $\rightarrow$ Dropout(p = 0.1) $\rightarrow$ Conv2D(in_channels = 32, out_channels = 64, kernel = 3) $\rightarrow$ MaxPool2D(2, 2) $\rightarrow$ Dropout(p = 0.1) $\rightarrow$ Conv2D(in_channels = 64, out_channels = 128, kernel = 3) $\rightarrow$ MaxPool2D(2, 2) $\rightarrow$ Dropout(p = 0.1) $\rightarrow$ Fully connected(128*2*2, 256) $\rightarrow$ ReLU() $\rightarrow$ Fully connected(256, 128) $\rightarrow$ ReLU() $\rightarrow$ Fully connected(128, 10) $\rightarrow$ Softmax().

We implemented these two neural networks with PyTorch. For the SGD algorithm (Algorithm 1) in FL and eFL, we employed its "mini-batch" version [MMR17] with batch size set to 10 for MNIST and 20 for CIFAR10; the learning rate η is set to 0.01 and 0.15, respectively. This parameter choice was chosen to fit the datasets reasonably. To form the participant graph in our edge assignment solution, the model divergence between two participants is computed using local models resulted after running SGD for 10 rounds. In computing this divergence, we use the Minkowski metric of order 1 (we found that order 2, which is the Euclidean distance, is worse as a representation for the statistical difference based on the raw data).

3.5.2 Effect of the number of edge servers

Figure 7 shows the test accuracy of eFL as an effect of the number of edge servers (M) for the MNIST case (here, we chose `eFL_metis` as an example; the result pattern for `eFL_rnd` is similar). Clearly, increasing the number of edge servers in eFL results in accuracy loss. This is understandable since, given a fixed number of participants, as more edge servers are placed, each will aggregate from fewer participants. This causes less diversity of data distribution at the edge aggregation and so regional models tend to accumulate divergently. As a result, the global model which is aggregated from these regional models gets worse. The pattern for the CIFAR10 dataset is similar (Figure 8).

3.5.3 Effect of edge server assignment

Figure 9 compares `eFL_metis` to `eFL_rnd`, plotting their ratio of test accuracy. A ratio larger (smaller) than 1 means better (worse) test accuracy for the former. First, it is observed that `eFL_metis` is comparable to or better than `eFL_rnd` in all configurations. They are comparable in the IID case, which is expected because a random assignment would neutralize the non-IID case effectively. They are also comparable when few edge servers are deployed, in either IID or non-IID case. This is because with few edge servers, the degree of decentralization in eFL is not that significant, giving little room for accuracy improvement as a result of an edge server assignment. That said, we still see a slightly better accuracy for `eFL_metis` compared to `eFL_rnd`.

In the non-IID case, especially significant with more edge servers deployed (when the server assignment is critical to the learning accuracy), `eFL_metis`'s superiority becomes more noticeable. For example, as seen in Figure 9(b) for the MNIST dataset, using 30 edge servers, `eFL_metis` improves over `eFL_rnd` by 15%. This improvement is 20% for the CIFAR10 dataset; see Figure 10(b).

Figure 11 provides a radar-chart overview of the comparison summarizing all the configurations in terms of the number of local updates, the number of edge updates, and the number of edge servers. It is expected that FL is the best and the difference between `eFL_metis` and `eFL_rnd` is not significant for the IID case; however, `eFL_metis` brings more substantial accuracy gain for the non-IID case especially with more edge servers deployed.

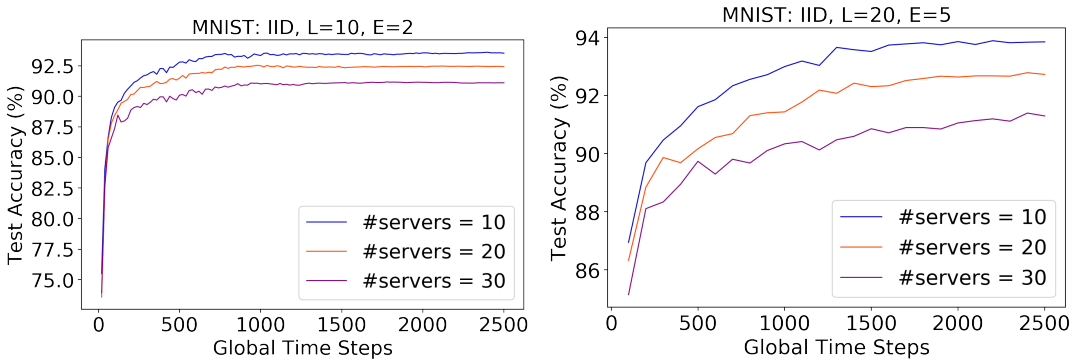
3.5.4 IID-ness at the edge vs. IID-ness at the local

Here we focus on the case that local data distribution is non-IID and investigate how eFL helps neutralize its effect. The edge IID-ness comparison of `eFL_metis` versus `eFL_rnd` is illustrated in Figure 12 for both MNIST and CIFAR10. In these figures, each plot shows the distribution of the training samples of the participants assigned to each server over the 10 classification labels; there are M (10 or 30) curves in each plot, where each curve represents a server.

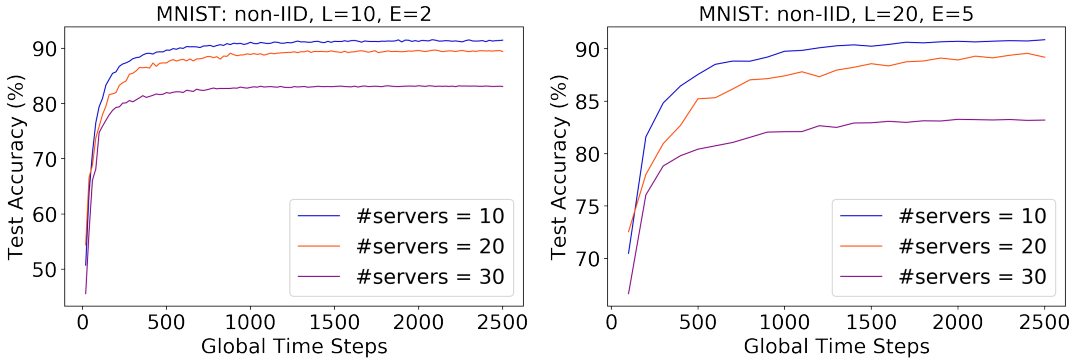
Good IID-ness is obtained if (1) these curves appear close to each other (ideally, each server should have the same distribution curve) and 2) the horizontal deviation should be small (ideally, each label should have the same number of samples; in our evaluation, the global dataset has identical numbers of samples for each label). `eFL_rnd` clearly has substantial deviations both horizontally and vertically. In contrast, `eFL_metis` produces a much more constant-line looking curve shape. It is also observed that edge IID-ness improvement of `eFL_metis` over `eFL_rnd` is more noticeable when there are more edge servers ($M = 30$ servers vs. $M = 10$ servers).

3.6 Conclusions

FL cannot scale when the number of participants is too large for the global server to aggregate. eFL improves scalability by using edge servers as regional aggregators, which, however, leads to degraded learning accuracy. We have shown that the edge server assignment is critical to minimizing this tradeoff. We have proposed a simple yet effective solution based on the idea that the local models to be aggregated by an edge server should be maximally diverse. This statistical diversity can be measured without violating data privacy. This solution has been shown in our evaluation study to outperform the de facto standard random assignment by up to 20% when tested on popular real-word datasets. The proposed assignment solution is especially helpful when more edge servers are deployed and the local training data distribution is non-IID. It does not require strong assumptions and is useful as a universal benchmark for comparing eFL algorithms.

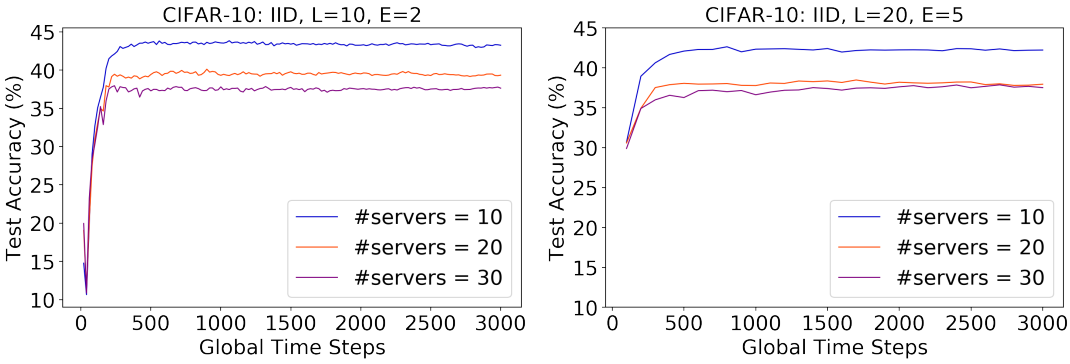


(a) IID case

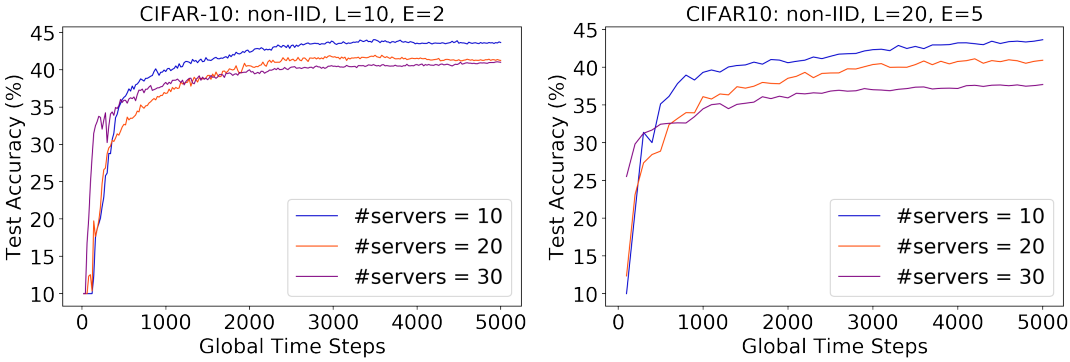


(b) Non-IID case

Figure 7: MNIST Dataset: The test accuracy of `eFL_metis` under the effect of the number of edge servers deployed for different combination settings of the number of local training updates and the number of edge training updates.

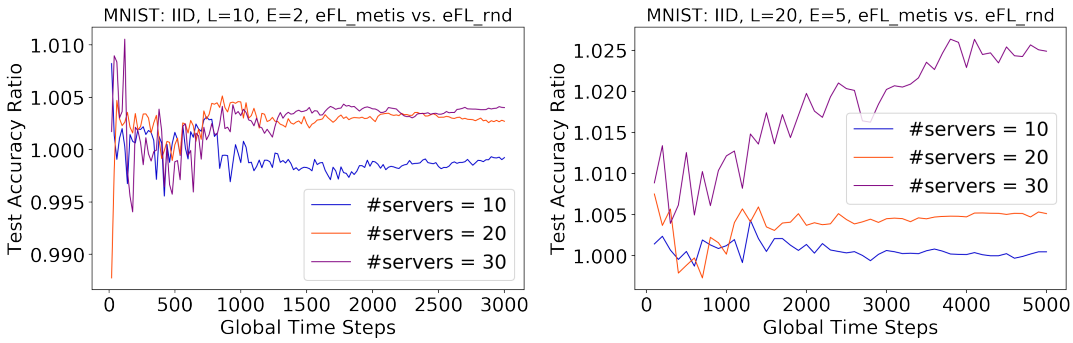


(a) IID case

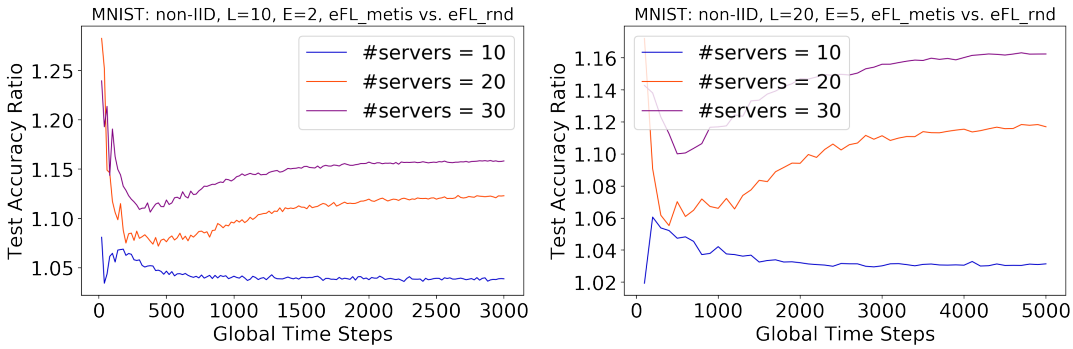


(b) Non-IID case

Figure 8: CIFAR10 Dataset: The test accuracy of `eFL_metis` for different combination settings of the number of local training updates and the number of edge training updates.

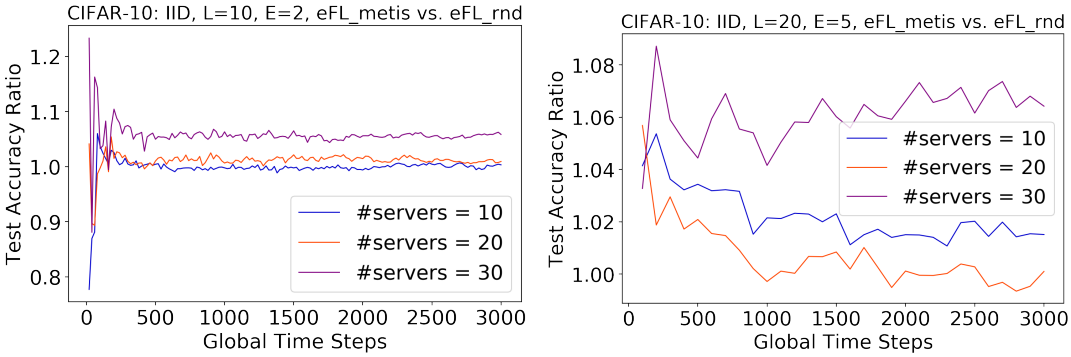


(a) IID case

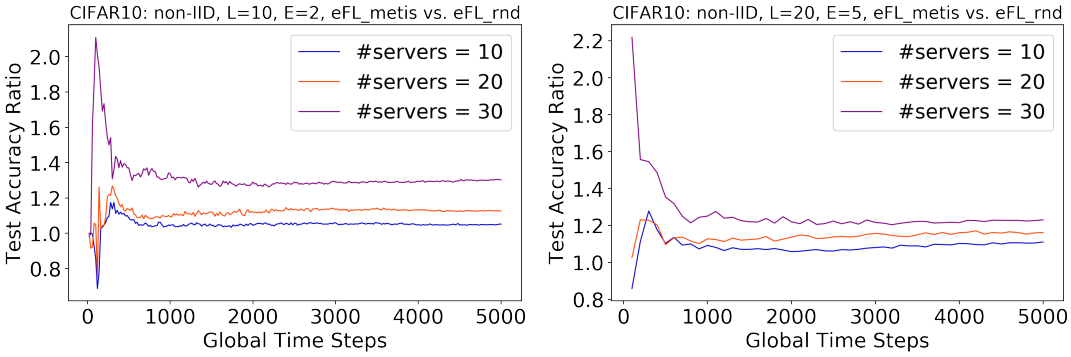


(b) Non-IID case

Figure 9: MNIST Dataset: The ratio of `eFL_metis`'s test accuracy to `eFL_rnd`'s test accuracy for different combination settings of the number of local training updates and the number of edge training updates.



(a) IID case



(b) Non-IID case

Figure 10: CIFAR10 Dataset: The ratio of `eFL_metis`'s test accuracy to `eFL_rnd`'s test accuracy for different combination settings of the number of local training updates and the number of edge training updates.

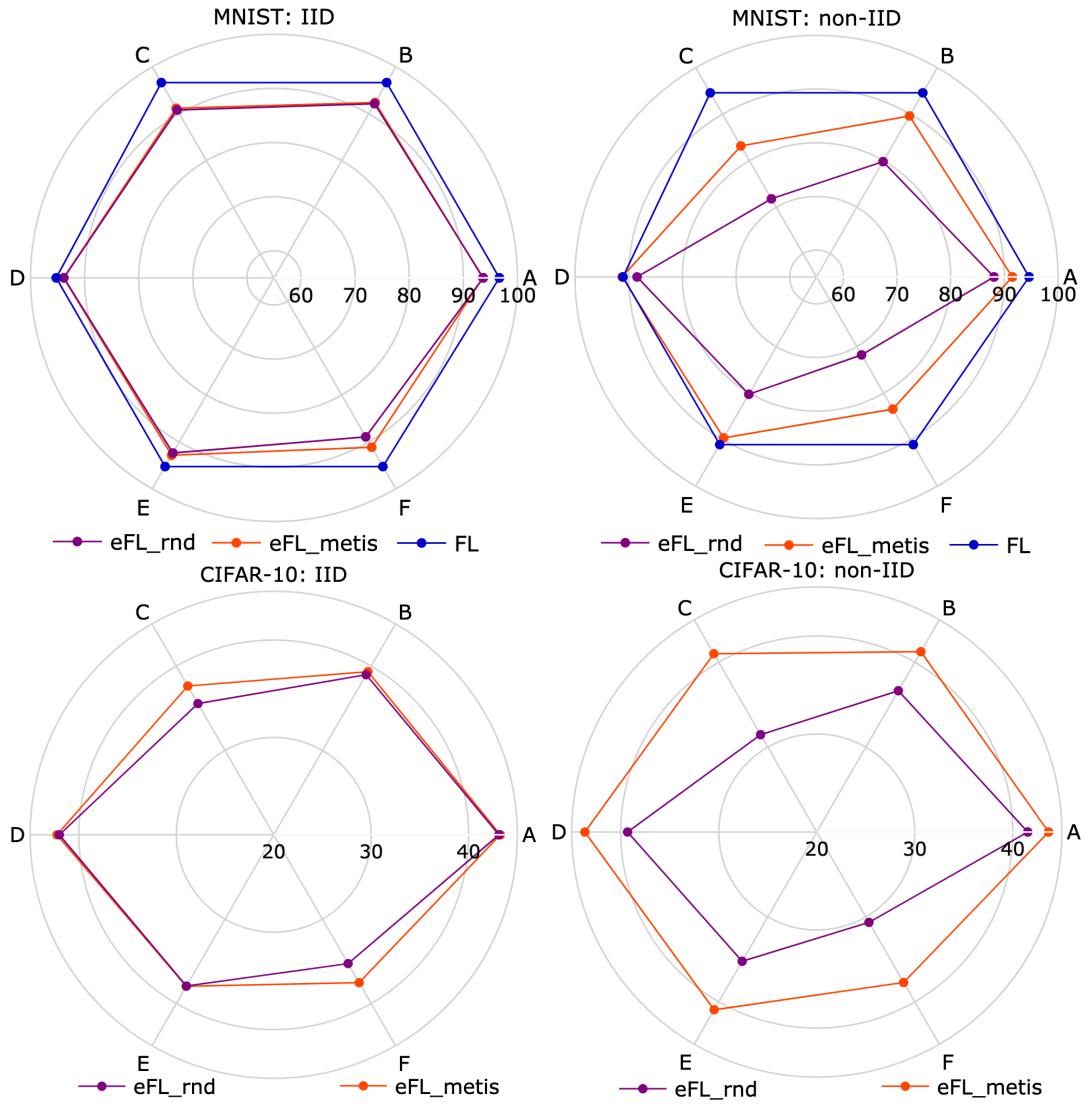


Figure 11: Accuracy summary comparing `eFL_metis`, `eFL_rnd`, and `FL` for different combination settings of the number of local training updates, the number of edge training updates, and the number of edge servers. Here, the meaning of the labels is as follows: A ($M = 10$), B ($M = 20$), C ($M = 30$) for case (L, E) =(10, 2) and D ($M = 10$), E ($M = 20$), F ($M = 30$) for case (L, E) =(20, 5).

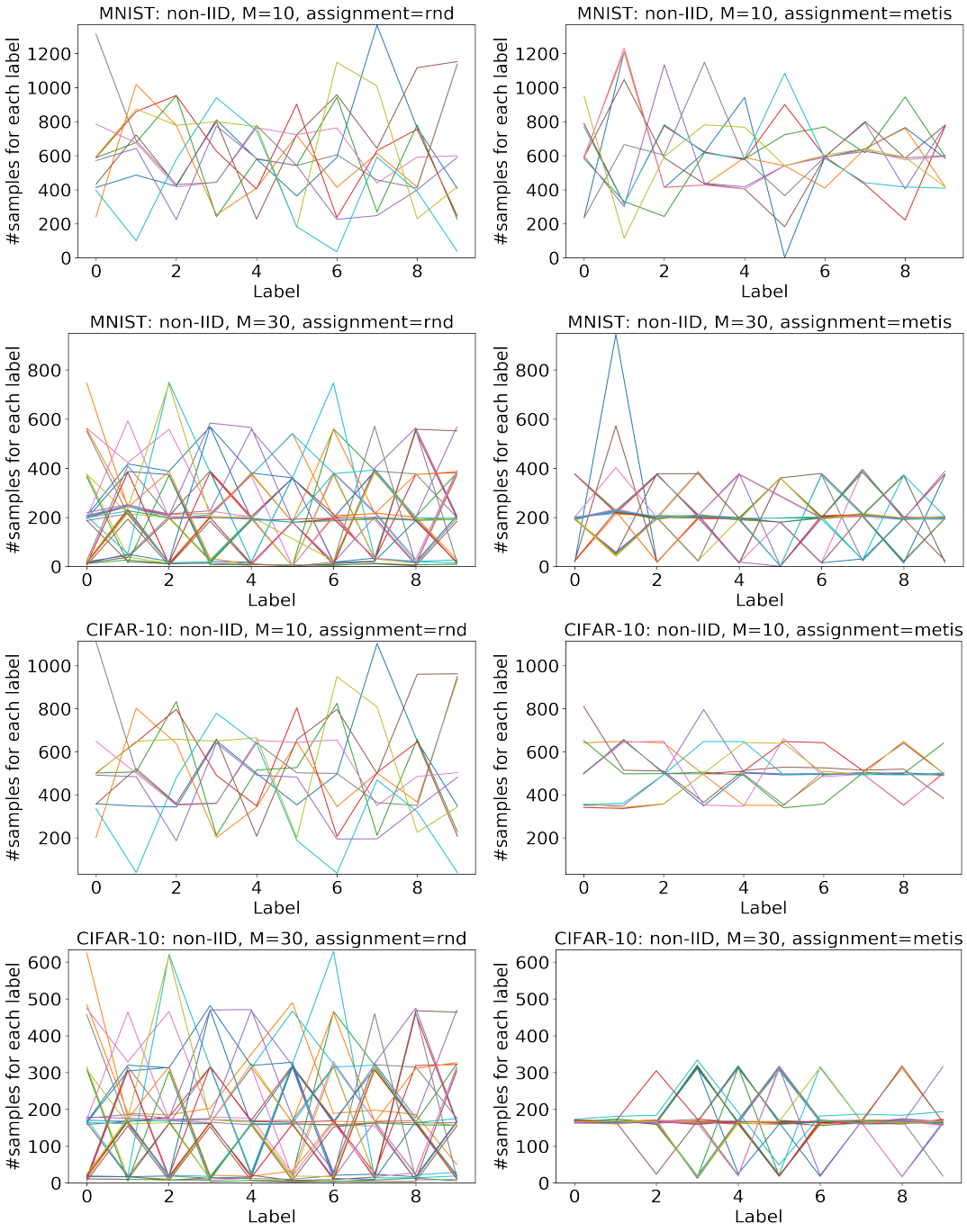


Figure 12: Edge IID-ness given IID training data across participants: Each plot shows the distribution of the training samples of the participants assigned to each server over the 10 classification labels. The x-axis is each label, the y-axis is the number of corresponding samples having this label, and each curve is for a server ($M = 10$ or $M = 30$).

CHAPTER 4

A STOCHASTIC GEO-PARTITIONING PROBLEM FOR MOBILE EDGE COMPUTING

The success of a mobile edge computing network depends critically on how to assign the edge servers to the user cells. The criterion for this assignment is application-specific. In many practical applications, the workload demanded by each cell is unknown and time-varying. So are the effective capacities of the servers. We need an assignment incurring minimum backhaul cost that is robust to these uncertainties. We also want to make the cells assigned to the same server form a geographically compact cluster. This challenge motivates us to introduce a novel stochastic geo-aware partitioning problem. As it is computationally hard, we propose a heuristic algorithm that can produce a range of solutions representing different tradeoffs between cost minimization versus geographical awareness. We evaluate the proposed algorithm using a real-world dataset.

4.1 Introduction

Mobile Edge Computing (MEC) [ETS14] has emerged as a viable technology for mobile operators to push computing resources closer to the users to avoid long-haul crossing of the network core, thus improving network efficiency and user experience. Originally initiated for cellular networks to realize the 5G vision, MEC has been generalized for broader wireless and mobile networks [MYZ17, AZT18]. It is becoming more of a

phenomenon with the Internet of Things; 5+ billion IoT devices would be connected to MEC by 2020 according to a recent forecast by BI Intelligence [New17].

In a nutshell, a typical MEC architecture consists of four layers of entities: the mobile devices (users), the base-stations (cells), the edge servers, and the cloud datacenter. The edge servers are introduced in the network edge which connects the base-stations to the network core, each server being an aggregation hub, or a mini datacenter, to offload processing tasks from the remote datacenter. Because the region served by an edge server, also known as a "cloudlet" [SBC09], is much smaller, a commodity virtualization-enabled computer can be used to run compute and network functions that would otherwise be provided by the datacenter.

A challenge with MEC is how to assign a given set of edge servers to the user cells to maximize the edge computing benefit. This challenge is not new outside MEC. Indeed, it belongs to the large body of work on distributed allocation of resources (virtual machines) in cloud computing; e.g., [HKL14, AL12, Man15, DHY17]. The MEC assignment problem is similar, however with unique constraints. Firstly, the MEC servers need be near the user side, not the datacenter side, and so the communication cost to be minimized is due to the use of the backhaul network towards the datacenter, not the front-haul towards the cells. Secondly, the geographic spread of the cells served by a MEC server should be a design factor [BC17, MSG15], which typically is not a priority for a distributed cloud solution.

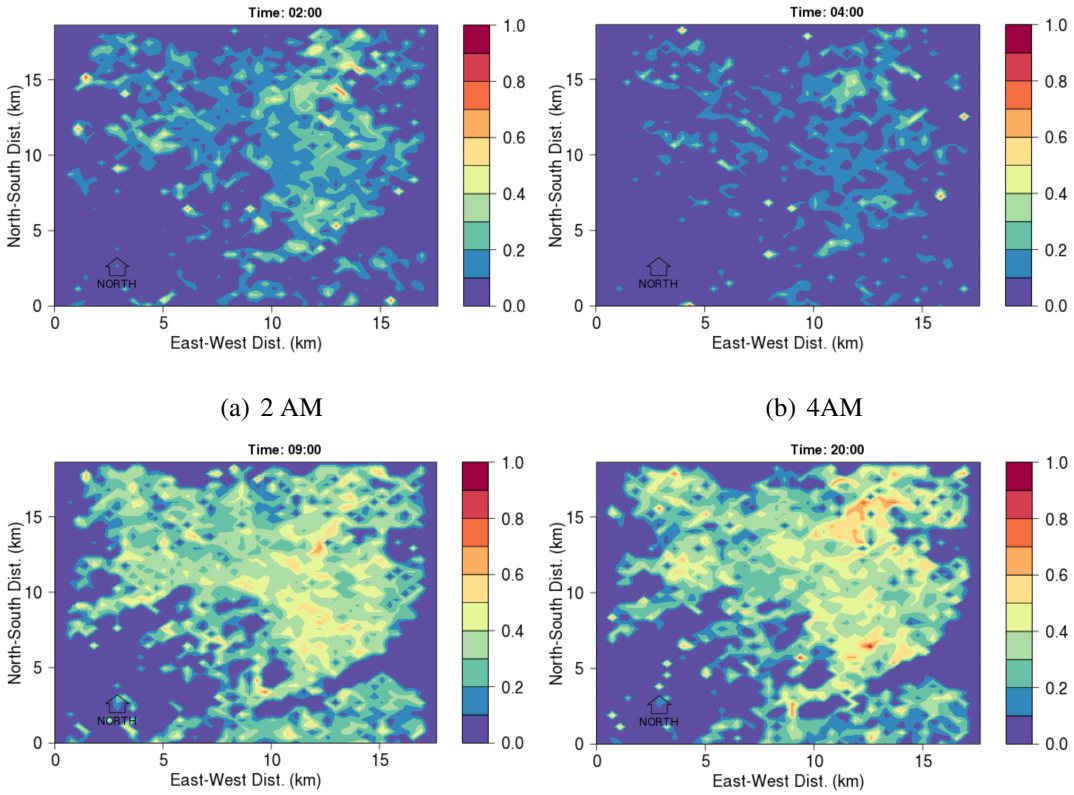
To address the MEC assignment challenge is application-specific. We focus on applications where requests for service are initiated by and processed for individual users (devices), which are plenty in the real world, e.g., video optimization [XLT17], content delivery [SHZ17], big data analytics [NRS17], roadside assistance [PGF18], and augmented reality [LHO18], to name a few. Applications that involve pairwise user-to-user

interactions such as peer-to-peer video streaming and multi-player online gaming are not considered.

Ideally, given a number of edge servers with limited capacity, a partition of the user cells into clusters (to assign to the edge servers respectively) should achieve there key goals: (1) Cost-Efficiency: The total backhaul cost is minimal. This cost is incurred for processing requests not fulfilled by the edge servers due to capacity limitation; (2) Geo-Compactness: The cells in each cluster should be close to each other. This makes the partition robust to user mobility and, also, results in the assigned server being geographically close to its users for efficient operation and shortened latency; and (3) Uncertainty-Robustness: The partition should remain efficient in the presence of time-varying workloads and the dynamics that a server's effective capacity may go up and down. Otherwise we would need to recompute the partition frequently to cope with these uncertainties.

What makes the research especially interesting is with the third goal: robustness. Indeed, while cost-efficiency is understandably a must-achieve in every MEC solution and geo-awareness has already been integrated in some recent partitioning techniques [BC17, MSG15, TV20, XLX16], we are aware of no prior research for the following questions: is there an efficient way to compute just one partition, no update needed, that can sustain a long period of time-varying workloads under time-varying effective server capacities? To demonstrate these dynamics, Figure 13 shows the cellular traffic intensity in a real-world cellular network, plotted for each base station at different hours on the same day. Obviously there is a significant workload variation over the time. The instability of effective server capacity in real-world computer systems is also well-known [IW11, KKS11].

Motivated by these questions, we introduce a new partitioning problem: Stochastic Geographically-aware Partitioning (StoGeoPar). Its objective is to optimize a stochas-



(a) 2 AM (b) 4AM

(c) 9AM (d) 8PM

Figure 13: Heat map of cellular traffic intensity for each base station at different hours during a day. The cellular network consists of 13,296 base-stations serving a large population in a medium-size city of China; courtesy of [CJQ15].

tic quantity formulated to incorporate all the three criteria of cost-efficiency, geo-compactness, and uncertainty-robustness. Our contributions are as follows.

- StoGeoPar is an original partitioning problem. Specifically, in the context of MEC, stochastic partitioning to represent uncertain workloads and/or capacities is not yet explored. So is the computation of a fixed partition optimized for an unknown future workload.

- We propose an approximate algorithm for StoGeoPar. Our approach is to optimize an approximative version of its original stochastic objective by fitting the time-varying input with a Gaussian distribution and applying a block-relaxation optimization method to solve it.

- We evaluate the proposed algorithm using a real-world mobile traffic dataset, showing that not only it is better than intuitive approaches, it can produce a range of partitions representing arbitrary tradeoffs between cost-efficiency and geo-compactness.

It is noted that, for an extended period of time-varying workloads, a typical approach is to adjust the partition dynamically upon each workload change. This is a theoretical perspective. In practice, because this approach is expensive if changes are frequent, one should only apply the repartitioning once after a sufficiently long period within which there may still be many workload changes. Our research is motivated by wondering if there is a way to compute a stable partition that is robust to such changes inside this period. Hence, our work is complementary to the typical approach. Indeed, given statistical parameters empirically obtained by monitoring the workload, we compute a partition robust to changes according to these parameters. When workload changes significantly, leading to new values of Gaussian parameters, we will re-run the algorithm as in the typical approach.

The remainder of this chapter is organized as follows. Related work is discussed in Section 4.2. The problem is introduced and formulated as a stochastic optimization problem in Section 4.3. The proposed algorithm is proposed in Section 4.4.2. The results of the evaluation study are analyzed in Section 4.5. The chapter concludes in Section 4.6.

4.2 Related Work

A MEC system, like every other finite computing system serving many resource-hungry requests, faces the challenge of how to assign resources to computing units to optimize hardware consumption and best satisfy application QoS requirements [CLL18]. The assignment problem in [WZT17] is aimed to support a set of applications with known request load and latency expectation and the challenge is to determine which edge servers to run the required virtual machines, constrained by server capacity and inter-server communication delay. In [BG17, WZL17], where only one application is being considered and consists of multiple inter-related components organizable into a graph, the challenge is how to map this component graph on top of the physical graph of edge servers to minimize the cost to run the application. In the case that edge servers must be bound to certain geographic locations, a challenge is to decide which among these locations we should place the servers and connect them for optimal routing and installation costs [CPS17, FA18b, XLX16, TV20].

The above works do not take into account the geographical spread of the region served by a server. The cells served by the same server can be highly scattered geographically, causing long latency and high management cost. While latency constraints can be set directly as in [XLX16], the works in [BC17, MSG15, TV20] introduce geographical constraints in the optimization of the server assignment. In [BC17], a spatial partitioning of the geographic area is proposed such that the cells of the same server are always contiguous while its capacity is not violated. In [MSG15], the cells served by each server must also be contiguous, but the objectives are different, for example, trying to minimize server deployment cost and cell-to-cell latency via the edge. Instead of enforcing the geographic contiguity in the partition, [TV20] aims to minimize the ge-

ographical spread of the cells served by each server. It is argued therein that these cells do not have to be contiguous; they just should not be too far from the serving server.

For time-varying workloads, the conventional or implicit approach in the literature is either to recompute the partition or make incremental adjustments. It is often required that workload rates be known prior to the update. In contract, our goal is to compute one single partition that stays unchanged but robust for a long sequence of randomly changing workloads and, importantly, without knowing the workload rates. The work in [VT19] considers time-varying workloads, but assumes fixed server capacities and is applicable only to workloads of interaction between user and user. In our problem setup, the server capacities can be time-varying and we focus on workloads of requests serving individual users.

Besides the assignment problem, the emergence of MEC has also given rise to other interesting problems that share some overlapping requirements but with different foci. For example, a network slicing problem is addressed in [LH19] for a multi-domain cellular edge network. Network slicing refers to building ”virtual clouds”, called *network slices*, sharing physical resources of the underlying mobile edge network. Functional isolation between different slices is a central goal for a network slicing solution, which does not apply in our partitioning problem.

The cloudlet placement problem is addressed in [FA19], whose goal is to determine the location to deploy and how much compute capacity to assign to each cloudlet. Recall that a cloudlet refers to the edge subnetwork of cells served by an edge server. This problem is not the same as our partitioning problem. The same authors also proposed a workload allocation method for a hierarchical cloudlet network to minimize the computing delay incurred by each cloudlet [FA18a]. In contrast, our research assumes a flat cloudnet/edge architecture.

A radio access network with a special scenario of having only two edge servers is considered in [RSN17], where a method is proposed for minimizing the service delay by controlling the computation delay via virtual machine migration and improving transmission delay via Transmission Power Control. When there are more clients and devices, the same authors also address the cloudlet activation problem (how to add cloudnet) [RSN18] to increase the scalability of an edge network. Wang et al. [WLK19b] propose to allocate computing and network resources adaptively to reduce the average service time in a dynamic MEC environment by adopting a deep learning approach. They also suggested a solution for IoT-based smart cities [WLK19a]. These aforementioned works choose the delay as the objective to minimize. The setting of our research problem is different not only in the nature of the problem (stochastic partitioning) but also the optimization objective (backhaul cost) and constraint (geographical awareness).

4.3 Problem Statement

The geographical area is partitioned into a set of N cells, indexed by the set $\mathscr{C} = \{1,2,...,N\}$, each cell $i \in \mathscr{C}$ served by a base-station (also referred to as base-station i). The meanings of "base-station" and "cell" are general, not necessarily understood in conventional meaning as in cellular networks; for example, as a Wi-Fi access router and its coverage area in Wi-Fi networks. Denote the location of each cell i by l_i (which can be 2D or 3D).

Let the time be discretized into time slots t = 1, 2, 3, ... The system workload at each time t consists of the workloads from all the cells at t. We model the time-varying workload rate of each cell i at time t as a random variable $X_i \in [0,\infty)$. Suppose that we can afford to deploy M edge servers, indexed by the set $\mathscr{S} = \{1,2,...,M\}$. Each server s has a compute capacity K_s. In practice, the effective capacity of a server may fluctuate

from time to time due to the dynamics of its resource usage (the server may involve in other computing tasks). Therefore, we also model K_s as a random variable in the range $[0, \infty)$.

The server assignment problem is how to partition the cellset $\mathscr{C}$ into M clusters, so that each cluster is assigned to a server to help process its workload. We represent this partitioning by a binary mapping $z : \mathscr{C} \times \mathscr{S} \to \{0, 1\}$ and denote by $z_{is} = 1$ the assignment of cell i to server (cluster) s and $z_{is} = 0$ otherwise. We need to compute z. In what follows, without explicitly specified otherwise, we use indices i, $j \in \mathscr{C}$ to denote a cell/base-station, and $s \in \mathscr{S}$ to denote a server.

4.3.1 Backhaul Cost

Minimizing the backhaul cost is the main reason for the adoption of mobile edge computing. There are other costs such as those involved at the edge servers we should also keep low. If we aim to minimize them together with the backhaul cost, there would be multiple costs to minimize, making the optimization problem a multi-objective problem difficult to solve. That said, we do not ignore these costs. We choose to address them by setting the constraint on the server capacity (the K_s quantity aforementioned). The server capacity is a general quantity that represents how much an edge server can process or afford, including computation, energy, storage, etc.

The backhaul cost is then defined as that incurred when the data center has to process the workloads assigned to a server that exceed its capacity. Given a partition z and the cell workloads X_1, X_2, ..., X_N, the workload demand put on server s (i.e., the amount of workload this server is supposed to process) is

$$W_s = \sum_{i=1}^{N} z_{is} X_i.$$

Not all the requests assigned to server s may be fulfilled by s due to its limited capacity K_s. As both workload demand and server capacity are uncertain, W_s may exceed the capacity K_s. In such a case, the requests that lead to violation of this capacity will be processed by the datacenter, incurring a backhaul cost of $W_s - K_s$. In any case, we can write the backhaul cost incurred by workloads assigned to server s as

$$\mathsf{cost}_s = \underbrace{\max\left(0, W_s - K_s\right)}_{overload\ amount\ of\ server\ s}.$$

The total backhaul cost is the sum of these amounts for all the servers:

$$\begin{aligned}\mathsf{cost} &= \sum_{s=1}^{M} \mathsf{cost}_s = \sum_{s} \max\left(0, W_s - K_s\right) \\ &= \sum_{s=1}^{M} \max\left(0, \sum_{i=1}^{N} z_{is} X_i - K_s\right). \end{aligned} \tag{4.1}$$

Ideally, we want a partition z such that when applied to a (future) time-varying workload the total backhaul cost is minimal.

4.3.2 Geographical Compactness

Our focus is on the cell partitioning, after which the servers will be deployed accordingly. Because the network delay is more or less an increasing function of geographical distance, a shorter distance from a server to a cell should result in shorter delay serving this cell. Ideally, a server would be placed at the center of its cluster, and so minimizing the average distance from this server to its assigned cells is equivalent to minimizing the average pairwise distance between these cells.

Consequently, for shorter latency and easier management, we should keep the geographical region of each cluster from spreading too far. We quantify the geospread of a cluster as the sum of pairwise squared distances of its assigned cells normalized by the

cluster size,

$$\mathsf{spread}_s = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} z_{is} z_{js} d_{ij}^2}{\sum_{i=1}^{N} z_{is}}. \tag{4.2}$$

where d_{ij} is the geographical distance between cell i and cell j (i.e., $\|l_i - l_j\|$ where l_i, l_j are the geographical location of cell i and cell j, respectively).

We want to minimize the total geospread for all the servers,

$$\begin{aligned}\mathsf{spread} &= \sum_{s=1}^{M} \mathsf{spread}_s \\ &= \sum_{s=1}^{M} \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} z_{is} z_{js} d_{ij}^2}{\sum_{i=1}^{N} z_{is}}.\end{aligned} \tag{4.3}$$

It is noted that this is the same objective as that of k-means, a classic clustering method.

4.3.3 Stochastic Optimization Formulation

Because X_i's and K_s's are random variables, cost (Eq. (4.1)) is also a random variable. To minimize this cost not knowing future workloads and server capacities, we aim to minimize its expectation,

$$\mathbb{E}[\mathsf{cost}] = \mathbb{E}\left[\sum_{s=1}^{M} \max\left(0, \sum_{i=1}^{N} z_{is} X_i - K_s\right)\right]. \tag{4.4}$$

We combine the two objectives, cost and spread, into a single objective and formulate the problem as follows, which we refer to as StoGeoPar (Stochastic Geo-Partitioning):

$$\min_{\boldsymbol{z}} \left\{\Omega = \lambda \times \mathbb{E}[\mathsf{cost}] + (1-\lambda) \times \mathsf{spread}\right\} \tag{4.5}$$

$$\text{s. t. } 1) \sum_{s=1}^{M} z_{is} = 1, \ \forall 1 \le i \le N \tag{4.6}$$

$$2)\ z_{is} \in \{0,1\}, \ \forall 1 \le i \le N, 1 \le s \le M \tag{4.7}$$

where coefficient λ is to tune the priority tradeoff between the backhaul cost and the geospread. Constraints (4.6) and (4.7) ensure that each cell must be assigned to only one server.

In the special case where $\lambda = 0$, StoGeoPar is precisely the optimization of Euclidean k-means clustering (which is NP-hard). In the other extreme case where $\lambda = 1$, the deterministic version of StoGeoPar can be cast as a Multiple Subset Sum Problem (MSSP) (which is NP-hard [CKP00]). Indeed, we can treat each cell i as an item with weight X_i and profit X_i and each cluster s as a bin of capacity κ_s. Then, we have an MSSP in which we must put N items in M bins to not violate their capacity while maximizing total packed profit (which is equivalent to minimizing the total unpacked profit). In the arbitrary case, StoGeoPar is a computationally hard stochastic optimization problem unexplored in the literature.

4.4 Solution Approach

The probabilistic objective Ω in Eq. (4.5) is not computationally-friendly. Hence, our approach is to approximate it with a non-probabilistic objective and accordingly solve a non-probabilistic optimization problem. Specifically, we propose to approximate workload rates X_i's and capacities K_s's with Gaussian distributions.

Why Gaussian? For a random distribution whose pattern is unknown, Gaussian is known to be the best fitting model as an approximation. Therefore, in search of a heuristic solution to our proposed stochastic optimization problem, we approximate the workload (X) and server capacity (K) as random variables of the Gaussian distribution. In practice, when these entities vary according to some unknown pattern, the first step is to use the Gaussian distribution to approximate them and the second step is to find the solution according to our proposed algorithm (to be presented below). The Gaussian approximation allows our solution framework to apply widely to many applications.

4.4.1 The Idea

For a Gaussian variable $X \sim \mathcal{N}(\mu, \sigma^2)$, its pdf and cdf are

$$\phi_X(x) = \frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right),\ \Phi_X(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$$

where ϕ and Φ are the pdf and cdf of the standard $\mathcal{N}(0,1)$,

$$\phi(x) = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}x^2},\ \Phi(x) = \frac{1}{2}\left[1 + \mathsf{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$$

and erf is the Gaussian error function

$$\mathsf{erf}(x) = \frac{2}{\pi}\int_0^x \mathrm{e}^{-t^2} dt.$$

The following lemmas are useful for our derivations.

Lemma 4.4.1. *Let $X \sim \mathcal{N}(\mu, \sigma^2)$. Then, we have the following nice closed-form*

$$\mathbb{E}[\max(0,X)] = \sigma\phi\left(\frac{-\mu}{\sigma}\right) + \mu\left(1 - \Phi(\frac{-\mu}{\sigma})\right) \tag{4.8}$$

or we can also write

$$\mathbb{E}[\max(0,X)] = \sigma^2\phi_X(0) + \mu\left(1 - \Phi_X(0)\right). \tag{4.9}$$

Proof. We have

$$
\begin{aligned}
\mathbb{E}[\max(0,X)] &= \int_0^\infty x\phi_X(x)dx \\
&= \int_0^\infty x\frac{1}{\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}dx \\
&= \int_{-\mu/\sigma}^\infty (y\sigma+\mu)\frac{1}{\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{y^2}{2}}d(y\sigma+\mu) \\
&= \int_{-\mu/\sigma}^\infty \frac{y\sigma}{\sqrt{2\pi}}\mathrm{e}^{-\frac{y^2}{2}}dy + \int_{-\mu/\sigma}^\infty \frac{\mu}{\sqrt{2\pi}}\mathrm{e}^{-\frac{y^2}{2}}dy \\
&= \sigma\int_{-\mu/\sigma}^\infty \frac{1}{2\sqrt{2\pi}}\mathrm{e}^{-\frac{y^2}{2}}d(y^2) + \mu\left(1-\Phi(-\mu)\right) \\
&= \sigma\int_{\mu^2/\sigma^2}^\infty \frac{1}{2\sqrt{2\pi}}\mathrm{e}^{-\frac{t}{2}}dt + \mu\left(1-\Phi(\frac{-\mu}{\sigma})\right) \\
&= \frac{-\sigma}{\sqrt{2\pi}}\mathrm{e}^{-\frac{t}{2}}\Big|_{\mu^2/\sigma^2}^\infty + \mu\left(1-\Phi(\frac{-\mu}{\sigma})\right) \\
&= \frac{\sigma}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\mu^2}{2\sigma^2}} + \mu\left(1-\Phi(\frac{-\mu}{\sigma})\right) \\
&= \sigma\phi\left(\frac{-\mu}{\sigma}\right) + \mu\left(1-\Phi(\frac{-\mu}{\sigma})\right) \\
&= \sigma^2\phi_X(0) + \mu\left(1-\Phi_X(0)\right)
\end{aligned}
$$

□

Suppose that the random workload rate of each cell i can be approximated with a Gaussian distribution with mean μ_i and variance σ_i^2 and the capacity of each server s can also be approximated with a Gaussian distribution with mean κ_s and variance γ_s^2.

$$X_i \sim \mathscr{N}\left(\mu_i, \sigma_i^2\right),\ K_s \sim \mathscr{N}\left(\kappa_s, \gamma_s^2\right).$$

Because a sum of Gaussian variables is also Gaussian,

$$\begin{aligned} Y_s &= \sum_{i=1}^{N} z_{is} X_i \\ &\sim \mathcal{N}\left(\sum_{i=1}^{N} z_{is}\mu_i, \sum_{i=1}^{N} z_{is}^2 \sigma_i^2 \right) \\ &= \mathcal{N}\left(\sum_{i=1}^{N} z_{is}\mu_i, \sum_{i=1}^{N} z_{is} \sigma_i^2 \right) \end{aligned}$$

(the last equality is because z_{is} is binary). Then,

$$\begin{aligned} Z_s &= Y_s - K_s \\ &\sim \mathcal{N}\left(\sum_{i=1}^{N} z_{is}\mu_i - \kappa_s, \sum_{i=1}^{N} z_{is} \sigma_i^2 + \gamma_s^2 \right). \end{aligned}$$

Applying Eq. (4.8), we have

$$\mathbb{E}[\max(0, Z_s)] = f\left(\sum_{i=1}^{N} z_{is}\mu_i - \kappa_s, \sqrt{\sum_{i=1}^{N} z_{is} \sigma_i^2 + \gamma_s^2} \right)$$

where function f denotes

$$f(\mu, \sigma) = \sigma \phi\left(\frac{-\mu}{\sigma} \right) + \mu \left(1 - \Phi(\frac{-\mu}{\sigma}) \right). \tag{4.10}$$

The backhaul cost expectation $\mathbb{E}[\mathsf{cost}]$ in the objective Ω of Eq. (4.5) becomes

$$\mathbb{E}[\mathsf{cost}] = \mathbb{E}\left[\sum_{s=1}^{M} \max\left(0, \sum_{i=1}^{N} z_{is} X_i - K_s \right) \right] \tag{4.11}$$

$$= \sum_{s=1}^{M} \mathbb{E}[\max(0, Z_s)] \tag{4.12}$$

$$= \sum_{s=1}^{M} f\left(\sum_{i=1}^{N} z_{is}\mu_i - \kappa_s, \sqrt{\sum_{i=1}^{N} z_{is} \sigma_i^2 + \gamma_s^2} \right). \tag{4.13}$$

Consequently, to minimize the objective cost Ω in StoGeoPar, which is probabilistic, our approach is to minimize the following non-probabilistic cost:

$$\min_{\boldsymbol{z}} \left\{ \Omega_G = \lambda \sum_{s=1}^{M} f\left(\sum_{i=1}^{N} z_{is}\mu_i - \kappa_s, \sqrt{\sum_{i=1}^{N} z_{is}\sigma_i^2 + \gamma_s^2} \right) + (1-\lambda) \sum_{s=1}^{M} \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} z_{is}z_{js}d_{ij}^2}{\sum_{i=1}^{N} z_{is}} \right\}. \tag{4.14}$$

4.4.2 The Algorithm

To solve the optimization problem (4.14), we propose an approximative algorithm based on block relaxation. Block relaxation is a fixed point method to optimize a multivariate function by making iterative local updates until a convergence. We start with an initial partitioning assignment and conduct iterative rounds of moving vertices between the clusters as long as the objective continues to improve. A nice property of block relaxation is that it always guarantees improvement as the number of iterations increases. With sufficiently many iterations, it may converge quickly to yield a close to optimal solution.

Let $\boldsymbol{z}^{(t)} = (\boldsymbol{z}_1^{(t)}, \boldsymbol{z}_2^{(t)}, ..., \boldsymbol{z}_N^{(t)})$ where $\boldsymbol{z}_i^{(t)} = (z_{i1}^{(t)}, z_{i2}^{(t)}, ..., z_{iM}^{(t)})$ be the partition in each iterative update round t; $\boldsymbol{z}^{(0)}$ denotes the initial partition. In each successive round t, we make the following updates in sequence:

- For i = 1 to N, update:

$$\boldsymbol{z}_i^{(t)} = \arg\min_{\boldsymbol{z}_i} \Omega_G\left(\boldsymbol{z}_1^{(t)}, ..., \boldsymbol{z}_{i-1}^{(t)}, \boldsymbol{z}_i, \boldsymbol{z}_{i+1}^{(t-1)}, ..., \boldsymbol{z}_N^{(t-1)} \right)$$

Consider the i^{th} update, in which we are finding a new cluster for cell i currently in cluster m. A feasible candidate for $\boldsymbol{z}_i$ is

$$\boldsymbol{z}_i \in \left\{ (\underbrace{0, ..., 0}_{s}, 1, \underbrace{0, ...0}_{M-s}) \mid 0 \le s \le M \right\},$$

meaning the moving of cell i to cluster s. If $s = m$, there will be no move. There are at most M candidates. We compute the cost Ω_G for each candidate and select the minimum. However, the computation of Ω_G can be expensive. Below we show how it can be done more efficiently.

In each update, we keep track of u_s and v_s the corresponding mean and variance of the workload sum in each cluster s,

$$u_s = \sum_{j=1}^{N} z_{is}\mu_j - \kappa_s$$
$$v_s = \sum_{j=1}^{N} z_{is}\sigma_j^2 + \gamma_s^2.$$

If the update moves cell i to a different cluster $s \neq m$, we update these quantities as follows:

$$u_m^{new} = u_m - \mu_i,\ v_m^{new} = v_m - \sigma_i^2 \tag{4.15}$$
$$u_s^{new} = u_s + \mu_i,\ v_s^{new} = v_s + \sigma_i^2 \tag{4.16}$$

The corresponding objective gain in terms of backhaul cost is

$$\mathsf{gain}_{cost} = f\left(u_m, \sqrt{v_m}\right) + f\left(u_s, \sqrt{v_s}\right)$$
$$-f\left(u_m^{new}, \sqrt{v_m^{new}}\right) - f\left(u_s^{new}, \sqrt{v_s^{new}}\right).$$

In each update, we also keep track of the number of cells and the geospread of each cluster s:

$$\mathsf{size}_s = \sum_{i=1}^{N} z_{is}$$
$$\mathsf{spread}_s = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} z_{is}z_{js}d_{ij}^2}{\sum_{i=1}^{N} z_{is}}.$$

After the update, these quantities are updated as follows:

$$\mathsf{size}_s^{new} = \mathsf{size}_s + 1, \mathsf{size}_m^{new} = \mathsf{size}_m - 1$$
$$\mathsf{spread}_m^{new} = \frac{\mathsf{spread}_m \cdot \mathsf{size}_m - \sum_{j=1}^{N} z_{jm} d_{ij}^2}{\mathsf{size}_m - 1}$$
$$\mathsf{spread}_s^{new} = \frac{\mathsf{spread}_s \cdot \mathsf{size}_s + \sum_{j=1}^{N} z_{js} d_{ij}^2}{\mathsf{size}_s + 1}.$$

The objective gain in terms of geospread is

$$\mathsf{gain}_{geo} = \mathsf{spread}_m + \mathsf{spread}_s - \mathsf{spread}_m^{new} - \mathsf{spread}_s^{new}.$$

The overall objective gain is

$$\mathsf{gain} = \lambda \times \mathsf{gain}_{cost} + (1 - \lambda) \times \mathsf{gain}_{geo}. \tag{4.17}$$

We will move cell i from its current cluster m to a new cluster s if that results in the maximal gain. Our block relaxation based algorithm is summarized in Algorithm 6, which we name the Gaussian Block Relaxation Algorithm. The number of iterations, L, in the outermost for-loop should be sufficiently large, for example, such that the objective gain at the end becomes negligible.

Our proposed algorithm is an application of the Block Relaxation Method, a well-known iterative method to find the optimum of an optimization problem. The convergence of our algorithm, therefore, is inherited from that of the Block Relaxation Method (the proof is provided in the paper [Lee94] by Leeuw). This proof is valid if we can find an optimum in each iterative step which applies in our case (line 5 of Algorithm 6).

4.5 Evaluation Study

We conducted an evaluation using the cellular traffic dataset made available by Chen et al. [CJQ15]. This dataset provides hourly HTTP traffic statistics at the base-station granularity during the period from August 19, 2012 to Aug 26, 2012. There are $N = 13,296$

Algorithm 6: Gaussian Block Relaxation Algorithm

Input: Geographical distances between cells, $\{d_{ij}\}_{N\times N}$; means and standard deviations, $\{(\mu_i, \sigma_i)\}_{i=1}^{N}$, of the workload rates; means and standard deviations, $\{(\kappa_s, \gamma_s)\}_{s=1}^{M}$, of the capacities

Output: Partition assignment $\{z_{is}\}_{N\times M} \subset \{0,1\}^{NM}$

Start with a random partition $\boldsymbol{z}$;

for *(L iterations)* **do**

    /* L: the larger value, the more improvement */

    for $i \leftarrow 1$ *to* N **do**

        m : current cluster of cell i ;

        Find $s^* \in \{1,2,...,M\}$ such that the quantity gain in Eq. (4.17) is positive and maximum;

        if *($s^* \neq m$)* **then**

            Remove cell i from cluster m;

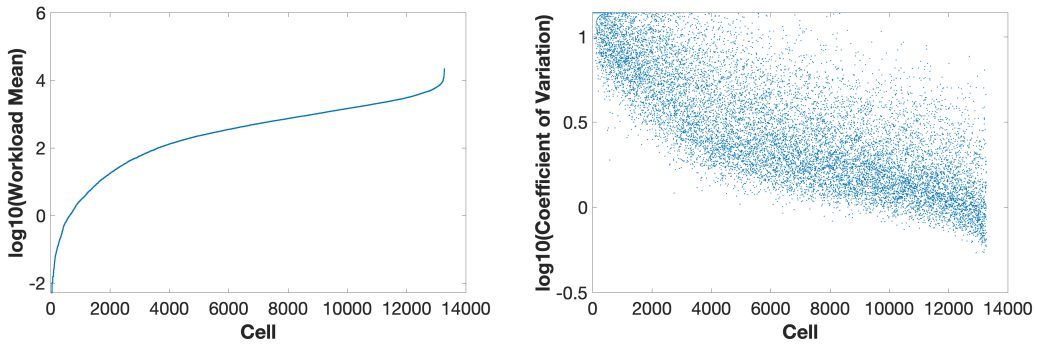
            Add cell i to cluster s^*;

For each cell i that is in cluster s, set z_{is} to 1;

base-stations serving a large population in a medium-size city of China. Specifically, we used the data about the number of transferred packets.

4.5.1 Setup

We compare the proposed block relaxation (BR) algorithm (Algorithm 6), hereafter referred to as BR/Gaussian, to the following benchmark: (1) Rand: a random partition, assigning a cell to a random server; (2) kMeans: a partition obtained from k-means, which is the best case scenario for geo-compactness; and (3) BR/Mean: a partition ob-



(a) Mean

(b) Coefficient of Variation

Figure 14: Statistics about the distribution of each cell's workload rate over the time. Plots are in log-linear scale, where the x-axis represents the cells which are sorted in the increasing order of workload rate mean. Left figure: the workload rate mean among the cells clearly varies widely. Right figure: the coefficient of variation varies from 0.8 (for "busiest" cells) to 10 (for "lightest" cells).

tained from a block relaxation algorithm similar to BR/Gaussian but using the mean information only but not variance information. By comparing to BR/Mean, our hypothesis is that the variance information is useful for finding a better partition.

In the dataset, there is a normalized longitude/latitude position for each cell, which we convert to a planar x/y position. The location of a cell is then defined as this x/y position and the Euclidean distance is used to measure the distance between cells. The time-varying workload X_i of each cell i is the number of transferred packets involving cell i. We collected this count at every hour during the entire 8-day period. From these samples (8 days $\times$ 24 hours/day = 192 of them for each cell), we computed the sample mean and variance for each cell; they serve as the mean μ_i and variance σ_i^2 for X_i. Figure 14 shows the mean μ_i and the coefficient of variation (cv) σ_i/μ_i for each cell's workload. It is observed that the distribution of μ_i spanning different cells varies

widely (Figure 14(a)). Also, as shown in Figure 14(b), the cv varies widely from 0.8 (for heaviest-workload cells) to 10 (for lightest-workload cells). It is implied that there is a wide variation not only for the workload demand of a single cell over the time but also across different cells.

We set the number of clusters (edge servers) to $M = 100$ which is reasonable given $N = 13,296$ cells. The measure for comparison is the empirical cost of the objective in Eq. (4.5) when applied to the actual 192 workload samples. For each server s's capacity at a specific hour, we generated a random value according to $K_s \sim \mathcal{N}(\kappa_s, \gamma_s^2)$, where $\kappa_s = \kappa \cdot \frac{\sum_{i=1}^{N} \mu_i}{M}$ and $\gamma_s = \gamma \times \kappa_s$. Various configurations are considered, where $\kappa \in \{0.7, 1.3\}$ (the higher, the more capacity) and $\gamma \in \{0.1, 0.3, ..., 0.9\}$ (the higher, the more time-varying for the capacity). For example, when $\kappa = 0.7$ and $\gamma = 0.1$, what it means in the context of MEC is that we deploy 100 servers with similar capacities, whose total can nominally handle 70% of the expected total workload and each server's capacity can expectedly vary within 10% of its mean.

It is noted that the objective function of our optimization problem combines two objectives that have different units (cost unit versus distance unit). As such, setting coefficient lambda to 0.5 does not mean giving equal priority to each objective. To balance these units, we applied a normalization method as follow. First, we generate a random assignment of cells to the servers. Let

$$A = \sum_{s=1}^{M} f\left(\sum_{i=1}^{N} z_{is}\mu_i - \kappa_s, \sqrt{\sum_{i=1}^{N} z_{is}\sigma_i^2 + \gamma_s^2} \right),$$

which is the backhaul cost part in Eq. (4.14), and

$$B = \sum_{s=1}^{M} \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} z_{is} z_{js} d_{ij}^2}{\sum_{i=1}^{N} z_{is}}$$

the geospread part in Eq. (4.14). Here, $\{z_{is}\}'s$ represent the random assignment. If coefficient λ is fair (i.e., same unit scale for both objectives), then A and B should have

similar values. In practice, they are not. So, instead of trying to minimize the original objective function of Eq. (4.14), we minimize

$$\Omega_G = \lambda B \sum_{s=1}^{M} f\left(\sum_{i=1}^{N} z_{is}\mu_i - \kappa_s, \sqrt{\sum_{i=1}^{N} z_{is}\sigma_i^2 + \gamma_s^2} \right) + (1-\lambda) A \sum_{s=1}^{M} \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} z_{is} z_{js} d_{ij}^2}{\sum_{i=1}^{N} z_{is}}.$$

Although this is not a perfect normalization method (which does not exist though), it is a simple method applicable to all applications.

4.5.2 Results

Figure 15 provides a perspective in which the four algorithms are compared in backhaul cost and geospread separately. Here we show the results for four cases: whether the server capacity is low ($\kappa = 0.7$) or high ($\kappa = 1.3$) and whether it is stable ($\gamma = 0.1$) or not ($\gamma = 0.9$).

Expectedly, kMeans offers the most geo-compact partition but incurs the worst backhaul cost. Rand has the worst geospread but much lower backhaul cost. This is because our simulation sets similar capacities for the servers and so Rand keeps the server load similarly balanced, hence a low backhaul cost.

While these two partitions represent extreme cases whose objective cost is not tunable, the proposed algorithm BR/Gauss offers a nice range of tunable partitions. When more priority is given to minimization of backhaul cost, i.e., increasing λ towards 1, BR/Gauss can produce a partition highly efficient in backhaul cost, which can be much better than that of Rand (it is noted that the plots in Figure 15 are log-log). When more priority is given to minimization of geospread, BR/Gauss can produce a partition highly geo-compact, which can approach that of kMeans. By tuning λ in-between, we can obtain a partition that represents a tradeoff between backhaul cost and geospread.

Another observation is that BR/Gauss provides better Pareto-efficient partitions than BR/Mean which ignores the variance information; a solution A is more Pareto-efficient than a solution B if A is no worse than B in any objective and strictly better in at least one objective. Also, BR/Mean is always worse than Rand in terms of backhaul cost, whereas by tuning λ large enough, BR/Gauss will incur less backhaul cost. It is therefore clear that the variance information helps BR/Gauss produce more desirable partitions.

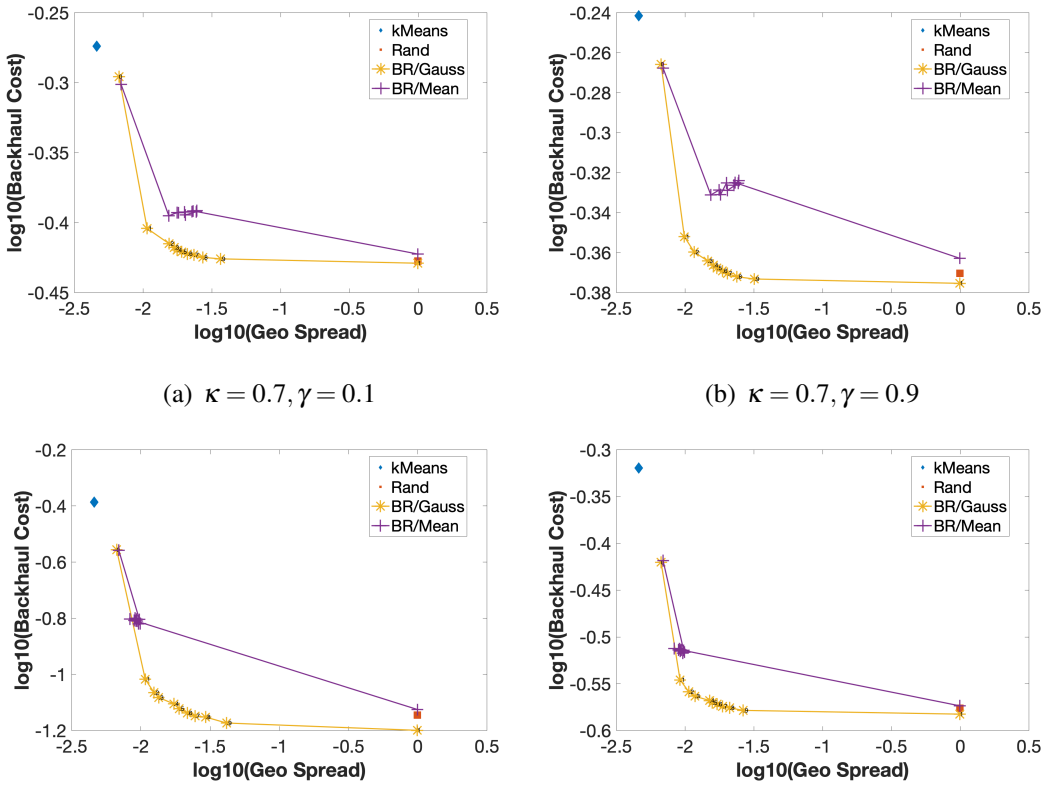
Figure 16 compares the four algorithms in terms of the empirical objective cost incorporating both backhaul cost and geospread by the tradeoff coefficient λ. For any λ, BR/Gauss is obviously the best solution, followed by BR/Mean. Rand (for small λ) or kMeans (for large λ) is always the worst, by a large margin. For example, when $\lambda = 0.2$ (Figures 16(a), 16(e)), BR/Gauss results in a cost roughly 8 times less than kMeans; when $\lambda = 0.8$, the improvement is almost 2 times for the low-capacity case (Figure 16(d)) and much more substantial for the high-capacity case (Figure 16(h)).

Comparing BR/Gauss over BR/Mean, the improvement is more noticeable when λ increases, putting more priority on backhaul cost. This is understandable because the influence of the variance information is in the backhaul cost, and so when we emphasize more on minimizing this cost, the role of the variance information becomes more important.

It is noted that kMeans always offers the best geospread. Figure 17 provides a 2D visualizations of the partition map resulted by BR/Gauss in comparison to that of kMeans. When $\lambda = 0$ (Figure 17(b)), i.e., all the priority is put for optimization of the geospread, BR/Gauss results in a partition that is geographically comparable to kMeans. As we increase λ to put more priority on minimizing the backhaul cost, the clusters start to get scattered, sacrificing geo-compactness for better overall objective cost, as we can see in Figure 17(d) for $\lambda = 0.8$. A nice feature of our proposed algorithm is its capability to tune the tradeoff coefficient λ to obtain any degree of geo-compactness.

4.6 Conclusions

Server assignment is an important problem in MEC. Its computational hardness is inherited from being a non-trivial instance of the combinatorial partitioning problem. We have introduced a novel stochastic setting for this problem in which we consider time-varying workloads and time-varying server capacities and seek a solution aimed to optimize backhaul cost, geographical compactness, and robustness to such uncertainty. Our solution approach is to model uncertainty information as Gaussian random variables and accordingly derive an objective function that is computationally friendly for optimization. Subsequently, we have proposed an approximate algorithm with the capability to produce a wide range of partitions representing different tradeoffs between backhaul cost and geographical compactness. Our research will result in less frequently having to reassign the cells to the edge servers due to workload dynamics.



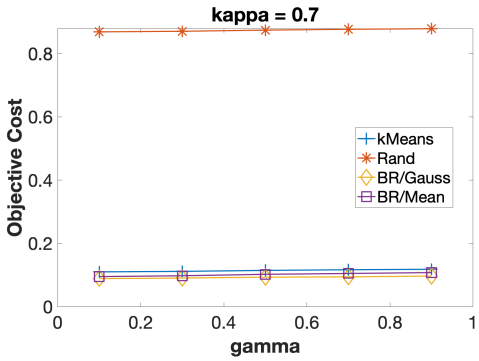
(a) $\kappa = 0.7, \gamma = 0.1$

(b) $\kappa = 0.7, \gamma = 0.9$

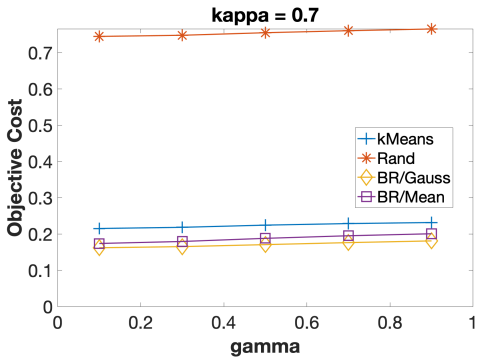
(c) $\kappa = 1.3, \gamma = 0.1$

(d) $\kappa = 1.3, \gamma = 0.9$

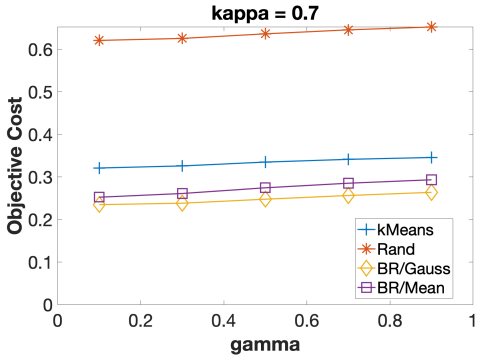
Figure 15: Pareto-efficiency comparison: The blue-colored point represents kMeans and the red point Rand. The yellow points, connected by line segments, represent BR/Gauss for different $\lambda \in \{0, 0.2, \ldots, 1\}$. The purple points, connected by line segments, represent BR/Mean for different $\lambda \in \{0, 0.2, \ldots, 1\}$.



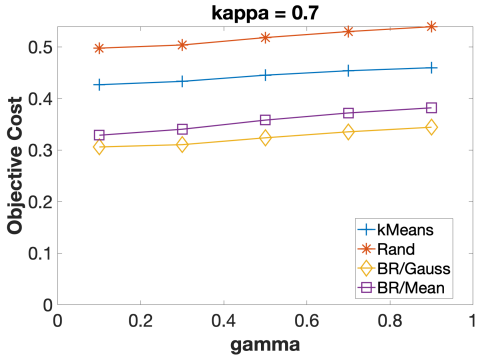
(a) $\kappa = 0.7, \lambda = 0.2$



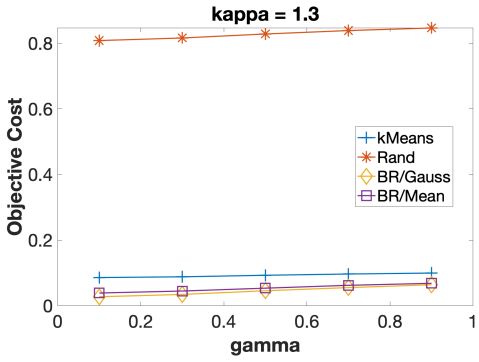
(b) $\kappa = 0.7, \lambda = 0.4$



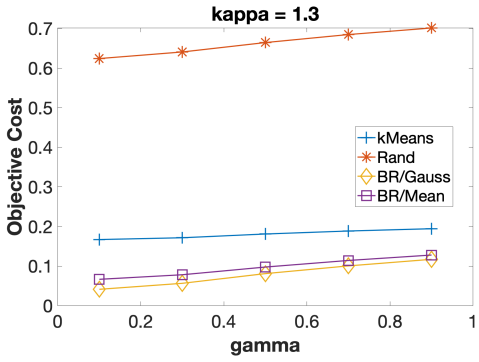
(c) $\kappa = 0.7, \lambda = 0.6$



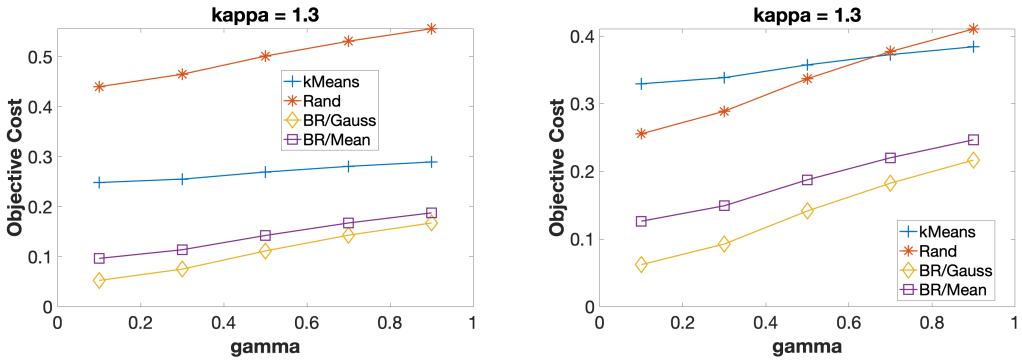
(d) $\kappa = 0.7, \lambda = 0.8$



(e) $\kappa = 1.3, \lambda = 0.2$

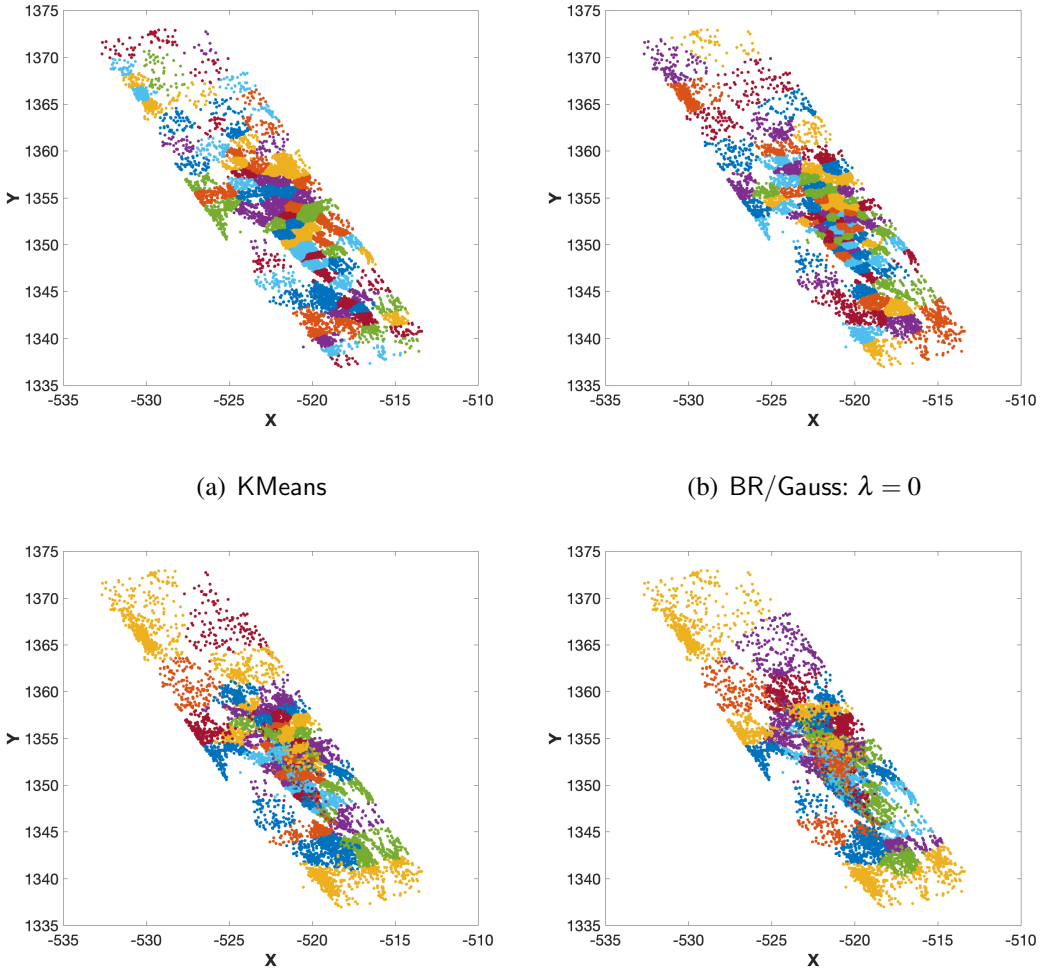


(f) $\kappa = 1.3, \lambda = 0.4$



(g) $\kappa = 1.3, \lambda = 0.6$ (h) $\kappa = 1.3, \lambda = 0.8$

Figure 16: Effect of the server capacity variation (by increasing γ) for different tradeoff cases between backhaul cost versus geospread. Top plots: when the capacity mean is fixed at $\kappa = 0.7$ (total capacity can nominally handle 30% less than the expected total workload). Bottom plots: when the capacity mean is fixed at $\kappa = 1.3$ (total capacity can nominally handle 30% more than the expected total workload).



(a) KMeans

(b) BR/Gauss: $\lambda = 0$

(c) BR/Gauss: $\lambda = 0.4$

(d) BR/Gauss: $\lambda = 0.8$

Figure 17: 2D visualization of the partition maps (each point represents a cell): by tuning parameter λ to set higher priority on geospread minization or backhaul cost minimization, we can obtain a partition ($\lambda = 0$) geographically comparable to kMeans or one that sacrifices geo-compactness for much better overall objective cost (e.g., $\lambda = 0.8$). In these plots, cells of the same cluster have the same color; however, since 100 clusters is too many for coloring, some colors are reused for different clusters.

CHAPTER 5

FUTURE WORKS

The dissertation is mainly focused on solving some optimization problems in decentralized learning, particularly in the context of Federated Learning and Edge Computing. Below I will summarize my work and explain how I will extend my research in the future.

5.1 Dissertation Summary

Federated Learning (FL) is a recent Machine Learning method for training with private data separately stored in local machines without gathering them into one place for central learning. It was born to address the following challenges when applying Machine Learning in practice: (1) Communication cost: Most real-world data that can be useful for training are locally collected; to bring them all to one place for central learning can be expensive, especially in real-time learning applications when time is of essence, for example, predicting the next word when texting on a smartphone; and (2) Privacy protection: Many applications must protect data privacy, such as those in the healthcare field; the private data can only be seen by its local owner and as such the learning may only use a content-hiding representation of this data, which is much less informative.

To fulfill FL's promise, I have addressed three important problems regarding the need for good training data, system scalability, and uncertainty robustness:

1. The effectiveness of FL depends critically on the quality of the local training data. We should not only incentivize participants who have good training data, but also minimize the effect of bad training data on the overall learning procedure. The first problem of my research is to determine a score to value a participant's contribution. My approach is to compute such a score based on Shapley Value (SV), a concept of cooperative game theory for profit allocation in a coalition game. In this direction, the main challenge is due to the exponential time complexity of the SV computation, which is further complicated by the iterative manner of the FL learning algorithm. I propose a fast and effective valuation method that overcomes this challenge.

2. On scalability, FL depends on a central server for repeated aggregation of local training models, which is prone to become a performance bottleneck. A reasonable approach is to combine FL with Edge Computing: introduce a layer of edge servers to each serve as a regional aggregator to offload the main server. The scalability is thus improved, however at the cost of learning accuracy. The second problem of my research is to optimize this tradeoff. This dissertation shows that this cost can be alleviated with a proper choice of edge server assignment: which edge servers should aggregate the training models from which local machines. Specifically, I propose an assignment solution which is especially useful for the case of non-IID training data which is well-known to hinder today's FL performance.

3. FL participants may decide on their own what devices they run on, their computing capabilities, and how often they communicate the training model with the aggregation server. The workloads incurred by them are therefore time-varying, unpredictably. The server capacities are finite and can vary too. The third problem

of my research is to compute an edge server assignment that is robust to such dynamics and uncertainties. I propose a stochastic approach to solving this problem.

5.2 Future Work

On the Federated Shapley Value work, there is room to optimize the computation further. Since FL is an iterative procedure trying to adjust the models of the previous rounds toward global model convergence, in the future work, I will explore ways to utilize the results in previous rounds to compute the SV faster in the future rounds. Also, the multi-issue decomposition in each round can be made more effective if we introduce weight coefficients the clusters of this decomposition to minimize the SV approximation error.

The work on Edge Federated Learning (eFL) can naturally be extended to cases where the edge servers and participants are limited by computing and communication constraints. I will investigate these extensions in the future. On the Edge Server Assignment work, I will go beyond the Gaussian distribution as the approximation model to incorporate uncertainty in the optimization. I will explore optimization methods other than block relaxation in order to further improve the partitioning result.

Computing SV for eFL is also an interesting problem. To the best of my knowledge, this problem has not been studied in the literature yet. Due to edge assignment, data valuation in eFL is more challenging than that in FL: different assignments would result in different SVs for the same data provider. I will investigate how my Federated SV algorithm can be revised to work for this setting.

REFERENCE LIST

[ACC14] Hossein Azari Soufiani, David M. Chickering, Denis X. Charles, and David C. Parkes. "Approximating the Shapley Value via Multi-Issue Decompositions." In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '14, p. 1209?1216, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[AHS22] Haftay Gebreslasie Abreha, Mohammad Hayajneh, and Mohamed Adel Serhani. "Federated Learning in Edge Computing: A Systematic Survey." *Sensors*, **22**(2), 2022.

[AL12] M. Alicherry and T. V. Lakshman. "Network aware resource allocation in distributed clouds." In *2012 Proceedings IEEE INFOCOM*, pp. 963–971, March 2012.

[AZT18] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie. "Mobile Edge Computing: A Survey." *IEEE Internet of Things Journal*, **5**(1):450–465, Feb 2018.

[BC17] Mathieu Bouet and Vania Conan. "Geo-partitioning of MEC Resources." In *Proceedings of the Workshop on Mobile Edge Communications*, MECOMM '17, pp. 43–48, New York, NY, USA, 2017. ACM.

[BCM18] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. "Analyzing Federated Learning through an Adversarial Lens.", 2018.

[BEG19] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. "Towards Federated Learning at Scale: System Design." In Ameet Talwalkar, Virginia Smith, and Matei Zaharia, editors, *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*. mlsys.org, 2019.

[BG17] Tayebeh Bahreini and Daniel Grosu. "Efficient placement of multi-component applications in edge computing systems." In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing, San Jose / Silicon Valley, SEC 2017, CA, USA, October 12-14, 2017*, pp. 5:1–5:11, 2017.

[BPF21] Samyadeep Basu, Philip Pope, and Soheil Feizi. "Influence Functions in Deep Learning Are Fragile." *ArXiv*, **abs/2006.14651**, 2021.

[CD19] Kevin Corder and Keith Decker. “Shapley Value Approximation with Divisive Clustering.” In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 234–239, 2019.

[CGT09] Javier Castro, Daniel Gómez, and Juan Tejada. “Polynomial Calculation of the Shapley Value Based on Sampling.” *Comput. Oper. Res.*, **36**(5):1726â1730, may 2009.

[CJQ15] Xiaming Chen, Yaohui Jin, Siwei Qiang, Weisheng Hu, and Kaida Jiang. “Analyzing and Modeling Spatio-Temporal Dependence of Cellular Traffic at City Scale.” In *Communications (ICC), 2015 IEEE International Conference on*, 2015.

[CKP00] Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. “The Multiple Subset Sum Problem.” *SIAM J. on Optimization*, **11**(2):308–319, February 2000.

[CLL18] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu. “Efficient Resource Allocation for On-Demand Mobile-Edge Cloud Computing.” *IEEE Transactions on Vehicular Technology*, **67**(9):8769–8780, Sep. 2018.

[CPS17] Alberto Ceselli, Marco Premoli, and Stefano Secci. “Mobile Edge Cloud Network Design Optimization.” *IEEE/ACM Trans. Netw.*, **25**(3):1818–1831, June 2017.

[CS04] Vincent Conitzer and Tuomas Sandholm. “Computing Shapley Values, Manipulating Value Division Schemes, and Checking Core Membership in Multi-Issue Domains.” In *AAAI’04: Proceedings of the 19th National Conference on Artificial Intelligence*, AAAI’04, p. 219â225. AAAI Press, 2004.

[CW80] R. Dennis Cook and Sanford Weisberg. “Characterizations of an empirical influence function for detecting influential cases in regression.” In *Technometrics ISSN 0040-1706*, p. 22(4):495?508, 1980.

[Den12] Li Deng. “The mnist database of handwritten digit images for machine learning research.” *IEEE Signal Processing Magazine*, **29**(6):141–142, 2012.

[DHY17] Hou Deng, Liusheng Huang, Chenkai Yang, Hongli Xu, and Bing Leng. “Optimizing virtual machine placement in distributed clouds with M/M/1 servers.” *Computer Communications*, **102**:107 – 119, 2017.

[eca] “Edge computing market value worldwide 2019-2025.”.

[ecb] "Number of new papers on Google Scholar related to edge computing worldwide 2010-2020.".

[ETS14] ETSI. "Mobile-Edge Computing: Introductory Technical White Paper." The European Telecommunications Standards Institute (ETSI), September 2014.

[FA18a] Q. Fan and N. Ansari. "Workload Allocation in Hierarchical Cloudlet Networks." *IEEE Communications Letters*, **22**(4):820–823, April 2018.

[FA18b] Qiang Fan and Nirwan Ansari. "Application Aware Workload Allocation for Edge Computing-Based IoT." *IEEE Internet of Things Journal*, **5**(3):2146–2153, 2018.

[FA19] Qiang Fan and Nirwan Ansari. "On Cost Aware Cloudlet Placement for Mobile Edge Computing." *IEEE/CAA Journal of Automatica Sinica*, **6**(JAS-2018-0416):926, 2019.

[FFZ21] Zhenan Fan, Huang Fang, Zirui Zhou, Jian Pei, Michael P Friedlander, Changxin Liu, and Yong Zhang. "Improving fairness for data valuation in federated learning." *arXiv preprint arXiv:2109.09046*, 2021.

[FFZ22] Zhenan Fan, Huang Fang, Zirui Zhou, Jian Pei, Michael P. Friedlander, and Yong Zhang. "Fair and efficient contribution valuation for vertical federated learning." *ArXiv*, **abs/2201.02658**, 2022.

[GBM21] Dhruv Guliani, Françoise Beaufays, and Giovanni Motta. "Training Speech Recognition Models with Federated Learning: A Quality/Cost Framework." In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pp. 3080–3084. IEEE, 2021.

[GKZ20] Amirata Ghorbani, Michael Kim, and James Zou. "A Distributional Framework For Data Valuation." In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3535–3544. PMLR, 13–18 Jul 2020.

[GZ19] Amirata Ghorbani and James Zou. "Data Shapley: Equitable Valuation of Data for Machine Learning." In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2242–2251. PMLR, 09–15 Jun 2019.

[HCP21] Dong-Jun Han, Minseok Choi, Jungwuk Park, and Jaekyun Moon. "FedMes: Speeding Up Federated Learning With Multiple Edge Servers." *IEEE Journal on Selected Areas in Communications*, **39**(12):3870–3885, 2021.

[HKL14] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee. "Online allocation of virtual machines in a distributed cloud." In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pp. 10–18, April 2014.

[IBM21] IBM. "Cost of a Data Breach Report 2021.", 2021.

[IW11] Rouba Ibrahim and Ward Whitt. "Wait-Time Predictors for Customer Service Systems with Time-Varying Demand and Capacity." *Operations Research*, **59**(5):1106–1118, 2011.

[JDW20] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gurel, Bo Li, Ce Zhang, Dawn Song, and Costas Spanos. "Towards efficient data valuation based on the Shapley value." 2020.

[Jov] Bojan Jovanovic. "Internet of things statistics for 2022 - Taking things apart.".

[KAT19] Pang Wei Koh, Kai-Siang Ang, Hubert H. K. Teo, and Percy Liang. *On the Accuracy of Influence Functions for Measuring Group Effects*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[KK98] George Karypis and Vipin Kumar. *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*, September 1998.

[KKS11] Emmanouil Kafetzakis, Kimon Kontovasilis, and Ioannis Stavrakakis. "Effective-capacity-based Stochastic Delay Guarantees for Systems with Time-varying Servers, with an Application to IEEE 802.11 WLANs." *Perform. Eval.*, **68**(7):614–628, July 2011.

[KMA21] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D?Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. "Advances and Open Problems in Federated Learning." *Foundations and Trends® in Machine Learning*, **14**(1?2):1–210, 2021.

[Kri09] Alex Krizhevsky. "Learning multiple layers of features from tiny images." Technical report, Department of Computer Science, University of Toronto, Canada, 2009.

[KZ22] Yongchan Kwon and James Y. Zou. "Beta Shapley: a Unified and Noise-reduced Data Valuation Framework for Machine Learning." *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain*, **151**, 2022.

[LCY21] Zelei Liu, Yuanyuan Chen, Han Yu, Yang Liu, and Lizhen Cui. "GTG-Shapley: Efficient and Accurate Participant Contribution Evaluation in Federated Learning." *ACM Trans. Intell. Syst. Technol.*, nov 2021. Just Accepted.

[Lee94] Jan de Leeuw. "Block-relaxation Algorithms in Statistics." In Hans-Hermann Bock, Wolfgang Lenski, and Michael M. Richter, editors, *Information Systems and Data Analysis*, pp. 308–324, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[LH19] Qiang Liu and Tao Han. "VirtualEdge: Multi-Domain Resource Orchestration and Virtualization in Cellular Edge Computing." In *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019*, pp. 1051–1060, 2019.

[LHO18] Qiang Liu, Siqi Huang, Johnson Opadere, and Tao Han. "An Edge Network Orchestrator for Mobile Augmented Reality." In *IEEE International Conference on Computer Communications (INFOCOM 2018)*, pp. 756–764, 04 2018.

[LHY20] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. "On the Convergence of FedAvg on Non-IID Data." In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[LLH20] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. "Federated Learning in Mobile Edge Networks: A Comprehensive Survey.", 2020.

[LYN20] Lingjuan Lyu, Jiangshan Yu, Karthik Nandakumar, Yitong Li, Xingjun Ma, Jiong Jin, Han Yu, and Kee Siong Ng. "Towards Fair and Privacy-Preserving Federated Deep Models." *IEEE Transactions on Parallel and Distributed Systems*, **31**:2524–2541, 2020.

[LZS20] Lumin Liu, Jun Zhang, Shenghui Song, and Khaled B. Letaief. "Client-Edge-Cloud Hierarchical Federated Learning." In *2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, June 7-11, 2020*, pp. 1–6. IEEE, 2020.

[MAM22] Naram Mhaisen, Alaa Awad Abdellatif, Amr Mohamed, Aiman Erbad, and Mohsen Guizani. "Optimal User-Edge Assignment in Hierarchical Federated Learning Based on Statistical Properties and Network Topology Constraints." *IEEE Transactions on Network Science and Engineering*, **9**(1):55–66, 2022.

[Man15] Zoltán Ádám Mann. "Allocation of Virtual Machines in Cloud Data Centers - A Survey of Problem Models and Optimization Algorithms." *ACM Comput. Surv.*, **48**(1):11:1–11:34, August 2015.

[MMR17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data." In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 20–22 Apr 2017.

[MSG15] R. Mijumbi, J. Serrat, J. L. Gorricho, J. Rubio-Loyola, and S. Davy. "Server placement and assignment in virtualized radio access networks." In *2015 11th International Conference on Network and Service Management (CNSM)*, pp. 398–401, Nov 2015.

[MYZ17] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. "A Survey on Mobile Edge Computing: The Communication Perspective." *IEEE Communications Surveys Tutorials*, **19**(4):2322–2358, Fourthquarter 2017.

[New17] Peter Newman. "IoT Platforms : Examining the Wide Variety of Software That Holds IoT Together." Source: BI Intelligence Store, January 2017.

[NN21] Lokesh Nagalapatti and Ramasuri Narayanam. "Game of Gradients: Mitigating Irrelevant Clients in Federated Learning." *Proceedings of the AAAI Conference on Artificial Intelligence*, **35**(10):9046–9054, May 2021.

[NRS17] S. Nastic, T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Ristov, and R. Prodan. "A Serverless Real-Time Data Analytics Platform for Edge Computing." *IEEE Internet Computing*, **21**(4):64–71, 2017.

[NSM20] Takayuki Nishio, Ryoichi Shinkuma, and Narayan B. Mandayam. "Estimation of Individual Device Contributions for Incentivizing Federated Learning." *2020 IEEE Globecom Workshops (GC Wkshps*, pp. 1–6, 2020.

[PGF18] G. Premsankar, B. Ghaddar, M. Di Francesco, and R. Verago. "Efficient placement of edge computing devices for vehicular applications in smart cities." In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, April 2018.

[PTB20] Shashi Raj Pandey, Nguyen H. Tran, Mehdi Bennis, Yan Kyaw Tun, Aunas Manzoor, and Choong Seon Hong. "A Crowdsourcing Framework for On-Device Federated Learning." *IEEE Transactions on Wireless Communications*, **19**:3241–3256, 2020.

[rot88] *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[RSN17] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato. "Hybrid Method for Minimizing Service Delay in Edge Cloud Computing Through VM Migration and Transmission Power Control." *IEEE Transactions on Computers*, **66**(5):810–819, May 2017.

[RSN18] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma. "Cloudlets Activation Scheme for Scalable Mobile Edge Computing with Transmission Power Control and Virtual Machine Migration." *IEEE Transactions on Computers*, **67**(9):1287–1300, Sep. 2018.

[SBC09] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. "The Case for VM-Based Cloudlets in Mobile Computing." *IEEE Pervasive Computing*, **8**(4):14–23, Oct 2009.

[Sha16] L. S. Shapley. *17. A Value for n-Person Games*, pp. 307–318. Princeton University Press, 2016.

[SHZ17] X. Song, Y. Huang, Q. Zhou, F. Ye, Y. Yang, and X. Li. "Content Centric Peer Data Sharing in Pervasive Edge Computing Environments." In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 287–297, June 2017.

[STW19] Tianshu Song, Yongxin Tong, and Shuyue Wei. "Profit Allocation for Federated Learning." In *2019 IEEE International Conference on Big Data (Big Data)*, pp. 2577–2586, 2019.

[SWM20] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. "Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data." *IEEE Trans. Neural Networks Learn. Syst.*, **31**(9):3400–3413, 2020.

[TDZ21] Duc A. Tran, Thuy T. Do, and Ting Zhang. "A Stochastic Geo-Partitioning Problem for Mobile Edge Computing." *IEEE Transactions on Emerging Topics in Computing*, **9**(4):2189–2200, 2021.

[TGY21] Siyi Tang, Amirata Ghorbani, Rikiya Yamashita, Sameer Rehman, Jared A. Dunnmon, James Zou, and Daniel L. Rubin. "Data valuation for medical imaging using Shapley value and application to a large-scale chest X-ray dataset." *Scientific Reports*, **11**(1), apr 2021.

[TV20] Duc A. Tran and Quynh Vo. "A geo-aware server assignment problem for mobile edge computing." *International Journal of Parallel, Emergent and Distributed Systems*, **35**(6):652–667, 2020.

[VT19] Quynh Vo and Duc A. Tran. "Probabilistic Partitioning for Edge Server Assignment with Time-Varying Workload." In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–8, 2019.

[WCZ16] Mike Wojnowicz, Ben Cruz, Xuan Zhao, Brian Wallace, Matt Wolff, Jay Luan, and Caleb Crable. ""Influence sketching": Finding influential samples in large-scale regressions." In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, dec 2016.

[WDZ19] Guan Wang, Charlie Xiaoqian Dang, and Ziye Zhou. "Measure Contribution of Participants in Federated Learning." *2019 IEEE International Conference on Big Data (Big Data)*, pp. 2597–2604, 2019.

[WKN20] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning." In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 1698–1707, 2020.

[WLK19a] Jiadai Wang, Jiajia Liu, and Nei Kato. "Optimal Edge Resource Allocation in IoT-Based Smart Cities." *IEEE Network*, **33**:30–35, 03 2019.

[WLK19b] Jiadai Wang, Jiajia Liu, and Nei Kato. "Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach." *IEEE Transactions on Emerging Topics in Computing*, **PP**:1–1, 03 2019.

[WRZ20] Tianhao Wang, Johannes Rausch, Ce Zhang, R. Jia, and Dawn Xiaodong Song. "A Principled Approach to Data Valuation for Federated Learning." In *Federated Learning*, 2020.

[WTS19] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. "Adaptive Federated Learning in Resource Constrained Edge Computing Systems." *IEEE Journal on Selected Areas in Communications*, **37**(6):1205–1221, 2019.

[WZL17] S Wang, M Zafer, and KK Leung. "Online Placement of Multi-Component Applications in Edge Computing Environments." *IEEE ACCESS*, **5**:2514–2533, 2017.

[WZT17] W. Wang, Y. Zhao, M. Tornatore, A. Gupta, J. Zhang, and B. Mukherjee. "Virtual machine placement and workload assignment for mobile edge computing." In *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, pp. 1–6, Sept 2017.

[XLT17] X. Xu, J. Liu, and X. Tao. "Mobile Edge Computing Enhanced Adaptive Bitrate Video Delivery With Joint Cache and Radio Resource Allocation." *IEEE Access*, **5**:16406–16415, 2017.

[XLX16] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo. "Efficient Algorithms for Capacitated Cloudlet Placements." *IEEE Transactions on Parallel and Distributed Systems*, **27**(10):2866–2880, Oct 2016.

[XNZ21] Yihao Xue, Chaoyue Niu, Zhenzhe Zheng, Shaojie Tang, Chengfei Lyu, Fan Wu, and Guihai Chen. "Toward Understanding the Influence of Individual Clients in Federated Learning." In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 10560–10567. AAAI Press, 2021.

[XYT21] Qi Xia, Winson Ye, Zeyi Tao, Jindi Wu, and Qun Li. "A survey of federated learning for edge computing: Research problems and solutions." *High-Confidence Computing*, **1**(1):100008, 2021.

[YAP20] Jinsung Yoon, Sercan Arik, and Tomas Pfister. "Data Valuation using Reinforcement Learning." In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10842–10851. PMLR, 13–18 Jul 2020.

[YLL20a] Yunfan Ye, Shen Li, Fang Liu, Yonghao Tang, and Wanting Hu. "EdgeFed: Optimized Federated Learning Based on Edge Computing." *IEEE Access*, **8**:209191–209198, 2020.

[YLL20b] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. *A Fairness-Aware Incentive Scheme for Federated Learning*, p. 393â399. Association for Computing Machinery, New York, NY, USA, 2020.

[YLL20c] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. "A Sustainable Incentive Scheme for Federated Learning." *IEEE Intelligent Systems*, **35**(4):58–69, 2020.

[ZLC21] Bowen Zhao, Ximeng Liu, and Wei-Neng Chen. "When Crowdsensing Meets Federated Learning: Privacy-Preserving Mobile Crowdsensing System." *ArXiv*, **abs/2102.10109**, 2021.

[ZLR20] Jingfeng Zhang, Cheng Li, Antonio Robles-Kelly, and M. Kankanhalli. "Hierarchically Fair Federated Learning." *ArXiv*, **abs/2004.10386**, 2020.

[ZWB21] Meng Zhang, Ermin Wei, and Randall Berry. "Faithful Edge Federated Learning: Scalability and Privacy." *IEEE Journal on Selected Areas in Communications*, **PP**:1–1, 10 2021.

[ZXL21] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. "Federated learning on non-IID data: A survey." *Neurocomputing*, **465**:371–390, 2021.

[ZZW20] Rongfei Zeng, Shixun Zhang, Jiaqi Wang, and Xiaowen Chu. "FMore: An Incentive Scheme of Multi-dimensional Auction for Federated Learning in MEC." *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pp. 278–288, 2020.